J. Appl. Math. & Informatics Vol. **41***(2023), No. 6, pp. 1317 - 1326*
https://doi.org/10.14317/jami.2023.1317

PERFORMANCE OF A KNIGHT TOUR PARALLEL ALGORITHM ON MULTI-CORE SYSTEM USING OPENMP

VIJAYAKUMAR SANGAMESVARAPPA*, VIDYAATHULASIRAMAN

Abstract. Today's computers, desktops and laptops were build with multi-core architecture. Developing and running serial programs in this multi-core architecture fritters away the resources and time. Parallel programming is the only solution for proper utilization of resources available in the modern computers. The major challenge in the multi-core environment is the designing of parallel algorithm and performance analysis. This paper describes the design and performance analysis of parallel algorithm by taking the Knight Tour problem as an example using OpenMP interface. Comparison has been made with performance of serial and parallel algorithm. The comparison shows that the proposed parallel algorithm achieves good performance compared to serial algorithm.

AMS Mathematics Subject Classification : 65D30, 65D32.
Key words : Knight-tour problem, parallel algorithms, multi-core architecture, openMP, performance analysis.

1. Introduction

In a research perspective, the use of multi-core architecture has significantly impacted the field of computer science and computing in general [1]. The ability to have multiple processors within a single chip has allowed for the development of highly parallel computing systems that can perform complex tasks much faster than traditional sequential computing systems. [2,3,4]

Multi-threading is a technique that enables parallel programming by allowing different parts of a program to be executed simultaneously on different processors within a multi-core system. This approach to programming has opened up new opportunities for high-performance computing and has made it possible to solve computationally intensive problems that were previously impossible to tackle using traditional computing techniques [5].

Received February 19, 2023. Revised April 22, 2023. Accepted September 12, 2023.
*Corresponding author.

OpenMP, as an API for implementing multithreading, has played a crucial role in enabling parallel programming on multi-core systems. By providing a framework for developing parallel code, OpenMP has made it possible for researchers and developers to harness the power of multi-core systems and achieve significant performance improvements in their applications.

Flynn's Taxonomy is a classification scheme used to categorize computer architectures based on their ability to perform instruction-level and data-level parallelism. In instruction-level parallelism, multiple instructions are executed simultaneously, while in data-level parallelism, multiple data items are processed simultaneously. The use of multi-core architecture falls under Flynn's Taxonomy's instruction-level parallelism category, as it involves executing multiple instructions simultaneously on different processors within a single chip.

In Summary the use of multi-core architecture and the development of parallel programming techniques such as multi-threading and APIs like OpenMP have revolutionized the field of computing, enabling researchers and developers to tackle complex computational problems that were previously impossible to solve using traditional sequential computing systems. The classification scheme of Flynn's Taxonomy provides a useful framework for understanding the different types of parallelism that can be achieved in computer architecture. The goal of this study is to investigate the performance of a parallel algorithm for solving a specific computational problem on a multi-core system using OpenMP. The problem involves finding the shortest path between two points in a large graph, which is computationally intensive and difficult to solve using traditional sequential algorithms. The objective of this study is to evaluate the effectiveness of the parallel algorithm in terms of speedup and scalability, and to compare the performance of the parallel algorithm with that of a sequential algorithm

Problem Statement

The problem statement for the Knight Tour problem is to find a sequence of moves for a knight to visit every square on an 8x8 chessboard exactly once. The knight must start from a specified square and move according to the rules of chess, which allows the knight to move two squares in one direction and one square in the perpendicular direction in a single move.

The knight cannot move off the board, and it must visit every square on the board exactly once. The problem is computationally complex and has been widely studied as a benchmark problem for testing the performance of algorithms and parallel computing techniques.

2. Related work

The Knight Tour problem has been a subject of interest for researchers in the field of computer science and optimization for many years. Several approaches have been proposed to solve this problem, including backtracking algorithms,

graph-based algorithms, and heuristic methods [1], [2]. In recent years, with the increasing popularity of multi-core architectures, researchers have focused on developing parallel algorithms for solving the Knight Tour problem. For example, researchers have proposed using parallel depth-first search algorithms to improve the performance of the solution on multi-core systems [3], [4]. Additionally, researchers have explored the use of parallel random-walk algorithms to solve the Knight Tour problem [5]. Regarding the use of OpenMP for parallel programming, several studies have explored the effectiveness of this approach for various applications. For example, researchers have used OpenMP to parallelize scientific simulations and have achieved significant performance improvements [6], [7]. Javier Cienfuegos et al. proposes and evaluates several heuristics for the Knight's Tour problem implemented in parallel on multi-core systems using OpenMP. It also includes a comparison to a sequential algorithm and analyzes the performance benefits of the parallel approach. Ali Jannesari et al provides an overview of various parallel computing models and tools, including OpenMP, and their suitability for multi-core and many-core processors. It also discusses the challenges and opportunities for developing efficient parallel programs. Alok Singh and Amanpreet Kaur presents a parallel algorithm for solving the Knight's Tour problem using OpenMP and analyzes its performance on a multi-core system. It also compares the results to a sequential algorithm and discusses the scalability of the parallel approach

