PROJECT REPORT

**KNIGHT’S TRAVAILS**

Meenasree Ananthavelu  
Department of Computer Science  
University of North Texas  
Denton, Texas, United States  
MeenasreeAnanthavelu@my.unt.edu

Roja Kamble  
 Department of Computer Science  
 University of North Texas  
 Denton, Texas, United States  
 RojaKamble@my.unt.edu

OBJECTIVE

The idea behind Knight’s Travails is to find the path a knight can travel on a chess board. The constraint is that the knight can traverse through a cell only once and it cannot travel outside the board. The knight’s tour can be open or closed. Closed tour is the one in which the knight starts in a particular position and can end the tour in the same position. If the knight does not end in the same position, then it is an open tour. The objective of this project is to use boards of different sizes and to find if it is possible for the knight to cover every cell of the board, considering the constraints. Also, the other objective is to experiment with different positions of the knight and to understand the type of tour that can be achieved for different board sizes along with the time complexity.

APPLICATIONS

Backtracking algorithm is used for Knight’s problem. Backtracking is an algorithmic technique that tries different solutions until it finds a solution that works. This technique has a property to solve the problem. These problems can only be solved by trying every possible configuration and each configuration is tried only once.

Backtracking operates in stages, and it works incrementally. It is an improvement over the Naive solution, and it tests all potential configurations generated. It is used in various puzzles and games like eight queens puzzle, Sudoku, crosswords, verbal arithmetic and so on.

For example, the eight queen’s conundrum, which asks for all possible arrangements of eight chess queens on a regular chessboard so that no queen assaults another, is a classic example of backtracking. Partial candidates are configurations of ‘*i*’ queens in the first ‘*i’* rows of the board, all in distinct rows and columns, in the standard backtracking strategy. Any partial solution containing two queens who are attacking each other can be discarded.

Another example, in Sudoku, each cell is checked for a valid number, and if one is found, the puzzle is solved by moving "back" and then forward again.

PREVIOUS WORK

We drew inspiration from Ian Parberry's work, who looked at the Knight tour problem in the same way we did. A knight tour, according to him, is a sequence of moves in which a knight visits each square on a n x n chessboard exactly once. The challenge of building such a tour,

given n, is known as the knight's tour problem. If the last square visited can be reached by a knight's move from the initial square, the tour is said to be closed; otherwise, it is said to be open.

Euler [10] is credited with starting the formal study of the knight's tour problem in 1759, using the conventional 8 x 8 checkerboard. Rouse Ball and Coxeter [7] provide a useful bibliography of the problem's history up to this point. An explanation of which rectangular chessboards include knight's tours may be found in Dudeney [8, 9].

Conrad et al. [12, 13] present a linear-time sequential technique for creating open knight's tours (with arbitrary endpoints) that may be modified to a parallel algorithm that runs in 0(1) time on 0(n^2) processors. With the same resource limitations, we'll provide a novel algorithm for closed knight's tours. The sequential version is simple to define and implement, whereas the parallel version is simple to describe and analyze.

In part 2 of the paper, they employed the "divide and conquer" approach to create a regular knight's tour, quadrisected knight's tours, and tours that are invariant under rotations. For numerous prominent parallel machine architectures, Section 3 shows versions of the new divide-and-conquer method. N stands for the set of natural numbers in this work (including zero). A "knight's tour" will always refer to a closed knight's tour unless otherwise specified. Their experiment was built with a time complexity of O(n^2). For constructing knight's tours, there are numerous independently devised linear time (i.e., O(n^2)) algorithms.

ANALYSIS

The below backtracking algorithm was analyzed and used to experiment.

If squares visited == all

{

print path

}

Else

{

1. Recursively add any one of the moves out of the 8 moves to the resultant vector. Also check if this move is resulting in finding the solution.
2. If the above move does not result in a solution, then take this move out of the resultant vector and experiment with other moves.
3. Return false if the other moves do not work.

}

The experiments were completed for two datasets with board sizes 5x5 and 6x6 and the result was as per expectation. The start position of the knight and the board size were taken as a dataset.

EXPERIMENT

1 DATASETS

Boards of the various sizes are taken as datasets for the knight’s tour problem. The start position of the knight is also taken as a dataset.

