Analysis of Different Approaches in Solving Knight's Tour Problem

CS7IS2 Project (2019-2020)

Aashima Goel, George John Chavady, Githu Jeeva Savy, Piyush Mankad, Soumen Ghosh

goelaa@tcd.ie, chavadyg@tcd.ie, jeevasag@tcd.ie, mankadp@tcd.ie, ghoshso@tcd.ie

Abstract. The Knight's tour problem on the chess board is a specially intriguing tour problem which fascinates computer scientists. In this paper we propose some novel algorithms based on the neural networks and ant colony optimization for solving the problem and compare their performance against the existing algorithms like depth first search, recursive backtracking and Warnsdorff's algorithm. The problem has been implemented using these specified algorithms and the appropriateness and performance of these algorithms has been evaluated.

1 Introduction

In the recent decades, chess board games have become a centre of attraction for many of the efforts involving Artificial Intelligence. The number of chess game engines with the capability of playing chess at a master's level are over 250. This can be achieved by using certain sophisticated techniques such as minimax algorithm for best move identification and alpha beta pruning methods for reducing the tree searches. [1] Even though these advanced algorithmic developments have boosted the positioning and sorting techniques in chess games, they are no longer near human intelligence.

The Knight's tour problem is an intriguing old puzzle. It is a peculiar case of NP hard problem used for finding a Hamiltonian path on a special graph. It has fascinated chess players, computer scientists and mathematicians for many years. The traversal movements are easy for a king, queen, rook etc. The movement followed by Knight is 'Ll shaped one.. The main goal of this puzzle is to find an optimum path from the beginning to the end point by traversing all the points on the chess board only once. The sequence of moves is done by following the rules in chess (L shape moves). This problem can also be translated to a graphical problem in which the existence of a Hamiltonian circuit is checked.

There are two types of tours offered by Knight are the open and the closed. If a single knight's move can reach the last square from the first one, it can be called a closed tour. The problem in which every square is visited once and don't have the ability to

return to the origin in one move can be called as an open Knight's tour. The problem of the closed tour is an instance of the Hamiltonian cycle problem.

We can use either algorithmic solutions or heuristics-based solutions for the tour. Some of the algorithms which we can use are categorized into Brute force algorithms, divide and conquer methods, Warndorff's method. We are also using and solutions based on machine learning and neural networks as well as an ant colony optimization algorithm.

One of the applications of the solution of the Knight's tour problem is in preserving digital image information security. This can also be used in image encryption schemes for visual cryptography. This encryption is done mainly by division of the image into 8x8 pixels and then shuffling the pixel values [2].

This paper has been organized in the following way. In section 2, some of the previous works and algorithmic approaches are discussed. The approaches used by us are discussed in section 3. The comparisons of performance evaluations for different algorithms are discussed in Section 4. A conclusion section is added at last.

2 Related Work

There are many existing possible solutions for solving the Knight's tour problem. There are different techniques and approaches used for solving the Knight's tour. One of the earliest systematic solutions for this problem was contributed by Euler in 1759 [2]. The method proposed by Euler was aimed at the movement of Knight around a chess board till only a few cells are left. To incorporate those cells Euler introduced a set of rules and these are extremely tedious. So it is not easily applicable as a computer algorithm.

German Mathematician H C Warnsdorff [6] came up with a solution which uses a simple greedy heuristic approach. He suggested to always make a move to a square with minimal degree and also an unvisited one. So we select a square with least possible moves using heuristics rather than random one.The methodology proposed by Warnsdorff doesn't produce the desired results every time. Also, the rule doesn't provide a tie breaking rule.

Genetic Algorithm [7] provides a modern way of implementing the knight's tour problem. There are also other algorithms like intelligent ant colony optimizations algorithms based on this problem. The most modern technique is the usage of a neural network based solution for the problem. Each possible Knight movement can be represented as a neuron.

In this paper, we are comparing the performance of various algorithms like depth first search, recursive backtracking, and Warnsdorff's algorithm against the performance of novel neural network based solutions and ant colony optimization algorithms. The performance evaluation metrics for this problem generated using different algorithms

are compared to find the pros and cons of each algorithm and also to find the optimum one.

