Optimizing backtrack solution in Knight Tour's problem based on prediction deadlock probability

Shamsipour Technical and Vocational College

Hamid Jolany

***Abstract*– In this article, we aim at optimizing the backtrack solution to solve knight tour's problem using deadlock probability prediction in chess board. In this way, the response time will be decreased to less than 1 second, despite of the Brute-force backtracking method, there are approximately 4×10^{51} possible move sequences that is well beyond the capacity of modern computers (or networks of computers) to perform operations on such a large set.[2] According to the deadlock probability, the closer result will be selected regardless the other choice, unless it needs to travel again in backtracking. This approach, in addition to elect the right direction, will decrease the count of backtracking and avoids the insufficient tracks.**

Keywords—Knight tour's problem; Prediction probability of deadlock; Backtrack; Algorithm Design; brute-force; optimizing Backtrack

I. INTRODUCTION

A knight's tour is sequence movements of the knight in a chessboard such that the knight visits every cell only once. If the knight ends on a cell that is one knight's move from the beginning cell (so that it could tour the board again immediately, following the same path), the tour is closed, and otherwise it is open. The knight's tour problem is a mathematical problem of finding a knight's tour. Creating a program to find a knight's tour is a common problem given to computer science students [1]. Variations of the knight's tour problem involve chessboards in different sizes rather than the usual 8 × 8, as well as irregular (non-rectangular) boards.

II. ULTIMATE AIM

The main target of this paper is optimizing the Brute-force backtrack solution in knight tour's problem, based on the deadlock probability prediction to decrease the response time and avoid the irrelevant routes.

III. BRUTE-FORCE ALGORITHMS

A brute-force search for a knight's tour is impractical on all but the smallest boards; for example, on a 8×8 board there are approximately 4×10^{51} possible move sequences, and it is well beyond the capacity of modern computers (or networks of computers) to perform operations on such a large set. However, the size of this number gives a misleading impression of the problem difficulty, which can be solved "by using human insight and ingenuity ... without much difficulty."

IV. DIVIDE AND CONQUER ALGORITHMS

By dividing the board into smaller pieces, constructing tours on each piece, and patching the pieces together, one can construct tours on most rectangular boards in polynomial[3][4].

V. NEURAL NETWORK SOLUTIONS

The Knight's Tour problem also lends itself to being solved by a neural network implementation.[5] The network is set up such that every legal knight's movement is represented by a neuron, and each neuron is initialized randomly to be either "active" or "inactive" (output of 1 or 0), with 1 implying that the neuron is a part of the final solution. Each neuron also has a state function (described below) which is initialized to 0.When the network is allowed to run, each neuron can change its state and output based on the states and outputs of its neighbors (those exactly one knight's move away) according to the following transition rules:

$$U_{t+1}(N_{i,j}) = U_t(N_{i,j}) + 2 - \sum_{N \in G(N_{i,j})} V_t(N) \quad (1)$$

$$V_{t+1}(N_{i,j}) = \begin{cases} 1 & if\ U_{t+1}(N_{i,j}) > 3 \\ 0 & if\ U_{t+1}(N_{i,j}) < 0 \\ V_t(N_{i,j}) & otherwise \end{cases} \quad (2)$$

Where represents discrete intervals of time,$\boldsymbol{U(N_{ij})}$ is the state of the neuron connecting cell i to cell j, $\boldsymbol{V(N_{ij})}$ is the output of the neuron from Itoj, and $\boldsymbol{G(N_{ij})}$ is the set of neighbors of the neuron. Although divergent cases are possible, the network should eventually converge, which occurs when no neuron changes its state from time t to t + 1.

When the network converges, either the network encodes a knight's tour or a series of two or more independent circuits within the same board.

VI. WARNSDORF'S RULE

Warnsdorf's rule is a heuristic for finding a knight's tour. The knight is moved so that it always proceeds to the cell from which the knight will have the fewest onward movements. When calculating the number of onward movements for each candidate cell, we do not count moves that revisit any cell already visited. It is, of course, possible to have two or more choices for which the number of onward moves is equal; there are various methods for breaking such ties, including one devised by Pohl [6] and another by Squirrel and Cull [7]. This rule may also be more generally applied to any graph. In graph-theoretic terms, each move is made to the adjacent vertex with the least degree. Although the Hamiltonian path problem is NP-hard in general, on many graphs which occur in practice, this heuristic is able to successfully locate a solution in linear time [6][7]. The knight's tour is a special case[9].

