Research and Implementation of Dots-and-Boxes Game System

Shuqin Li
Department of Computer Science, Beijing Information& Science Technology University, Beijing 100192, P.R. China
Email: lishuqin_de@126.com

Dongming Li
Department of Computer Science, Beijing Information& Science Technology University, Beijing 100192, P.R. China
Email: dongmingbiti@126.com

Xiaohua Yuan
College of Information, Shanghai Ocean University，Shanghai 201306, P.R. China
Email: yuanxia8631_cn@sina.com

***Abstract*—This paper has studied the game rules, victory or defeat rules, and the key techniques of Dots-and-Boxes, and has designed and realized one 6×6 Dots-and-Boxes based on the representing method of Strings and Coins. By corresponding Dots-and-Boxes chessboard to strings and coins chessboard, the proposed representing method is not only intuitive and convenient in chessboard representation, but also has other 3 merits which include: (1) can reduce the size of representing data largely, and simplify the chessboard stored matrix, thus easy to store and analysis the data, (2) help to judge the key chains and rings among elements of chessboard, and (3) help to calculate the number of chains, and apply the long-chain rule. In the 4th machine gambling tournament took in 2010, the realized Dots-and-Boxes system gained the second best, which indicates the feasibility of the proposed chessboard representing method and the validity of the Dots-and-Boxes system.**

***Index Terms*—Computer game; dots and boxes; description of chessboard; Strings and Coins**

I. INTRODUCTION

Computer gambling, also called as Machine Gambling, is one of the important research fields of artificial intelligence (AI), and just from computer gambling began the research of early AI. Since computer gambling is not only simple and convenient, economical and practicable, but also rich of connotation, full of changeful thoughts, and can take effect quickly in short cycle when used to check the intelligence of computer, thus it is called as the drosophila in AI [1]. Computer gambling has induced many important method and theories into AI, and has produced a comprehensive influence on society and science, thus at a certain sense we can say that computer gambling is the token of AI development.

Dots-and-Boxes is one civilian chess popular in America. It is a point matrix game played by two persons. The chessboard of Dots-and-Boxes is showed in Fig. 1.

Fig.1 indicates the description of Dots and Boxes board, it is a 6*6 dots matrix or 5*5 boxes matrix, and it can be another size also. Rules of the game are that each player connect adjacent dots on alternate turns, it is required that player should only connect adjacent dots and should not make cater corners; the lines, which are also called edges, are not belong to any side, and players only need to take boxes into account. The player can get a box by connecting the four adjacent dots with lines, and when one player make a box, he(or she) can put his(or her) name in it, and move again, when all boxes are formed, the game is over, and the winner is the player who get the most boxes.

The complex degree of Dots and Boxes is moderate, and it is a typical adding chessman game that suit for some deeply research. Generally, computer gambling is composed by modules of board representation, estimation function, and searching algorithm, in which whether the representation method is efficient will directly affect the realization and the efficiency of other modules. In this paper, using the board representation method of Strings-and-Coin as reference, we proposed one new representation method of Dots and Boxes board, which can conveniently and quickly judge the phase of play, thus at a certain extent can improve the speed of estimating and searching during the play.

II. BASIC CONCEPTS OF DOTS AND BOXES

During the gambling, there can be more than one phase, and machine's every move can be taken as the combination of recognition model and gambling strategy. In analyzing of each phase, firstly only some commonly used board elements are recognized, for example the number and position of C-shape box, Long Chain, Short Chain, Cycle, Then abide by strategies of First-Hand, After-Hand, Odd Number, or Even Number to calculate the go position that favor to self side, and etc. For the convenience of description, here give the definitions of some main board elements.

Definition 1 Legality of Box

doi:10.4304/jsw.7.2.256-262

Refers to box whose grid coordinates are within the board, and will not cause any array subscript out of boundary.

Definition 2 Freedom degree of Box

Refers to unowned edge number of Box.

Definition 3 C-shape Box

Box formed by three edges.

Definition 4 Dead Box

Box if and only if can be caught by the current rival.

