Literature Review For The Knight's Tour

Jien Wei, Lim

registration: 100276935

1 Introduction

The Knight's Tour is a sequence of moves made by a knight chess piece such that it visits every square of an $n \times n$ chessboard exactly once, with the tour being open or closed. This is an example of the Hamiltonian path/circuit problem where every vertex on a graph are visited once. Multiple sources have been read through to determine the algorithms, environments and visualization tools suitable to be used for the project.

2 Knight's Tour Solutions and Algorithms

2.1 Weisstein (2002)

While not an article in PDF format, it provided many references to articles and gave quick summary regarding solutions and findings by previous authors. This helped give a quick rundown of which article will be useful and which one will not. It was from this website where I found Conrad et al. (1994) that piqued my interest.

2.2 Conrad et al. (1994)

Written by Axel Conrad, Tanja Hindrichs, Hussein Morsy, and Ingo Wegener, the article explains how the Hamiltonian Path, and s-t Hamiltonian Path problem are solved. The Hamiltonian Path problem merely finds a knight's tour in a chessboard, whereas the s-t Hamiltonian Path problem decides whether a knight's tour from the source vertex, s, to the terminal, t, exists.

One of the theorems in the article states that G_n has a knight's tour if and only if $n \geq 5$. Since G_2 and G_3 have vertices impossible to be reached by the knight, a tour cannot be created. For G_4, there will always be at least one vertex that cannot be reached. Upon learning this, there program will need to check the input value to prevent the user from creating a board that is impossible to create a tour from.

The solution itself involves finding tours in $n \times n$ chessboards where $5 \leq n \leq 9$. If $n > 9$, an Γ-shaped section is "cut" out from the main chessboard, C_n, made up of 5×5 chessboards, and/or a Γ-shaped chessboard, leaving the remaining chessboard, C_l, where l is the length of the remaining board. The Γ-shaped section is then solved by finding a tour in each sub board using a solution given in the article's appendix. If $l > 9$, the process repeats. If not, C_l can be solved straight away. The algorithm definition states it performs one-to-one mapping. From what I understand, the algorithm operates by creating tours identical/mirrored to the solutions given in the appendix, meaning it will always check the vertex of the board in order to put the appropriate move number. While I believe the breaking down of the board is a good idea and can be used as part of the program, having to use predetermined tours to solve it may be difficult, as stated

by Weisstein (2002), the algorithm will be complex as it will have to deal with many special cases.

2.3 Ian Parberry

In Parberry (2020), many different solutions were given by Ian Parberry to solve the Knight's Tour such as the Join Algorithm, rails, tourney generation, etc. The main focus will be a divide-and-conquer algorithm presented by him in Parberry (1997).

Parberry (1997) provides a recursive algorithm that can generate a closed knight's tour on chessboards with size $n \times n$ or $n \times (n+2)$, where $n \geq 6$ and n is even, or size $n \times (n+1)$, where $n \geq 6$, in linear time $O(n^2)$. Smaller $n \times n$ or $n \times (n+2)$ chessboards, where $6 \leq n \leq 10$, are first used to create close knight's tours as a base. When solving a larger chessboard, it is divided into 4 quadrants. A closed knight's tour is built in each quadrant and 4 edges are removed to be replaced with another 4 edges at the inside corners of the quadrants to combine the smaller closed knight's tours to form the closed knight's tour of the larger board. The only issue with this algorithm is that it only works with the specified board size, and fails on boards that are more rectangular in shape.

2.4 Schwenk (1991)

This article presents a characteristic for rectangular boards where a closed knight's tour exists. On an $n \times m$ board, where $n \leq m$, a closed knight's tour exists unless it has one of the conditions: (1)Both n and m are odd, (2)n = 1, 2, or 4, (3)n = 3 and m = 4, 6, or 8.