3. OVERVIEW OF PROPOSED WORK

3.1. Knight Tour Problem.

In the chess game, the knight can jump in a special manner. Knight can move either two squares horizontally and one square vertically or two squares vertically and one square horizontally in each direction, So the complete movement looks like English letter 'L' as shown in the figure 1

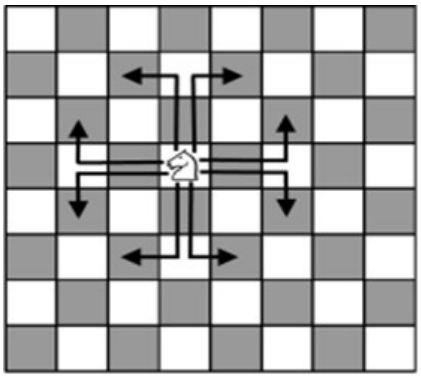


FIGURE 1. 8X8 Chessboard shows the possible moves

In this problem, there is an empty chessboard, and the knight starts from any location in the board and it has to reach the given goal state [1], [7]. For our convenience the 8X8 chessboard is numbered as shown in the figure 2

Given a start state S and a goal state G, the knight has to start from S and

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

FIGURE 2. 8X8 Chess board each square numbered from 0 to 63.

reaches G with legal moves [1]. Many algorithms were developed using graph approach with backtracking. In this paper a sequential and parallel algorithm has been developed and the comparison is made to show the minimization of time in the parallel approach. Graph approach with backtracking" refers to a common algorithmic approach for solving problems like the Knight Tour problem. In this approach, a graph is constructed to represent the possible moves of the knight on the chessboard. Backtracking is used to explore the graph and find a solution by trying different paths until a valid solution is found. The sequential algorithm for the Knight Tour problem is based on a backtracking approach. It starts from an initial position on the chessboard and moves the knight to an adjacent square that has not been visited before. The algorithm continues moving the knight to unvisited squares until all squares have been visited or it reaches a dead end. If the knight reaches a dead end, the algorithm backtracks to the previous square and tries a different path. The algorithm continues backtracking until it finds a path that covers all squares on the board or until all possible paths have been exhausted. The parallel algorithm for the Knight Tour problem uses a divide-and-conquer approach, where the board is divided into smaller sub-boards, and each sub-board is assigned to a different processor. Each processor runs a copy of the sequential algorithm on its assigned sub-board. The solutions found by each processor are combined to form the final solution. The OpenMP interface

is used to implement the parallel algorithm. OpenMP provides a framework for developing parallel code using shared-memory multiprocessing. The parallelism is achieved by dividing the work among multiple threads that run concurrently on different processors

3.2. Eight Possible Moves of a Knight.

From the figure 3, eight possible moves are shown

Move 1: row=row-1 column=column+2
Move 2: row=row-2 column =column+1
Move 3: row=row-1 column =column-2
Move 4: row=row-2 column =column-1
Move 5: row=row+1 column =column+2
Move 6: row=row+2 column =column+1
Move 7: row=row+1 column =column-2
Move 8: row=row+2column =column-1

In the above 8 possible moves from the start state S some of the moves may be possible and the some of the moves may be not possible. For example, from the square 1only 4 moves are possible. ie move 1, move 5, move 6 and move 8 are impossible moves from square1 as shown in the figure 3

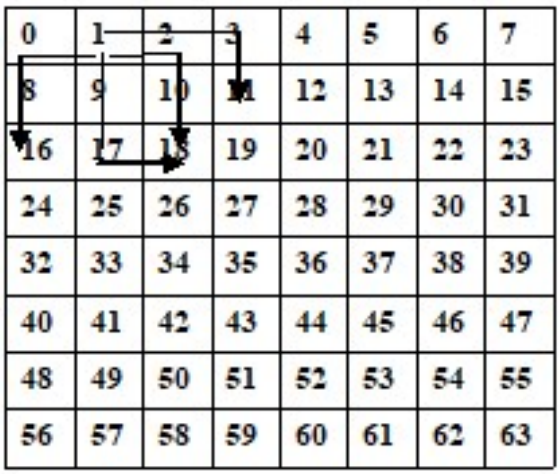


FIGURE 3. All possible moves of square1.