Definition 5 Dead Tree

Of each side in the gambling, there are a set of moves, and every move can catch one box which is called as Dead Box. Taking a Dead Box as a node, and link the adjacent nodes with one edge, then all nodes connected together will form a Dead Tree.

Definition 6 Chain

Chain is formed by a connected pairs of box in a legal, each box along the chain is legal and only enveloped by two edges. The number of boxes connected by the chain is the length of that chain.

Definition7 Long Chain

Refers to a chain whose length is more than 2.

Definition 8 Short Chain

Refers to a chain whose length is equal to 1or 2.

Definition 9 Cycle

Refers to a long chain whose length is equal or more than 4 and which is end to end.

III. THE NEW DESCRIPTION OF DOTS CHESSBOARD BASED ON STRINGS-AND-COINS

The traditional description of the Dots chessboard is a 6*6 dots grid, and a 2*2 sub grid is called a box, node(i,j) and node(k,l) are adjacent only when the expression |i-k|+|j-l|=1 be satisfied. When all the four edges of one box have been connected, then the box will be got by the player.

Strings-and-Coins is another game invented by Elwin Berlekamp and we can use it to generalize Dots-and-Coins to general graphs. The game is played on a graph, where the edges are strings and the vertices are coins.

The rules of game Strings-and-Games are simply, the players alternate cutting strings. When a player cuts all the edges surrounding a coin, he takes the coin and moves again. The player having the most coins wins the game.

Strings-and-Coins plays on the dual graph of Dots-and-Boxes, which means that: for any Dots-and-Boxes position, the corresponding Strings-and-Coins game is constructed by considering the boxes as coins, and the edges of the Dots-and-Boxes game as strings. Since placing an edge in the Dots-and-Boxes position will separate two boxes, so this has exactly the effect of cutting a string in the corresponding Strings-and-Coins position. From this point of view, the two games above are equivalent.

Strings-and-Coins has the advantage that it can be played on any graph, and it can also simplify certain Dots-and-Boxes observations. Inspired by the equivalent game Strings-and-Coins, we can deprive the ideas that providing another easier description of the board. Fig.1 shows the initial board in traditional way, while Fig.2 shows the Strings-and-Coins chessboard.

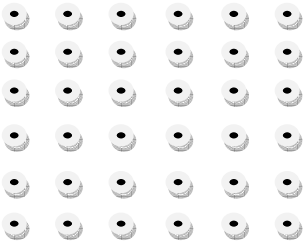


Figure 1. Dots-and-Boxes chessboard

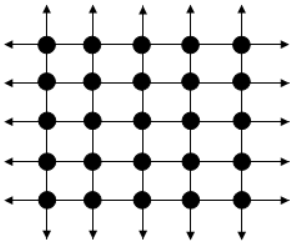


Figure 2. Strings-and-Coins chessboard

The corresponding relationships of the two types of boards are:

- Boxes correspond with coins

Fig.1 shows the initial Dots chess board of 6*6 type and we can get 5*5 boxes in this Dots-and-Boxes board, while Fig.2 shows the description of Stings-and-Coins chess board of 5*5 type. We can regard the boxes in Dots-and-Boxes as the coins in the Strings-and-Coins, so we get the corresponding relationship of boxes and coins.

- Edges correspond with strings

The initial chess board of Dots-and-Boxes has no connected edges at all, but in Stings-and-Coins chess board, each coin is connected with four strings, we can take the edges of the Dots-and-Boxes game as strings, but since placing an edge in the Dots game position will separate two boxes, so this has exactly the effect of cutting a string in the corresponding Strings-and-Coins position. Similarly each coin has a degree property which represents the count of strings connected to it.

Each edge in Dots will has the corresponding position in Strings, and Fig.3 shows how the edges correspond with strings.

Figure 3. Corresponding of edges and strings

In Fig.3, the left one is a box in Dots and it can have four edges, which are a, b, c and d, while the right one is a coin with four strings, which also are the a, b, c and d connected to it in Strings game. When placing the edge a in the left Dots box, it will has the same effect in the right coin by cutting the corresponding string, when all of the four edges of one box are connected, it represents that all of the four strings will be cut from the coin, and Fig.4 shows the coin's corresponding state while setting an edge a in the left box, we can see that in Strings-and-Coins, the string of this coin has been removed.