2.5 Lin and Wei (2005)

Lin and Wei (2005) presents 2 algorithms for a stretched knight's tour and double-loop knight's tour. The stretched knight's tour is an open knight's tour with 2 extra conditions. The first being that the start and end points of the tour must be a corner square and adjacent square respectively. The second condition states that the other 3 corners that do not have the start and end points must have the requirements of a structured knight's tour, shown in Figure 1. The double-loop knight's tour is defined as a pair of closed knight's tours where each tour forms a Hamiltonian cycle that visits half of the squares on the board and occupies 2 adjacent squares per column. The double-loop tour must also satisfy the requirement of a structured knight's tour. Both of these algorithms have linear run time $O(nm)$ that solve the weakness of Parberry's algorithm where it does not work on rectangular-shaped boards.

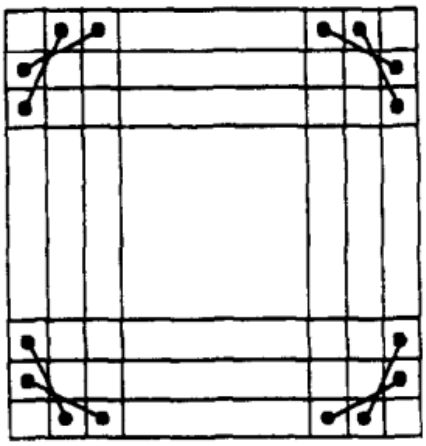

Figure 1: Moves required for a structured tour

3 Environments

With the different types of programming languages that currently exist right now, there are many that can be a possible candidate to be used for the Knight's Tour project. Due to having experience in Java, C#, and Python, I will be choosing one of the 3 to code the program with.

3.1 Python

Python is a high-level programming language released in 1991 that supports multiple programming paradigms such as Object-Oriented, Functional, and Procedural Programming. Its design philosophy emphasizes code readability with the use of indentation (Aruoba and Fernández-Villaverde, 2015).

Srinath (2017) states that its design concept prioritizes code readability and allows less code to be written compared to Java or C#. Python's lack of declaring data types for variables is a good example of it. Not only that, Srinath (2017) also mentions that Python contains many libraries that makes programming considerably easier by eliminating the need to write an entire code segment on your own.

Python is not without its drawbacks however. In an article by Aruoba and Fernández-Villaverde (2015), the computer has to reference many variables and objects to know their definitions when running a program which can slow Python down. So while versatility can be a good thing, too much of it can also be detrimental (Bahar et al., 2022). Not only that, Aruoba and Fernández-Villaverde (2015) also states that Python can be difficult to maintain as the program increases in complexity. Errors can be harder to find and fix which can make the code difficult to manage.

3.2 C#

C# (C-sharp) is an Object-Oriented Programming (OOP) language released in 2000 that uses the .NET Code framework, primarily used with Microsoft's Visual Studio program for visual programming (Bahar et al., 2022). Some of its major uses include desktop applications and game development.

While C# does not have the luxury of Python's less-is-more approach in terms of coding size, C# does have an easier time integrating with visual programming as Visual Studio has a drag-and-drop system for creating desktop applications without needing to code. Because C# is an OOP language, coding can be efficient and is easy to maintain (Payne, 2021).

However, Payne (2021) states that C# has slow load times as any changes to code will result in having to recompile, wasting computer resource. Also, the drag-and-drop way for creating desktop applications is only exclusive for Windows and not available for Linux and Mac, which means unless a Windows machine is available, another .NET Core Framework is required to be learned and used in order to create a Graphics User Interface (GUI).

3.3 Java

Java is an OOP language released in 1995 by James Gosling and his team. In order to use it, all the system requires is a Java Virtual Machine.

Like C#, Java is an OOP language, meaning it is efficient and code is easy to maintain. Not only that, Java is robust. Java checks code during compile time and run time which means many hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java (Schildt, 2020).

However, Java has 2 flaws that may make it the last choice for the project which is execution speed due to compilation of Java code to bytecode and difficulty of creating GUIs.

4 Graphical Tools

4.1 PyQt (Harwani, 2018)

PyQt is the combination of Python and the Qt library resulting in a toolkit for GUI widgets. One advantage of PyQt is that it has a Qt Designer tool that can be used to build GUIs rather than typing out the code. PyQt contains various UI widgets (buttons, menus, etc.) having a basic styling ready to be used. If required, they can be customized easily. Finally, it allows styling and widgets to be customized during runtime, allowing customization even while the application is running.