3 Problem Definition and Algorithms

The Knight in a chess board can move only in one of the eight possible directions which is shown in figure 1 [1]. Every square on the chess board will be visited by the Knight only once until it has finished with the 64 squares. After traversing all the squares, we are returning to the start position. Since we arrive at the position where the Knight's movement started, such a tour can be called a closed Knight's tour problem.
An open Knight's tour uses a Hamiltonian path while a closed tour uses a Hamiltonian circuit. The main difference between the closed and open tour is that we are not required to get back to the starting square in an open one.

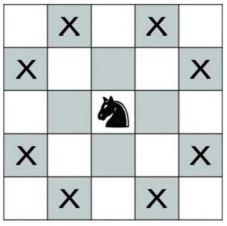

Figure 1: The Eight Possible Knight Moves

Our focus was to concentrate on the knight's tour problem on an m x m sized board. Then start from an initial position and then perform the Knight tour traversal on until all the squares are visited.
There are certain conditions to be met for the existence of a Knight's tour [3]. For the existence of Knight's tour on an m x n chessboard with m less than or equal to n, following conditions need to be met:

1. both numbers m and n should be odd
2. m equals to 1, 2, or 4
3. m equals to 3 and n equals to 4, 6, or 8.

Out of the several approaches used for solving the problem we are comparing the performance of certain algorithms in performing the tour on our board.

3.1 Brute Force Algorithms

Here we are considering the performance of two off the shelves approaches in solving the problem. We are using the depth first search for the Knight's tour. In the depth first search [4] the traversal starts from a root node and then proceeds along the same branch as far as possible before back-tracking and switching the branches. Depth first search solution can be used for solving a large number of artificial intelligence and combinatorial theory problems. The pseudocode for our dfs approach is as follows.

3.2 Warnsdorff's Algorithm

As previously suggested, Warnsdorff gave a simple heuristic to solve the Knight's tour problem on an 8x8 chessboard. He stated that, in order to select the next move for the Knight, the available moves should satisfy three conditions declared below [6];

1. the position should be adjacent to the current position.
2. the position be unvisited by the Knight
3. the position should have the fewer number of unvisited squares adjacent to it.

This led to a number of problems when the proposed heuristic ended up in a tie. Warnsdorff suggested that any random move may be chosen to break the tie however experiments [6] show that this rule (breaking ties at random) decreases in effectiveness as board size increases.
This problem could be solved by specifying some additional rule to the tie breaking problem rather than solving it randomly. An example of such a solution was given by Paul [8] in his paper, where he solves the problem by fixing the move order (by giving every move an index from 1 to 8) and selecting the first move from the move order shall a tie breaking situation arise. His method produced far greater results than breaking ties at random.

3.3 Backtracking Algorithm

Backtracking is a recursive algorithmic approach [5]. This algorithm proceeds to find the solution of a problem by using an incremental approach. Whenever there is a need for a different approach to a problem at one stage, this algorithm tries all the possible combinations of options recursively. By the nature of the Knight's tour problem, a backtracking algorithm can provide an efficient traversal. But the naive backtracking algorithm may be very slow even in more advanced powerful computers due to the number of traversals required. This is where the advantage of Warnsdorff's algorithm comes into picture as it uses simple heuristics and thereby reduces the complexity involved. A Knight's tour on a chessboard has different options at different stages during the tour. Sometimes, some of our moves lead to a dead end and then we have to start again by recursive backtracking and choose some alternate options.