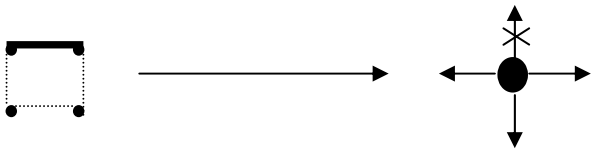


Figure 4. Corresponding when placing one edge in Dots

● The corresponding relationship of chains and cycles

Fig.5 shows that a chain has been made, and from what we introduced above we can easily learn how to get the corresponding graph in Strings-and-Coins game. The way is just to cut the corresponding strings and we will get the right chessboard description just like what Fig.6 shows. We can see that the representation of chain in Strings-and-Coins is clearer and it is vivid for us to judge if there's a chain exists or distinguish the different coins. Similarly, Fig.7 and Fig. 8 shows the corresponding relationship of cycles in two games.

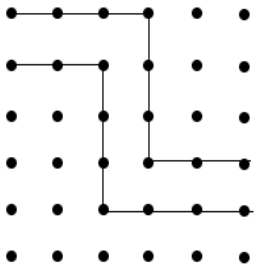


Figure5. Chain in Dots-and-Boxes

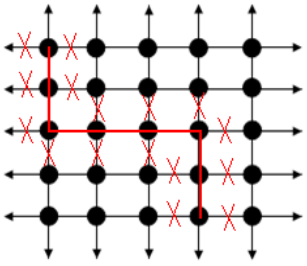


Figure 6. Corresponding Chain in Strings-and-Coins

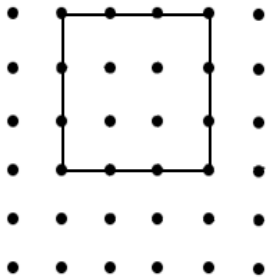


Figure 7. Cycle in Dots-and-Boxes

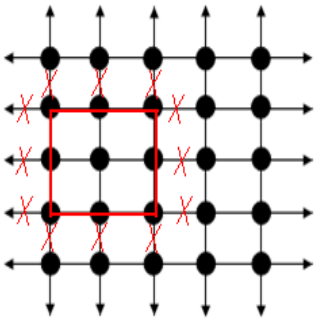


Figure 8. Cycle in Strings-and-Coins

IV. Strategy Analysis of Dots-and-Boxes System

Our system divides the whole play of Dots-and-Boxes into three phases, which include the Start, the Middle, and the Final, and adapts different strategies according to the characteristic of different phases. Since in the play of Dots-and-Boxes, Long Chain and Cycle are two frequently present formations, thus how to process Long Chain and Cycle is key to win, and the parity of Long Chain number is very important. In our system we abide by Berlekamp's Long-Chain theorem, which controls the number of Long Chain in the whole play.

Theorem 1 Long-Chain Theorem [4]

If total number of the board nodes is odd, then the Upper Hand side must form an odd number of Long Chain in order to win, and the After Hand side must form an even number Long Chain for win, and vice versa.

In a Dots-and-Boxes play, the Taking the chess side in the last run is the side which force the rival first enter into the Long Chain or Cycle. As to a 6×6 Dots-and-Boxes, which has 36 nodes (case of even number), the Upper Hand side will try hard to form an even number Long Chains, and the After Hand side then will try hard to form an odd number Long Chains. Since it is difficult to form more than one Long Chain, thus at the start, the Upper Hand side will consider to form 2 or 4 Long Chains, and the After Hand side will search moves that in favor of forming 3 or 5 Long Chains.

We take for that the Start Phase is the transition period for form Long Chain, Short Chain and Cycle, the Middle Phase is the formation period of the three formation, and the process of Chains and Cycles are in the Final Phase, because in the Start and Middle Phase Chains and Cycles will always not be processed, and only C-shape Boxes will be caught. In the Final Phase, Chessboard is always combined by all of or some of Long Chains, Short Chains, and Cycles.