4. PROPOSED PARALLEL APPROACH ALGORITHM

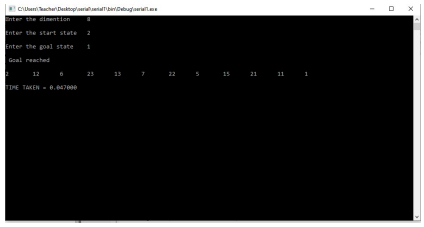
The parallel portion of this algorithm is implemented in the eight function calls from any state with eight possible legal moves.

Algorithm Parallel_Knighttour()
Step1. Initialize nxn dimension array as shown in Figure 2, n, s, g and visited[]
//s start state & g goal state. Visited[] is an array to store the visited state
Step 2.Store s in visited[]
Step 3.Find the (r,c) value for s
Step 4.Call move(int r, int c)
Step 5.move(r, c)calls the 8 possible moves from s which gives p1, p2, p3, p4, p5, p6, p7, p8 in parallel
For a valid move, $pi >= 0$ other wise pi = -1 $1 <= i <= 8$
Step 6.If g==p1or g==p2 or g==p3 or g==p4 or g==p5or g==p6 or g==p7 or g==p8 then
Print("Goal Reached")
Print the elements in visited[].
Step 7.Go to step17.
Step 8.Else
Step 9.If p1 >= 0 //valid move Store p1in visited[] s=p1
goto step 2
Step 10.else
If p2 >= 0 //valid move Store p2 in visited[] s=p2
goto step 2
Step 11.else
If p3>= 0 //valid move Store p3in visited[] s=p3
goto step 2
Step 12.else
If p4>= 0 //valid move Store p4 in visited[] s=p4
goto step 2
Step 13.else
If p5 >= 0 //valid move Store p5 in visited[] s=p5
goto step 2
Step 14.else
If p6 >= 0 //valid move Store p6 in visited[] s=p6
goto step 2
Step 15.else
If p7 >= 0 //valid move Store p7 in visited[] s=p7
goto step 2
Step 16.else
If p8 >= 0 //valid move Store p8in visited[] s=p8
goto step 2

step17. EndParallel_Knighttour()

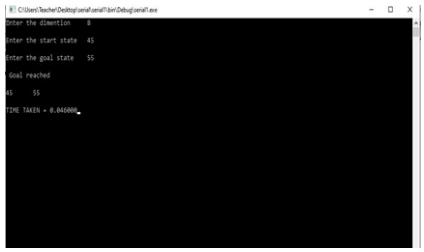
5. Results and Discussion

The same start state and goal state were tested in sequential and parallel programs [2]. The programs were executed on Intel® Core(TM) i3-5005U CPU @2.00GHz (4 CPUs) machine using Code block software and gcc compiler with windows 10 operating system. Table 1 [2] shows that time taken for serial and parallel algorithms with a given Start state and Goal State. It also shows the Number of intermediate state and the speedup. The chart [2] shows the time taken for each algorithm. "Speedup" refers to the ratio of the time taken to run a program on a single processor (serial) to the time taken to run the same program on multiple processors (parallel). A speedup of 2 means that the parallel program runs twice as fast as the serial program



(A) Figure 4.a

(B) Figure 4.b



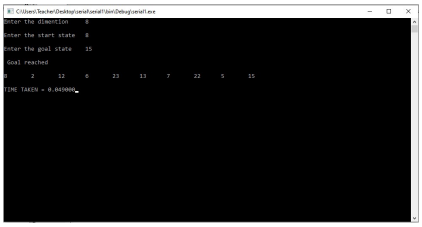
(C) Figure 4.c

(D) Figure 4.d

(E) Figure 4.e

(F) Figure 4.f

(G) Figure 4.g



(H) Figure 4.h

FIGURE 4. Screen shots of Sequential and Parallel Algorithm executions

TABLE 1. Table Title

S.No	Start State	Goal State	No. of States	Serial Time	Parallel Time	Speed Up	Efficiency
1	2	1	12	0.047	0.031	1.516	0.379
2	5	9	12	0.047	0.032	1.468	0.367
3	45	55	2	0.046	0.038	1.210	0.302
4	8	15	10	0.049	0.031	1.580	0.395
5	3	57	46	0.047	0.031	1.516	0.379
6	5	60	25	0.040	0.031	1.290	0.322

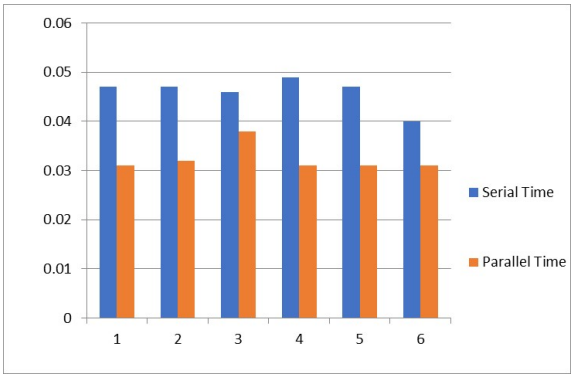


FIGURE 5. Execution time of serial and Parallel Algorithm of Knight Tour

From the figure 4 The result shows that the parallel algorithm is efficient than the Sequential algorithm.