3.4 Neural Networks

According to Takefuji, the Knight's tour problem could also be solved by using McCulloch-Pitts Neurons, a predecessor of the modern Neural Networks. He proposed a triangular network with a two dimensional representation consisting of $p(p-1)/2$ processing elements(neurons) where $p = m * n$ chessboard [9]. The two possible states of neurons are active and inactive generating output of 1 or 0. An active neuron can be a part of the solution.After the start of the network, every active neuron is configured to reach a stable state which means for a neuron to have exactly two neighbouring neurons that are in an active state. if any of the neurons are not in a stable state then the state of those neurons keeps on changing until a solution is

obtained. The following transition equation is used to solve the problem for the *ij* neuron:

$$\text{if } d_{ij} = 1 \text{ then}$$

$$\frac{dU_{ij}}{dt} = -\left(\sum_{k=1}^{p} V_{ik} d_{ik} - 2\right) - \left(\sum_{k=1}^{p} V_{kj} d_{kj} - 2\right)$$

$$\text{if } d_{ij} = 0 \text{ then}$$

$$\frac{dU_{ij}}{dt} = 0 \,,$$

where d_{ij} is the state of the neuron

$\frac{dUij}{dt}$ is the next state for the neuron

$V_{ik}\ d_{ik}$ is the output of the current neuron

The output of the neuron V_{ij} gets is updated by using the hysteresis McCulloch-Pitts function:

$V_{ij} = 1$, if $U_{ij} > 3$

$V_{ij} = 0$, if $U_{ij} < 0$

$V_{ij} = V_{ij}$, if $0 \leq U_{ij} \leq 3$ (no changes otherwise)

We initialize all initial states of neurons to 0 after the first iteration. The output of each neuron will be initialized randomly to zero or one. The squares on the chessboard are counted in row-major order for updating the neuron states. The neurons that represent a knight's move out of each square are also enumerated.

3.5 Ant Colony Optimization

The basis of ant colony optimization is based on a natural phenomenon associated with ants. They are smaller creatures with smaller brains and are almost blind. Despite these inabilities they are capable of finding a food source and the return back to their nest by following the shortest path possible. A colony of ants uses the same approach to find the shortest path to the nest when confronted with an obstacle along their path. The movement of ants deposits a chemical called pheromone whose concentration can be more along the path that the ants follow the shortest route to its nest.

This probabilistic nature of ant colony optimization has been converted into various algorithms using a problem specific heuristic [10] . The knight's tour problem can be considered as an ant colony traversal along a graph in which the squares on the board are represented using the graph nodes and the edges correspond to the legal set of moves by a Knight. The start position of an ant is a square. It moves to the next previously unvisited square by following the legal move of the Knight.

4 Experimental Results

This section compares the efficiency and effectiveness of different algorithmic approaches in solving the Knight's tour puzzle. We are comparing the performance of different algorithms on both the boards with odd dimension as well as that with even dimension. According to the rules of existence of a Knight's tour, there is no possible

Knight's tour for 3 x 3 and 4 x 4 boards. For the odd dimensional boards we are taking into account the 5 x 5 board and for the even one we are considering the normal 8 x 8 chess board. The complexity of algorithms and time taken to perform the tour is directly proportional to the board size. The moves specified from the start location on the board to reach the next position are kept the same for all the algorithms. The moves from a given start position are (2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1). The choice of one of this moves for finding the next position varies between different algorithms. Simple brute force algorithms may use a random selection whereas complex algorithms select a more intelligent move or the move may also depend on the heuristic specified. All of our algorithms start the tour at a fixed position 0,0. From there the knight's tour is performed until it traverses all the squares on the board.

4.1 Appropriateness

The Appropriateness of the algorithm to be selected will depend on the results that we desire. There are $4x10^{51}$ possible steps for a knight to take on an 8x8 chessboard, and because of it being an NP-hard problem all of the conclusive solutions that we have implemented here would have a Big O notation in exponential time except one which is the one using Warnsdorff Heuristic but the downside would be that it is a non deterministic solution meaning that it might not give you a solution when the chess board size grows too big.
The Neural networks implementation will always give you a solution in the closed tour bracket of all the 26 trillion undirected solutions possible on an 8x8 chessboard, which is a number that is 750 times smaller than all the possible open and closed knight tours. When you apply the dedicated tie breaking heuristic to the Warnsdorff Rule, instead of randomly assigning the next move a successful solution is almost always guaranteed for a chessboard size till 50x50, which would also be the single most fastest and probably the most reliable solution for the problem.