(1) Process of Short Chain

Short Chain is the first problem need to deal with after entering the Final Phase, that is to own the conjunct edge that link two adjacent boxes, otherwise, the 2C active opportunities will be remised to the rival.

(2) Moves to Grid treatment

In order to force the rival first own the edge in Dead Tree, each side in the play will usually take the strategy of Moves to Grid. Dead Tree can be formed from Long Chain or Cycle, and to those two kinds of Dead Trees, the player can take different treatment.

A. To Dead Tree from Long Chain

When there only left two adjacent dead boxes not been captured, in which one is open, the other is closed and of C-shape, the Our side will take the open side of the open box, thus to form a 2C-shape, that is to remise two boxes to the rival.

B. To Dead Tree from Cycle

To Dead Tree from Cycle, the strategy of Moves to Grid must be adapted carefully. If adapting the strategy of Moves to Grid will depends on one precondition, that is when Boxes+2Doublecrosses≤12, adapt a Moves to Grid strategy, otherwise adapt an Ate Grid strategy(in each move catch one C-type box), thus finally catch all the dead box. In the above judge expression, Boxes is the number of boxes captured by the rival, Doublecrosses is the number of 2C-shape formation at the end of the play, which can be calculated by

$$\text{Doublecrosses}=\text{Chains}-1+2\times\text{Cycles}$$

Where Chains donates the number of Chains formed during the whole play, and Cycles is the number of Cycles at the end of the play. For example, when only four boxes left, in which two C-shape are closed, and the other two are open and not be captured, then one side of the gambling will take the public edge between two open boxes, thus to form 2 2C-shape formations and remise 4 boxes to the rival.

V. THE DESIGN AND IMPLEMENTATION OF DOTS CHESSBOARD BASED ON THE GAME OF STRINGS-AND-COINS

Based on the analysis of Strings-and-Coins, we can play Dots-and-Boxes in a general graph, and in section 3 and 4 we have provided the simply design of the description of chessboard, and in this section we will provide a simply implementation of the program playing the game Dots-and-Boxes. Firstly we will outline how to implement the suggested theory in a game playing system.

- Game rule implementation

We should make sure that both player obey the rules and make their moves when they are supposed to, like that you can choose if you make move first or make the computer move first, and when the player get boxes(coins) he/she should make another move until no boxes can be obtain in this turn.

So we should record all the moves in program made on the board to check for illegal moves, and record the score of both players. This is enough to play a complete game of Dots-and-Boxes or Strings-and-Coins, obeying the rules as described in sections above. Besides these, we also need to record some other features of the game in order to implement a game playing device. The structures that arise during the game must be represented after each move. During the game, just like an human player there is not anything that needs to be updated just before the computer's next turn, but whenever a move is made, the rivals will thinks all the time, not only at it's one's turn.

- Game representation

By the function DoMove, during the game we will record whose turn it is and what is the score. We can set who moves first at the beginning, if the human player first or the computer first. At the beginning of the game it prints an empty board. After the first move, it assigns the turn to another player and wait for the move, after which it switches turns again. All the moves will be recorded in a log file. After each move it will checks whether the turns should be switched to the other player, and assign the player to move next. For both players the same protocol is used every turn. First the current playing field is printed then the move is made, and after a move is made it is added to the move list, and is updated in the playing field, after which the structure list is updated by adjusting all structures to the new situation.

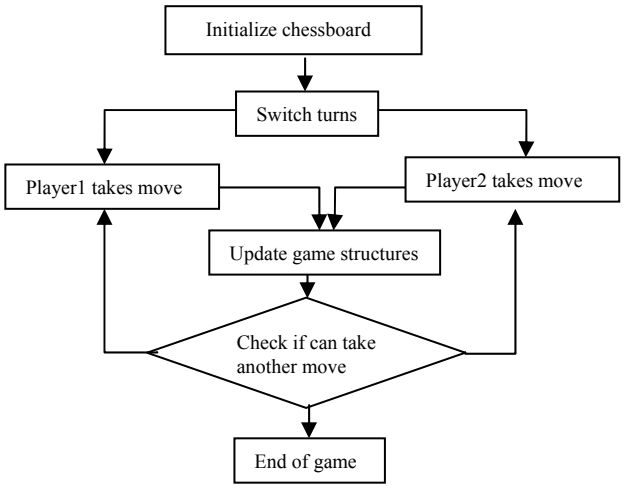


Figure 9. Simple flow of Dots-and-Boxes playing system

- Chessboard representation

The playing board consists of n*m boxes. The playing field is stored in three arrays: Square array, Horizontal edge array and Vertical edge array. In the square array, we will store the information of all the coins in the

