Implementation of Graph in Artificial Intelligence Path Finding

Dariel Valdano - 13512079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
dariel.valdano@students.itb.ac.id

Abstract**—*****Computer games has been around for many decades that contains various kinds of Artificial Intelligence. These Artificial Intelligence system is often very sophisticated, even for games. This paper will be focused on the problem of adjusting Artificial Intelligence (AI) Path Finding on a Game Space for real world applications using some sample implementation of Graph and Tree that is used to calculate the path. An imaginary hypothetical machine will be used to explain the problems and implementation algorithm-wise, including some illustrations to illustrate the problems. The real world implementation may include autonomous transportation system and security patrol robots.***

Index Terms**—Artificial Intelligence Path Finding in Three Dimensional Space, Using Graph for Artificial Intelligence Path Finding, Real World implementation of Artificial Intelligence Path Finding**

I. Introduction

Computer games has been around for many decades. Almost all of them has implemented some sort of Artificial Intelligence system to give Non-Player Character (NPC) a reasonable, human-like actions to their decision making, including –but not limited to- path finding. Whether it be a turn-based strategy game to Role-Playing (RPG) games, Artificial Intelligence implementation is as old as the first computer game itself. Even though there are numerous examples of games with excellent artificial intelligence, there is much less real world implementations of such artificial intelligence algorithms. There are many reasons why artificial intelligence algorithms were rarely used in real world, one of which is because of hardware limitations and the fact that most game path finding algorithm is for 2 dimensional space. For example, in game all of the NPC's location and status were explicitly given and stored in memory, such as altitude, speed, heading and status such as threat level, focus level and much more. In the real world however, physical properties such as these are quite hard to obtain in real time, requiring precision instruments and expensive hardware. Setting aside hardware difficulties as they are outside the scope of this paper, the algorithm behind Real Life artificial intelligence is quite similar with game artificial intelligence.

There were already quite a numerous amounts of graph-based path finding algorithm, perhaps some of the most famous examples being Dijkstra's algorithm and its extension, the A* (A-Star) search algorithm. Both of these algorithm calculates the shortest route possible from a specific vertex in a graph to another vertex. For implementations that require all vertex to be visited, such as a security patrol robot, Hamilton circuit of a graph can be calculated and used, or alternatively, an Euler Circuit can be calculated and used if the patrol requires all edges (routes) to be visited in each loop.

II. Graph Theory

Graphs are a way of representing discrete structures and their relations using multiple vertices and edges that connects them. It is a representation that is often used in various areas of science, such as the IT field for computer network interconnection, the civil engineering field for intercity roadways, in biology for ecological food mesh, even in arts for 3d modelling and computer sculpting. Graph allows arbitrary data to be represented in a more understandable manner that can be easily computed by humans and machines alike.

A. Definition of Graph

A graph G = (V, E) consists of V, a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints. [1]

B. Categories of Graph

Seen from the existence of double edge in a graph, a graph can be categorized as Simple Graph – where a graph does not contain a loop or dual edge, and Unsimple Graph – where a graph contains either a loop or a dual edge.

Seen from the amount of vertices, a graph can be categorized as a Limited Graph – where vertices inside a graph is not infinity, and Unlimited Graph – where vertices inside a graph is infinite.

Seen from the directionality of edges, a graph can be categorized as an Undirected Graph – where there is no directed node present in a graph, and Directed Graph – where directed nodes are present.

The types of graph is listed on table 1

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simple Graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple Directed Graph	Directed	No	Yes
Directed Multigraph	Directed	Yes	Yes
Mixed Graph	Directed and Undirected	Yes	Yes

TABLE 2.1 – Types of Graph [1]

C. Euler and Hamilton Path

Euler and Hamilton path is a way to define a unique set of path or circuit that will be very useful in path finding.

An Euler circuit in a graph G is a simple circuit containing every edge of G [1]. An Euler circuit will travel each edge in graph G exactly once and ends back at the first vertex, and is a very efficient way to "patrol" a pre-defined route graph. In the other hand, An Euler path in G is a simple path containing every edge of G [1].and does not have to be a closed circuit, the final vertex does not have to be the first one. This is less efficient, but a good alternative. A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree [1].