1.1 Board size:

Minimum board size starts with 5x5 for a knight to travel a complete tour and anything below the size 5x5, the knight will have to loop the same cell again and no complete tour is possible. We implemented the tour for board sizes 5x5, 6x6, 7x7 and 8x8 from the corner position and, from different middle positions. The time of execution from the corner position to different middle positions has been calculated.

The inference is that the execution time differs with respect to the board size. The execution time is directly proportional to the size of the board, meaning the execution times increases as the size of the board increases.

1.2 Position of the knight

From the corner position, the knight can cover all the cells of the board and can have a complete tour. From the middle position, the tour can be a closed tour or not. If there is no next move, the knight would stop, and the tour would end.

Below figures are for Datasets:

1. The size of the board

2. Start position of the knight.

Here we considered (0,0) as start position of x and y co-ordinates.

The middle position varies depending on board sizes as shown in below figures.

A picture containing shoji, crossword puzzle

Description automatically generatedA picture containing text, shoji, crossword puzzle

Description automatically generated

A picture containing text, shoji

Description automatically generatedA graph with numbers and letters

Description automatically generated with low confidence

2 PURPOSES OF THE EXPERIMENT

When a Knight starts in one of the middle positions in an 8x8 board, it will have 8 possible next moves. It is checked if the Knight can traverse the whole 64 cells starting in cells with coordinates (3,3), (3,4), (4,3) and (4,4). Implemented a knight tour from middle positions and the solution is displayed only if the path covers all 64 cells in an 8x8 chess board. The time complexity is also compared between different board sizes when a tour starts from corner cells and from middle positions.

It is evident that if a knight starts at one of the corner cells on an 8x8 chess board, it can traverse the whole 64 cells. It is also checked if the same is possible for other sized boards from 5x5 to 7x7.

2.1 Execution Time

Comparison of execution time with two different devices is done for the knight travelling from corner position and for the knight travelling from middle positions.

For 8x8, there is no complete tour we found for middle positions we initially chose. Hence, we picked little away from the middle position to check if complete tour is possible for few other positions.

The below measures are execution times for our algorithm on an 8x8 board.

Device 1: (MacBook Pro Laptop)

Corner position [0,0]:

Time for execution: 521322 microsec

Middle position [3,2]:

Time for execution: 164461 microsec

Middle position [4,5]:

Time for execution: 739928 microsec

Device 2: (Dell windows Laptop)

Corner position [0,0]:

Time for execution: 595231 microsec

Middle position [3,2]:

Time for execution: 245837 microsec

Middle position [4,5]:

Time for execution: 1110990 microsec

The running time of this algorithm has a maximum run time of O (N x N), where N is size of the chessboard.

3 EXPERIMENTAL RESULTS

Figures 1-4: Corner position knight tour for all boards, that is from 5x5 to 8x8.

Size: 8x8 Figure 1

**A picture containing text

Description automatically generated**

Size: 7x7 Figure 2

**Text

Description automatically generated with medium confidence**

Size: 6x6 Figure 3

Text

Description automatically generated

Size: 5x5 Figure 4

**Text

Description automatically generated**

Figure 5 and 6: Middle position knight tour for 8x8 board.

Middle position (3,2) Figure 5

**Text

Description automatically generated**

Middle position (4,5) Figure 6

**Text

Description automatically generated with low confidence**

4 TIME COMPLEXITY

The time complexity is defined as the length of the input that determines how long an algorithm takes to run. It calculates how long it takes an algorithm to execute each piece of code.

The time complexity for knight travails using backtracking algorithm is O(N^2), which is exponential.

When we run the code in online compiler, we observed that execution time is inconsistent as it takes the available server to run during runtime. Here we had an option to compile 100 times and take average execution time to analyze the time complexity. Though we take average this could be inconsistent again. Good implementation would be the considering consistent time of execution.

We came up running the code in a software that is VS code using C++, when we run our program here, we observed execution time was consistent with slight variation depending on the devices. We took average value after several executions, and we were able to plot a graph comparing the time complexity for below two scenarios.

Scenario 1: Different board sizes from 5x5 to 8x8.

Scenario 2: Middle positions of the knight of different boards.

Note: Only if complete tour is possible for the positions defined in the table, then those start positions will have solution, otherwise not.