chessboard, including their locations, and the counts of edges connected to the coins. The Square array is a 2-dimensional array, it stores the ground node existed in the Strings-and-Coins. Not like the nodes in the center area, the degree of each ground node is 1, which means at most only one edge connected to it. But each of other coins will has four edges connected to it, and it will have the degree of 4. When a node is formed, this also means the 4 edges are all removed from it, its property of degree will be zero, and this coin will be caught by one player.

The type of the elements in Square array is Node, the detailed definition of class Node is represented by Fig.10. In this class the pointer array edge [4] is used for storing the four strings connected to each coin, and the property degree represents the count of strings connected to one coin.

```
class Node
{
   public:
      Edge* NextEdge(Edge* Next);
      Node* NextNode(Edge* Next);

      int x,y; //the location of this coin
      Edge* edge[4]; //the four edges connected to this
                        coin

      int degree; //the count of  edges connected to it
      int owner; //the player got this coin
      int ground; //is it a ground node
}
```

Figure 10. Definition of class Node

We will store all the horizontal edges in the Horizontal edge array, to each edge store the information of its two nodes, and it is similarly to vertical edges. Each edge will have the property of removed, when it is removed from one connected coin, the value of this property will be one. Each array element in the horizontal array impliedly indicates the location of one horizontal string and also the information about the start node and end node of this edge. For example, horizontal[1][0] represents that this string is an horizontal string and it is connected by two nodes: square[0][1] and square[1][1]. Similarly, in the vertical edge array, element vertical[6][6] indicates that this string is a vertical string and it is connected by node square[1][0] and node square[1][1]. Fig.11 shows the detailed definition of Edge class.

```
class Edge
{
   public:
      int length;
      Node* node[2];
      Edge* next;
      Edge* prev;
      int removed;
}
```

Figure 11. Definition of class Edge

The two edge arrays are important because they provide the full information of edges in this board, and we will use this information to form the game structures like chains or cycles.

In the above description of chessboard, we store the coins and edges separately because it is efficient and simply for program to manage.

- Game structure representation

In Dots-and-Boxes game, count of chain is key to the player for win a Dots game, and Chain Rule tells how many chains one player should make to force his(or her) opponent to open the first long chain or cycle, that is:

If it is an odd number of dots, then the first player should make an odd number of chains and the second player should make an even number of chains. If it is an even number of total dots, then the first player should make an even number of chains and the second player should make an odd number of chains.

For a board sizes of 6*6 dots (5*5 boxes), since the first player should make an even number of chains, and the second player should make an odd number of chains, so the chain is an important aspect of wining the Dots game. In our board description, it is important for the player or program that how to inspect if the chain is formed after one move and get the count of chains.

All structures that are discovered during the game will be stored in a corresponding list. Each structure will have its own list, that is, the chain will be stored in the chain list, and cycle will be stored in the cycle list. Chains or cycles will be stored in the array of Chains[] or Cycles[], so we can easily get the chain and the count of chains or cycles which will be used in other methods. Each new structure discovered during the game will be stored and the full structure of the game will also be changed.

Keys of these lists are the different kinds of structures we distinguished. We record the total number of current and past structures. We keep a list of the following different structures: chains of all lengths, loops of all lengths, joints, handouts. The values that these keys correspond with are arrays containing the boxes forming that particular structure. In case of joint, the array contains the joint box and the structure names connected to that joint.