A simple path in a graph G that passes through every vertex exactly once is called a Hamilton path, and a simple circuit in a graph G that passes through every vertex exactly once is called a Hamilton circuit [1]. A Hamilton path and circuit can also be used as an alternative to patrolling a building, with each room represented as a node.

D. Dijkstra's Algorithm

Dijkstra's Algorithm is a way to calculate the shortest route or path from node to node in a graph. It is invented by computer scientist Edsger Dijkstra in the 1959. The pseudo code algorithm of this algorithm is included in Appendix A.

III. Artificial Intelligence

Artificial Intelligence, or AI for short, is a computer-based intelligence system. Currently most AIs were constructed with specific tasks in mind. This allows an easier algorithms, since they only need to focus to one job they are specifically made for. Human-Equivalent AI system that can think and learn exactly like a human is still the forefront of current technology.

There are currently a lot of applications of Artificial Intelligence; Expert Systems, Computer Vision, Heuristics Classification, Speech Regocnition, Game Playing, and many more.

AIs were generally a set of programs in an operating system that is run on-demand, or it can also be a dedicated computer platform specifically constructed for the AI. Perhaps a very popular example of a platform created specifically for an AI is the IBM's Deep Blue chess computer. In 1997, Deep Blue, using raw brute force power of its 11.38 GFLOPS[1] processor, won a six-game match against world champion Garry Kasparov.

IV. Path Finding Implementations

A. Building Room-to-Room Patrol

First, we will see how a graph can be used to find a route for an automated machine to patrol a multi-story building with multiple rooms. For an example, imagine the following building and its graph equivalent; a two-story building with multiple rooms in it that must be patrolled. Here is the first floor:

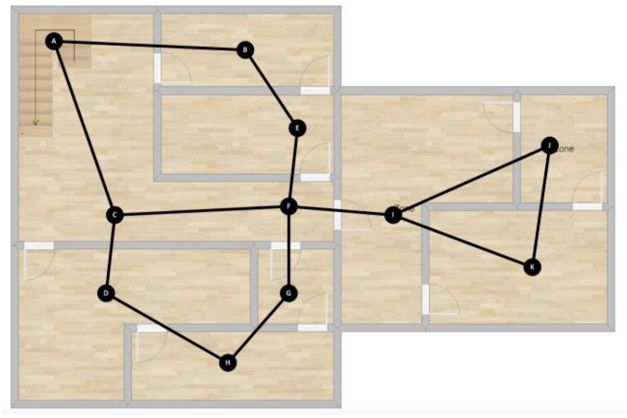

Figure 3.1 – The First Floor

Then, the second floor:

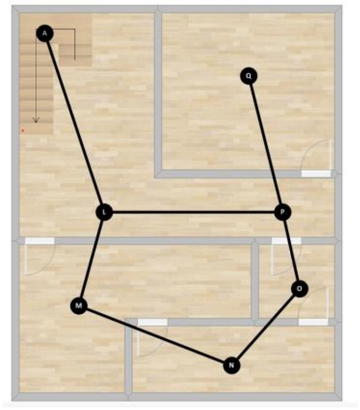

Figure 3.2 – The Second Floor

[1] FLOPS – Floating Point Operations per Second, a measure of computer performance that counts maximum floating point calculations per second (GFLOPS – Giga-FLOPS)

Both floor is linked with a staircase in the northwest corner of each floor. This example did not weigh the edges, because the current example emphasizes on visiting each room at least once, and at most twice. With the graph system laid out that points out the connection between rooms, we can merge both room into a single graph, illustrated below:

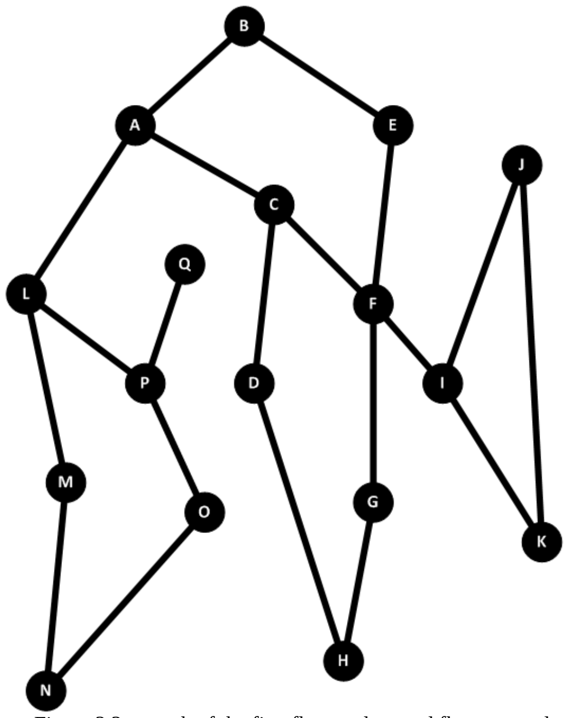

Figure 3.3 – graph of the first floor and second floor merged

Here, each room is represented by a vertex, with vertex A being the staircase, while the doors between rooms is represented by an edge between the vertices. After merging both graph, then we can find a route where each room is only visited once.

Looking at the graph, and referring to the definition of Hamilton circuit or path, we can conclude that it does not have a Hamilton circuit or path, since Vertex Q is a Vertex of degree one. Furthermore, even if Q is does not exist, a Hamilton circuit is still impossible to obtain. Look at the vertices I, J and K. vertices J and K is a degree two vertex, which means to have a Hamilton circuit both must be connected to another vertex of degree two, which is not the fact, since vertex I is a degree three vertex.

Because a Hamilton circuit or path is impossible to obtain, we now try to find an Euler circuit or path. Unfortunately, finding an Euler circuit is also impossible for this graph, because there exist vertex Q that is a degree one vector. Furthermore, even if Q did not exist, according to the a Theorem of Euler Circuit that states "A connected undirected graph G have an Euler Path if and only if there exist two odd-degree vertex in G [3]", there is 6 odd-degree vertex in the graph. In conclusion, an Euler Path is also an impossibility.

When neither Hamilton path nor Euler path can be used to find a route to patrol the room, the algorithm falls back to the last resort, brute-forcing the routes to list every possible route that travels every room, then find one that travels each room the least, hence most efficient.

Hence, the algorithms that can be used for patrolling the building is as follows, listed from most preferred to less preferred:

1. Hamilton Circuit
2. Hamilton Path
3. Euler Circuit
4. Euler Path
5. Brute-Force Listing

B. Building Target Intercept

The second example will explain the method to intercept a target in a specific room from any other room using the fastest way possible. Consider the following figure:

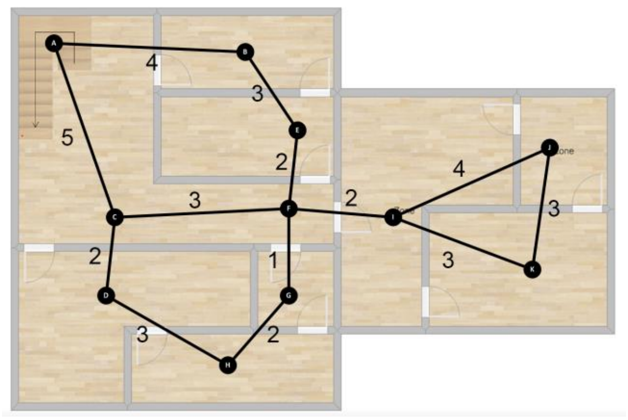

Figure 3.4 – The first floor, now weighted

This is the first floor of the same building, now weighted with the time needed to travel from each rom to the one connected to it. Now for example the security bot is currently in position A, just at the base of the stairs, while an intruder is found at room K. To find the most efficient path from A to K, Dijkstra's Algorithm is used.

First the algorithm marks A as solved, then find the length of vertices connected to the solved vertex. First the algorithm checks A-B, and puts the value 4 to B. then it checks A-C, and puts the value 5 to C. it then chooses the least value between the two, A-B. it then adds edge A-B to a set of solution edge. Then it iterates again, finding the length of vertices connected to each solved vertex and repeating the process all over, until it founds a solution, which will result in the most optimum path, A-C-F-I-K. (The Dijkstra's Algorithm Pseudocode is shown in Appendix A)

Optimizations to the Dijkstra's Algorithm such as using min-priority queue implemented by a Fibonacci heap could further decrease the time needed to compute a fastest path. Originally Dijkstra's Algorithm runs in $O(|V|^2)$ where V is the number of vertices, while the optimized Dijkstra could run in O(|E|+|V| \log|V|)

V. Conclusion

It can be concluded that finding a suitable route for building patrol can be found using some checks. First, check for Hamilton Circuit, as it is the most efficient patrolling method. If the shape of the building does not allow Hamilton circuit, try finding Hamilton path. If it still does not allow Hamilton path, check for Euler circuit, if still impossible, Euler path, and if still impossible, check using brute-force technique to list all the possible ways, then find one that least visits the same node.