4.2 Comparison

The performance of our novel approaches are better when compared with the old off-the-shelf algorithms. The results are better in terms of time and space complexity as well as with the traversals for finding the knight's tour.
Traditional approaches to Knight's tour problem along with their pros and cons.

Recursive Backtracking

Pros	**Cons**
Simple to understand and implement.	Multiple function calls are expensive
Stack is used for state changes so that it is easily accessible	Increase in branching causes inefficeincy
Intuitive approach of trial and error.	Requires recursion which can be something worse, because CPU stack space is limited and

	can be consumed quickly by recursion.

Depth First Search

Pros	**Cons**
Nodes have a linear memory requirement	No guarantee that it will give you a solution.
Reduced time and space complexity compared with breadth first approach	Smaller cut off depth increases time complexity
Solution can be found without much more searching.	Determination of depth until the search reaches maximum depth

The pseudo code for our two novel approaches are mentioned below along with their pros and cons:

Neural Network based solution

Pros	**Cons**
Would always provide you with a closed tour solution	Providing only closed tour solution means missing out on lot of potential solutions
This approach will definitely give you a solution without any restart or running out of memory	Cannot evaluate a solution on odd no of squares on a chessboard
The algorithm has a convergence rate of 19.4% on 8x8 chessboard	Random initialization can mean a significant time gain in finding the solution

Pseudocode

1. Initialise the chessboard size p=M*N and create p(p-1)/2 neurons with their states and outputs.
2. Connect all the neurons with their neighbours based on an 'L' shaped pattern
3. Randomly initialise the neuron outputs with 0 or
4. For a chosen number of iterations:
   Update the neuron outputs and neuron states of all the neurons based on the conditions given in the algorithms
5. If a state arises where all the degree of vertices of a neuron are 2
   Check: whether the solution found completely interconnected, that is it is not 2 or more subtours of a knight tour on the same chessboard.
   Otherwise: Repeat from Step 3

Ant Colony Optimization

Pros	**Cons**
Higher chance of obtaining the shortest path by choosing edges with more pheromones.	Difficult for theoretical analysis

The algorithm produces 0.076 tours per attempt when compared to the naive depth first search which yields 0.00003 tours per attempt.	Uncertain time to converge
The information sharing between the ants. It means that the knowledge from ants starting on remote positions can be used by other ants.	The method is experimental rather than theoretical

Pseudocode

1. Start and Initialise the chessboard: each T r,c,k = 10 ^ -6 ; where k = edges corresponding to legal moves
2. or each cycle
   Evaporate pheromones:
   T r,c,k = (1 – p) * T r,c,k ; 0 < p < 1 and p equals evaporation rate
   For every starting square position
   Initiate an ant move: list tch[r, c]
   While the move is not finished
   Choose the available next move
   Move to a new position or square on the board
   once the tour is complete: length of list = (n * n) - 1, save it
   Lay pheromone: T a,r,c,k = Q * (|moves| - i)/(n*n – i); where i = ant a's ith iteration move,
   if edge is equal to k
   Update pheromones: T r,c,k = T r,c,k + T a,r,c,k
3. Stop

4.3 Analysis and improvements

This section talks about the various implemented solutions and the possible ways that they could have been improved:
From an analysis perspective, we look at the various execution times of the algorithm. For the performance of the algorithms to remain unaffected by the underlying environment, we decided to run them on a common online platform (Google Colab) for a total of 20 times with a standard timeout of 10 minutes which if exceeded would result in a failed attempt to provide a solution. The graph for the execution time is shown in Fig. 2.

It has been observed that several algorithms when starting from the same initial position have a higher tendency to converge to the same solution which was basically due to them being governed by the same set of rules which behaves similarly in familiar situations. Special care has been taken while experimenting to avoid the outcome of a single solution for more than one time.