Because structures of particular lengths and shapes have important properties, which have influence on the strategies we need to follow, so we make a small refinement if we want to check for these strategies. The type of 1-chains and 2-chains are special because they can be offered as a hard-hearted handout, so these types of chains are stored in separate arrays. The third refinement is the structure refers to a possible loop. This is a chain from which the first and last box connected to the same joint. This means that such structures are most likely to be played as a loop, so it is useful to count possible loops as loops and not as chains. For each type of chain, we keep a list, which records the chain number of each chain type.
By our proposed chessboard representation, the core code used to judge Chain and Cycle is listed below Fig.12.

```
k=newedge->length-1;
  if(k<3)
    {
       INSERT(newedge, &moves[k]);
     }
   else
     {
         Edge *rgedge;
           if (newedge->node[0]= =newedge->node[1])
                 rgedge=loops;
            else      if(newedge->node[0]->ground      &&
newedge->node[1]->ground)
                 rgedge=chains;
             else
              rgedge=strings;
       }
```

Figure 12. the core code used to judge Chain and Cycle

where k donates the number of boxes along a Chain or in a Cycle. Since the number of boxes incluced in Chain or Cylce is at least 3, thus in the code firstly there is a check of k, that is if k>=3, then judge the end nodes node[0] and node[1]of the long edge. If node[0]=node[1], that shows the long edge is one Cycle, otherwise if node[0] and node[1] are peripheral nodes , then this long edge is one Chain, and otherwise, this long edge is only one common long edge in the board.

- Searching Algorithms

In computer gambling of this paper, we use an Alpha-Beta pruning algorithm of negative max form to find out the most favorable move. The algorithm is that: the value of father node is the opposite number of the max of all child nodes, lest that odd level get the minimal and even level get the maximum, thus $F(v)$ will be

$$F(v) = -\max\{-F(v_1), -F(v_2), \cdots, -F(v_n)\}$$

Where $F(\cdot)$ is the estimated value of one node, and $v_1, v_2, \cdots, v_n$ are the child node of node v. Here we integrate the Alpha pruning and Beta pruning to a same pruning-Beta pruning, whose algorithm is described in Fig. 13.

- Isomorphism

Since the representation result of Dots-and-Boxes chessboard is a matrix of size n*m, thus Dots-and-Boxes games have an important advantage; of isomorphism. From the new board description in section 3 we can see this, because of the symmetry in the field, to a certain box position there exist corresponding position in the board. Boxes at corresponding positions are actually similar as its equivalent, and can be treated by the same way in estimation or calculation. In Fig. 14 we have described all the isomorphic positions of the chessboard, and we can deduce three simple rules stated below.

As to a grid matrix of n*m (n and m are not equal), three positions, that are 2~4 in Fig. 14, are the equivalent to the original position 1 in Fig. 14, these equivalent positions can be obtained by rule 1 and rule 2.

```
int nega_alpha_bata(int alpha,int beta,int depth)
   int  val;
   // processing recursive export
       ……
GenaAllmove();
while (m=GetnextMove())
{
  MakeMove(m);
  Val=-nega_alpha_beta(-beta,-alpha,depth-1);
   UnmakeMove(m);
   if (alpha<val){
      alpha=val;
      if (bete<=val)
          return val;
          }
    }//end_while
    Return alpha;
}
```

Figure 13. Alpha-Beta searching algorithm of negative max form

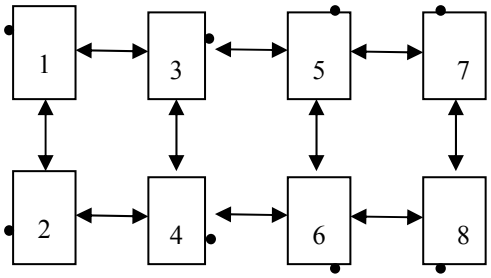


Figure 14. Representation of symmetry in the Dots-and-Boxes grid.

Rule1: Symmetric about X-axis

If two boxes positions are symmetric about X-axis then they are the equivalent.

Rule 2: Symmetric about Y-axis

If two boxes positions are symmetric about Y-axis then they are the equivalent.

And as to chessboard of n*n grids, we need to use a third rule to describe the equivalent positions.

Rule3: Symmetric about line x=y

If two boxes positions are symmetric about the line of x=y then they are the equivalent.