Scenario 1: Different board sizes from 5x5 to 8x8.

* knight’s path from corner start position

With respect to start position (0,0) of x and y co-ordinates and having board sizes from 5x5 to 8x8, where we considered the average execution time of each board by running several times. Below table depicts what is explained.

Start Position (0,0):

|  |  |
| --- | --- |
| Board size | Time of execution |
| 5x5 | 825 |
| 6x6 | 19120 |
| 7x7 | 428380 |
| 8x8 | 510894 |

Based on the measures, we plotted graph considering board size on x axis and time of execution on y axis. The curve depicts the time complexity is exponential.

Chart, line chart

Description automatically generated

Scenario1 Result:

A picture containing table

Description automatically generated

Scenario 2: Middle positions of the knight of different boards.

* By backtracking and to attain closed tour of the knight’s path from middle start position.

With respect to start middle positions of x and y co-ordinates and having board sizes from 5x5 to 8x8, where we considered the average execution time of each board by running several times. Below table depicts what is explained.

Start Position (Middle)

|  |  |
| --- | --- |
| Board size | Time of execution |
| 5x5 (2,2) | 822 |
| 6x6 (2,2), (2,3), (3,2), (3,3) | 157346, 35020, 6570, 44978 |
| 7x7 (3,3) | 189019 |
| 8x8 (3,3), (3,4), (4,3), (4,4) | No Solution |
| 8x8 (3,2), (4,5) | 160320, 709642 |

Based on the measures, we plotted graph considering board size on x axis and time of execution on y axis. The curve depicts the time complexity is exponential.

Chart, line chart

Description automatically generated

Scenario 2 Result:

Calendar

Description automatically generated

Time complexity is O (N x N) which is nothing but O (N^2) as you can see in the middle position it does follow the O (N^2) curve closely, whereas, for the start position of (0,0) since we do not have enough data set to visualize the full curve, it loosely follows O (N^2) curve.

CONCLUSION

Implemented the knight’s problem and the previously established goals after study. Even though there were solutions available, the challenge was to study the problem, understand the algorithm and to modify the same according to the requirements of this project. Also, we did notice that our time complexity is O (N^2), but due to lack of datasets we partially followed the O (N^2) graph.

Future Outcomes:

[1] The implementation on higher size of boards; for example, 16x16 and 32x32.

[2] The implementation of knight with incomplete tour, meaning having a tour till the point where it is possible without visiting the same cell. Partial tour, where no further moves are available.

REFERENCES

[1] <https://cs.stackexchange.com/questions/13181/time-complexity-of-a-backtrack-algorithm>

[2] <https://www.chess.com/terms/knights-tour-chess>

[3] <https://www.mrexcel.com/board/threads/tough-problem-4-vba.365052/page-3>

[4] <https://blogs.sap.com/2018/10/29/solving-the-knights-tour-problem-with-hana-graph/>

[5] <https://youtu.be/pwlxQeHchFQ>

[6] <https://youtu.be/ab_dY3dZFHM>

[7] W.W. Ball and H.S.M. Coxeter, Mathematical Recreations and Essays (University of Toronto Press, Toronto, 12th ed., 1974).

[8] H.E. Dudeney, Amusements in Mathematics (Thomas Nelson & Sons, 1917).

[9] H.E. Dudeney, Amusements in Mathematics (Dover, New York, 1970).

[10] L. Euler, Solution d’une question curieuse qui ne paroit soumise a aucune analyse, Mem. Acad. Sci. Berlin, (1959) 310-337.

[11] Link for referred paper: <https://core.ac.uk/download/pdf/81964499.pdf>

[12] A. Conrad, T. Hindrichs, H. Morsy and 1. Wegener, Wie es dem springer gelang, schachbretter beliebiger groesse und zwischen beliebig vorgegebenen anfangs- und endfeldem vollstaendig abzuschreiten, Spektrum der Wissenschaft, (1992) p. 10-14.

[13] A. Conrad, T. Hindrichs, H. Morsy and I. Wegener, Solution of the knight’s Hamiltonian path problem on chessboards, Discrete App. Math. 50 (1994) 125-134.

**GitHub**

<https://github.com/Roja-Kamble/algorithm-project>