Fig. 2: Comparison of Execution Time of Algorithms

Iterative Warnsdorff: The main problem with Warnsdorff solution is that it's nondeterministic and unreliable in breaking ties, the heuristic however works perfectly fine. so, in a way we could implement the warnsdorff rule in the iterative manner in order to break the ties but the drawback of such a solution would be that we won't know where the tie breaking stops which could take the execution time of the algorithm from polynomial time to exponential time defeating the whole purpose of a faster solution Bring it down to the level of all other algorithms and hence we use fixed move orderings to break the ties.

Multiple Ant Colony Optimization: Using only a single ant colony more often than not leads to convergence towards a single path that ultimately manages all ants to go on that path which leads to a dead end. For the algorithm to find multiple paths on the chessboard one can use multiple colonies of ants that repel each other's pheromones. By using this method, one is guaranteed to find multiple paths that lead to a greater pool of knowledge which might be shared by the colonies at a later stage. This kind of an approach can also help us in finding unique solutions.

Backtracking with Warnsdorff: As we have previously seen that randomly traversing any path on the chessboard doesn't give any fruitful results. On the same note, naive backtracking is generally slow and can lead to dead ends which would require us to re-evaluate our decisions. One way to optimise backtracking would be to use Warnsdorff's heuristic in the decision-making process this might lead to better decisions made over time with fewer revaluations.

Divide-and-conquer using Neural Networks: It is said that large chess boards take a lot of time in computing their solution. One possible approach for this problem would be to divide the chess board into smaller proportions on which the solutions are computed faster comparatively and then combine those solutions. an algorithm that one could take advantage of would be neural networks because one could use the information of the degrees of the all edges inside it and tweak the algorithm for the divide-and-conquer approach to work.

5 Conclusion

In conclusion, P=NP is a hard problem to solve. Our experimental analysis compared the performance of the traditional algorithms with that of the proposed novel approaches. Results showed that for a fixed 8 x 8 size board, neural networks solution has a performance similar to that of traditional algorithms. As the board size increases, neural network based solutions are more optimal and faster. The ant colony approach produced the results faster and is easier to control the path style irrespective of whether the tour is open or closed. Some future works include the estimation of all tours using neural networks and ant colony algorithms, validation of knight's tour on a board with holes, best first search approach for finding the path and knight's tour on rectangular boards.

References

1. Escalante, R.G. and Malki, H.A., 2006, July. Comparison of artificial neural network architectures and training algorithms for solving the knight's tours. In The 2006 IEEE International Joint Conference on Neural Network Proceedings (pp. 1447-1450). IEEE.
2. Singh, M., Kakkar, A. and Singh, M., 2015. Image Encryption Scheme Based on Knight's Tour Problem. Procedia Computer Science, 70, pp.245-250.
3. Schwenk, A.J., 1991. Which rectangular chessboards have a knight's tour?. Mathematics Magazine, 64(5), pp.325-332.
4. Cormen, T.H., 1990. Leiserson, Charles E.-Rivest, Ronald L. Introduction to algorithms.
5. Ghosh, D. and Bhaduri, U., 2017, September. A simple recursive backtracking algorithm for knight's tours puzzle on standard 8× 8 chessboard. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1195-1200). IEEE.
6. Squirrel, D. and Çull, P., 1996. A Warnsdorff-Rule Algorithm for Knight's Tours.
7. Al-Gharaibeh, J., Qawagneh, Z. and Al-Zahawi, H., 2007. Genetic Algorithms with Heuristic-Knight's Tour Problem. In *GEM* (pp. 177-181).
8. Pohl, Ira. (1967). A method for finding Hamilton paths and Knight's tours. Commun. ACM. 10. 446-449. 10.1145/363427.363463.
9. Takefuji, Yoshiyasu & Lee, Kuo. (1992). Neural network computing for knight's tour problems. Neurocomputing. 4. 249-254. 10.1016/0925-2312(92)90030-S.
10. Hingston, P. and Kendall, G., 2005, September. Enumerating knight's tours using an ant colony algorithm. In *2005 IEEE Congress on Evolutionary Computation* (Vol. 2, pp. 1003-1010). IEEE.