Other than games, all of the aforementioned algorithms can be used in real life, especially in the field of security, due to the fact that it will find the optimum way to either patrol a building or intercept a room in the quickest way possible. Other than security, many fields can still benefit from the use of AI Path Finding. For example, automated delivery vehicles that delivers products to locations around a city, can benefit with this path finding with the vertex being street intersections and edges as roads. Other field of implementation might be in personal transport, with graph representation similar as with the automated delivery systems. Other than Artificial Intelligence, it even find its way in even more applications. It can even be applied for recreation. For example, when one wants to have some sightseeing on a mountain he/she can first plot the most direct route to reach all sightseeing locations in minimum time using this path finding algorithm

VII. Appendix

A. Dijkstra's Algorithm Pseudocode

The Dijkstra's Algorithm Pseudocode, taken from [3]

```
Procedure Dijkstra (input m: matrix, a: first node)
{ To find the shortest distance from first vertex a to all other vertices.
  Input: adjacency matrix (m) from a weighted graph G and first vertex a
  Output: shortest route from a to all other vertices
}

DICTIONARY
  $S_1$, $S_2$, ..., $S_n$ : integer {Array of Integer}
  $D_1$, $D_2$, ..., $D_n$ : integer {Array of Integer}

ALGORITHM
  {Initialization}
  For i←1 to n do
    $S_i$ ← 0
    $D_i$ ← $M_{ai}$

  {Step 1}
  $S_a$ ← 1 {for the first step, the initial vertex must be the shortest}
  $D_a$ ← ∞ {there is no shortest route from vertex a to a}

  {Step 2, 3, ..., n-1}
  For i←2 to n-1 do
    Find j so that $S_j$ = 0 and $D_j$ = min($D_1$, $D_2$, ..., $D_n$)
    $S_j$ ← 1 {j chosen as the shortest route}
    Renew $D_i$, for i=1, 2, 3, ..., n with $D_i$(new)=min { $d_i$(old), $d_j$ + $m_{ij}$ }
```

VIII. ACKNOWLEDGMENT

First, the author would like to express his highest praise to God Almighty for the grace and enlightenment that allows me to complete this paper. Author would also like to give his biggest thanks to his parents, who's without their guidance, the author might not be able to have a chance in studying this high. Author would also like to express his thanks to Mr. Rinaldi Munir, for his teachings allows the author to understand the concepts behind Discrete Mathematics, including the theory of Graph, which this paper is based on.

REFERENCES

[1] K. H. Rosen, "Discrete Mathematics and its Applications" 6th ed. New York: McGraw-Hill, 2007, pp. 589-682

[2] http://www-formal.stanford.edu/jmc/whatisai/whatisai.html retrieved on 2013-12-16 10.35PM

[3] R. Munir, Diktat Kuliah Struktur Diskrit. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2008 Ch. 8

[4] Leyzorek, M.; Gray, R. S.; Johnson, A. A.; Ladew, W. C.; Meaker, Jr., S. R.; Petry, R. M.; Seitz, R. N. (1957). *Investigation of Model Techniques — First Annual Report — 6 June 1956 — 1 July 1957 — A Study of Model Techniques for Communication Systems*. Cleveland, Ohio: Case Institute of Technology.

[5] Fredman, Michael Lawrence; Tarjan, Robert E. (1984). "Fibonacci heaps and their uses in improved network optimization algorithms" IEEE. pp. 338–346.

STATEMENT

I hereby declare that I wrote this paper with my own writing, not adaptation, or translation of someone else's paper, and not plagiarism.

Bandung, 16 Desember 2013

[illegible]

Dariel Valdano - 13512079