The heuristic was first described in "Des Rösselsprungseinfachste und allgemeinsteLösung" by H. C. von Warnsdorf in 1823.A computer program that finds a knight's tour for any starting position using Warnsdorf's rule was written by Gordon Horsington and published in 1984 in the book Century/Acorn User Book of Computer Puzzles

VII. THE RESEARCH METHOD

in this method, using the deadlock probability prediction at the next movement of the knight in the chessboard, a movement will be chose that it's tending to the deadlock probability is less than the other ones, we know at the first step this deadlock probability is zero for every cells and it will be changed gradually. The knight in the chessboard has between 2 and 8 moves, so each cells according to the pattern illustrated in figure 1 has predetermined value for next move.

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

Figure.1. Chess Board marked by number of allowed movements for the knight

Selecting the cells that have less available movement is best choice because it will tend to the deadlock in the future unless it is filled. There is an inverse relationship between allowed movement number and reach an impasse.

As seen in the image the outer cells is in the highest priority, As regards in a knight's tour problem the knight has to cross a cell only once, these value will be changed gradually in future travels.

Then in the next step a cell will be chose that has these conditions

1- The number of its adjacent empty cells is less than others, or in the other words the probability to be filled is more [10].

n = Count of the cell movements

$\boldsymbol{C_f}$ =Number of the adjacent filled cells of current cell

$$p=(\sum_{k=0}^{n}(\boldsymbol{C_f}))/\boldsymbol{n} \qquad (3)$$

2- After selecting, the adjacent houses doesn't going to deadlock

Whit these principles in a fraction of second the result will be obtained.

VIII. PSEUDO CODE

```
Procedure TakeTour(x,y)
Begin
   If (ChessBoard[x,y]=null) AND
      (Knight.Move(x,y)<>null) AND
      (ChessBoard[x,y]<>Locked)
   Begin
Foreach(Cell in getAvailableMovements(Knight))
      Begin
TakeTour(Cell.x,Cell.y)
      End
   End
End


Function Cells[]getAvailableMovements(K)
Begin
Cells[] Array Of Cells
foreach (position in k.Movements())
   Begin
       If Board[position] <> null)
Cells.add(position)
   End
       Return Cells.Where(lockProbability< 1
       ANDvalue == 0).OrderBy(lockProbability)
End
```

IX. CONCLUSION

In the backtrack, due to much more failed travels increase the response time, moving to the right direction is possible by predicting deadlock and recognizing the inappropriate events and going to the result much more faster, in the current backtrack solution there are approximately 4×10^{51} possible move sequences that it is well beyond the capacity of modern computers (or networks of computers) to perform operations on such a large set, but in this paper we proved to perform these tours in a fraction of second.

X. REFERENCES

[1]. H. M. Deitel, P. J. Deitel. "Java How to Program Fifth Edition." Prentice Hall, Upper Saddle River, New Jersey, pp. 326–328. 2003.

[2]. http://www.josiahland.com/archives/781

[3]. Cull, P.; DeCurtins, J. (June 1978). "Knight's Tour Revisited". Fibonacci Quarterly 16: 276–285.

[4]. Parberry, Ian (1997). "An Efficient Algorithm for the Knight's Tour Problem". Discrete Applied Mathematics 73: 251–260. doi:10.1016/S0166-218X(96)00010-8.

[5]. Y. Takefuji, K. C. Lee. "Neural network computing for knight's tour problems." Neurocomputing, 4(5):249–254, 1992.

[6]. Pohl, Ira (July 1967). "A method for finding Hamilton paths and Knight's tours". Communications of the ACM 10 (7): 446–449. doi:10.1145/363427.363463.

[7]. A. Conrad, T. Hindrichs, H. Morsy, I. Wegener Solution of the knight's Hamiltonian path problem on chessboards Discrete App. Math., 50 (1994), pp. 125–134

[8]. Squirrel, Douglas; Cull, P. (1996). "A Warnsdorff-Rule Algorithm for Knight's Tours on Cell Boards". Retrieved 2011-08-21.

[9]. Alwan, Karla; Waters, K. (1992). Finding Re-entrant Knight's Tours on N-by-M Boards (PDF). ACM Southeast Regional Conference. New York, New York: ACM. pp. 377–382.doi:10.1145/503720.503806. Retrieved 2008-10-28.

[10]. Knight’s Tour Analysis - https://interactivepython.org/runestone/static/pythonds/Graphs/KnightsTourAnalysis.html

[11]. An efficient algorithm for the Knight’s tour problem-Ian Parberry * Department of Computer Sciences, University of North Texas, P. 0. Box 13886, Denton TX 762034886, USA Received 17 October 1994; revised 3 1 October 1995