A main issue of PyQt is that it is not fully open source and some things may require paying for a commercial license in order to use them. With the amount of components in PyQt, there may be a steep learning curve in order to fully grasp it.

4.2 Tkinter (Lundh, 1999)

Tkinter is also a Python GUI library that is well known for its simplicity. It comes installed with Python so a separate installation is not required. The good thing about Tkinter is that it is fast to implement the UI as it is already integrated with Python. With the simplicity of Tkinter syntax, it is easy to understand it and use.

Although it is fast and simple to use, the projects that it can handle are also required to be relatively simple as well. Large-scale projects cannot be handled by the library. Unlike PyQt, Tkinter does not have a UI designer tool to help build UIs and does not allow UI customization during runtime.

4.3 JavaFX (Fetter et al., 2017)

JavaFX is a Java library and a GUI toolkit designed to develop and create web and desktop applications. It is very versatile as it can be run on different operating systems like Windows and Linux. It contains a number of APIs, classes, and interfaces that are sufficient to create GUI applications with intense graphics.

While JavaFX has the versatility and sufficiency for GUIs, it does have a steep learning curve.

References

Aruoba, S. B. and Fernández-Villaverde, J. (2015). A comparison of programming languages in macroeconomics. *Journal of Economic Dynamics and Control*, 58:265–273.

Bahar, A. Y., Shorman, S. M., Khder, M. A., Quadir, A. M., and Almosawi, S. A. (2022). Survey on features and comparisons of programming languages (python, java, and c#). In *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, pages 154–163. IEEE.

Conrad, A., Hindrichs, T., Morsy, H., and Wegener, I. (1994). Solution of the knight's hamiltonian path problem on chessboards. *Discret. Appl. Math.*, 50(2):125–134.

Fetter, M., Bimamisa, D., and Gross, T. (2017). Tuiofx: A javafx toolkit for shared interactive surfaces. *Proc. ACM Hum.-Comput. Interact.*, 1(EICS).

Harwani, B. M. (2018). *Qt5 Python GUI Programming Cookbook: Building responsive and powerful cross-platform applications with PyQt*. Packt Publishing Ltd.

Lin, S.-S. and Wei, C.-L. (2005). Optimal algorithms for constructing knight's tours on arbitrary n× m chessboards. *Discrete applied mathematics*, 146(3):219–232.

Lundh, F. (1999). An introduction to tkinter. *URL: www. pythonware. com/library/tkinter/introduction/index. htm*.

Parberry, I. (1997). An efficient algorithm for the knight's tour problem. *Discret. Appl. Math.*, 73(3):251–260.

Parberry, I. (2020). Tourneys and the fast generation and obfuscation of closed knight's tours. *CoRR*, abs/2001.06044.

Payne, J. (2021). Benefits of c#. *Retrieved from TechnologyAdvice: https://www. code-guru. com/csharp/benefit s-of-c*.

Schildt, H. (2020). The complete reference java.

Schwenk, A. (1991). Schwenk, a.j.: Which rectangular chessboards have a knight's tour? math. mag. 64, 325-332. *Mathematics Magazine*, 64.

Srinath, K. (2017). Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, 4(12):354–357.

Weisstein, E. W. (2002). Knight graph. *https://mathworld. wolfram. com/*.

Literature review

Introduction: brief description of project, areas of knowledge required, roadmap	First	2.1	2.2	3	Fail
Discovery of suitable quantity and quality of material	First	2.1	2.2	3	Fail
Description of key issues and themes relevant to the project	First	2.1	2.2	3	Fail
Evaluation, analysis and critical review	First	2.1	2.2	3	Fail

Quality of writing

Clarity, structure and correctness of writing	First	2.1	2.2	3	Fail
Presentation conforms to style (criteria similar to conference paper reviews)	First	2.1	2.2	3	Fail
References correctly presented, complete adequate (but no excessive) citations	First	2.1	2.2	3	Fail

Revised Workplan (if applicable)

Measurable objectives : appropriate, realistic, timely	First	2.1	2.2	3	Fail

Comments

Supervisor: Dr. Katharina Huber

Markers should circle the appropriate level of performance in each section. Report and evaluation sheet should be collected by the student from the supervisor.