6. Conclusion

In the multi-core PC or Laptop developing serial program does not utilize the resources' available. Parallel programming platform like OpenMP will utilize

the recourses' in Multi-core systems, so that we can increase the performance. The results of Knight tour problem proves this concept. The main contribution of this study is the implementation of a parallel algorithm using OpenMP to solve the Knight Tour problem on a multi-core system. The results show that parallel programming can significantly improve the performance of the application. One limitation of this study is that the implementation of the parallel algorithm is not optimized for all possible system configurations. The parallelization strategy, data distribution scheme, and synchronization mechanisms used in the OpenMP implementation were tailored to the specific multi-core system used in this study. Future work could include optimizing the implementation of the parallel algorithm to work on a broader range of multi-core systems, investigating the scalability of the parallel algorithm to larger problem sizes, and exploring the use of other parallel programming platforms and techniques. Additionally, the Knight Tour problem could be used as a benchmark for evaluating the performance of parallel algorithms on different multi-core systems .

Conflicts of interest : The authors declare no conflict of interest.

Data availability : Not applicable

Acknowledgments : We gratefully acknowledge Periyar University for providing the resources and support.

References

1. F. George, *Artificial intelligence structures and strategies for complex problem solving*, IV Edition, Pearson Education, 2007.
2. Vidyaathulasiraman, S. Vijayakumar, *An Algorithm to Avoid Backtracking for a Generalized Knight Tour Problem with Parallel Approach*, International Journal of Computational Intelligence and Informatics **9** (2019), 383-391.
3. Najem N. Sirhan, Sami I. Serhan, *Multicore processors: concepts and implementations*, International Journal of Computer Science & Information Technology (IJCSIT) **10** (2018), 1-10.
4. Musaev Muhammadjon Mahmudovich and Berdanov Ulug'bek Abdumurodovich, *The Technology of Parallel Processing on Multicore Processors*, International Journal of Signal Processing Systems **4** (2016), 252-257.
5. Balaji Venu, *Multi core processors An overview*, https://arxiv.org/ftp/arxiv/papers/1110/1110.3535.pdf, October 2011.
6. Norma Alias and Md. Rajibul Islam, *A Review of the Parallel Algorithms for Solving Multidimensional PDE Problems*, Journal of Applied Sciences **10** (2010), 2187-2197.
7. Mohd Muzafar Ismail, Ezreen Farina Shair, *A Preliminary Study on Solving Knight's Tour Problem Using Binary Magnetic Optimization Algorithm*, Science & Engineering Technology National Conference, 2013.
8. Sanjay Kumar Sharma, Dr. Kusum Gupta, *Performance Analysis of Parallel Algorithms on Multi-core System using OpenMP*, International Journal of Computer Science, Engineering and Information Technology (IJCSEIT) **2** (2012), 55-64.

9. G. Chen, X. Wu, and Z. Zhou, *A parallel algorithm for knight's tour problem on graphics processing unit*, 2014 IEEE International Conference on High Performance Computing and Communications, 2014, 955-961.
10. H.K. Kim, K. Kim, and S. Lee, *Parallelization of the Knight's Tour Problem Using CUDA*, Journal of Computational Science Education **6** (2015), 32-38.
11. H. Chen, Y. Zhang, and J. Huang, *Parallel Iterative Deepening Search Algorithm for Solving Knight's Tour Problem on GPU*, 2019 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2019, 111-116.

Vijayakumar Sangamesvarappa is a Ph.D. research scholar in the Department of Computer Science, Periyar University, Salem, Tamil nadu, India. He is working as Assistant professor in the Department of MCA, Priyadarshini Engineering College, Vaniyambadi, Tamil nadu, India.

Assistant Professor, Department of MCA, Priyadarshini Engineering College, Vaniyambadi, Tamil nadu, India.
e-mail: vijayviswak@gmail.com

Vidyaathulasiraman, Ph.D. is working as Assistant Professor, Department of Computer Science, Government Arts and Science College for Women, Bargur, India. She has vast experiences in the teaching field. She has published her papers in various conferences and journals.

Assistant Profssor, Department of Computer Science, Government Arts and Science College for Women, Bargur, India.
e-mail: vidyaathulasi@gmail.com