As we see in Fig. 14 the four new positions, that are 5-8, are the same as the four we already obtained for the n*m grid but mirrored in the x=y axis.

Of course this can be possible only if the n and m dimension are the same, or the grid cannot be mirrored in the x and y axis.

Based on the above analysis, we can take the advantage of isomorphic propriety to cut off leaves and branches from search trees which are equivalent to the already found positions, thus to reduce unnecessary searching and computation and improve the efficiency of the system.

VI. CONCLUSIONS

This paper has studied the game rules, victory or defeat rules, and the key techniques of Dots-and-Boxes, and based on Strings and Coins representation has designed and realized one 6×6 Dots-and-Boxes computer gambling system, which includes modules of chessboard representation, rule description, and searching algorithms. Since chessboard representation will directly influence the function and efficiency of the left two modules, so this paper focused on chessboard representation, and proposed a new chessboard representation method referring a Dots-and-Boxes board to a Strings-and-Coin board, which is more intuition and convenient. Because compared with common board representation, the proposed method can reduce the representation data size largely, it is more convenient for judging the formation of Long Chain, Short Chain and Cycle, and can improve the system efficiency remarkably. Although the system realized by the proposed method got the second best in the 4th computer gambling tournament took in 2010, there need further studies on and improvement of the set of start library and evaluation of chess game.

ACKNOWLEDGMENTS

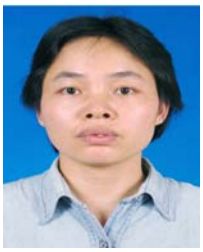
This research is sponsored by the Funding Project for Academic Human Resources Development in Institutions of Higher Learning under the Jurisdiction of Beijing Municipality (PHR201007131), by the Funding Project for Graduate Quality Courses Construction of Beijing Information & Science Technology University, and by the Funding Project for Graduate Science and Technology Innovation Projects of Beijing Information & Science Technology University.

REFERENCES

[1] Xu Xinhe, Deng Zhili, Wang Jiao. Challenging issues facing computer game research. Caai Transactions on Intelligent Systems, 2008, (4): 288−293.
[2] Lian Lian, Xu Xinhe, Zhang Xuefeng, Yan Ning. Key Technologies Analysis of Dots and Boxes Game System.Progress of Artificial Intelligence in China, 2009: 19-724.
[3] Calabro C. Analysis of dead boxes in dots-n-boxes. Current available online via http: //cseweb.ucsd.edu/ccalabro/.
[4] Ilan Vardi. The mathematical theory of dots. Current available online via http://cf.geocities.com.
[5] Chrisc Berlekamp E R. The dots-and-boxes game [M]. Massachusetts: A K Peters Ltd ,2000.
[6] Gerben M. Blom. An artificial intelligence approach to Dots-and-Boxes. Multi-Agent systems, 2007
[7] Wikipedia, http://en.wikipedia.org/wiki/Dots-and-Boxes
[8] Lex Weaver, Terry Bossomaier, Evolution of Neural Networks to Play the Game of Dots-and-Boxes, In Artificial Life V: Poster Presentations, May 16-18 1996, pages 43-50.

Shuqin Li received a B.S. degree in computer Science from the University of ShanXi in 1985, and the M.S. and Ph.D. degrees in Computer application from the Nanjing University of Science and Technology in 1988 and 2006. She has been with the Department of computer science at Beijing Information& Science Technology University, where she is a Professor. She is also a Technical Committee of China Computer Games Commission and a member of China Association for Artificial Intelligence (CAAI). Her major study is machine study and artificial intelligence.

Dongming Li received a B.S. degree in computer Science from Beijing Information& Science Technology University. Now he is a Master student at Beijing Information& Science Technology University. His research interest is machine study and computer game.



Xiaohua Yuan received the B.S. degree from the South-East Normal University, Chongqing, in 1991, the M.S. degree from the Chinese Academy of Sciences, Beijing, in 1999, and the Ph.D. degree from Nanjing University of Science and Technology, Nanjing, in 2006. She is currently an associate professor in the Department of Information, Shanghai Ocean University. Her research interests include Image Processing and Artificial Intelligence.