Modeling Character Recovery Option in Super Smash Brothers Ultimate on Survivability using Directed Graph

Mahmudia Kimdaro Amin - 13524083
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mkaxydaro@gmail.com , 13524083@std.stei.itb.ac.id

Abstract*— This paper models character survivability in Super Smash Bros. Ultimate by applying graph theory to recovery mechanics. We categorize character recovery options into archetypes, representing each as a directed graph where nodes signify states and edges represent player actions. By enumerating all successful recovery paths using a Depth-First Search algorithm, we establish a quantitative link between the number of available options and a character's survivability. The results show that a higher number of recovery paths correlates with greater survivability, offering a formal method for analyzing this critical gameplay element and demonstrating the utility of discrete mathematics in interpreting complex game systems ****CRITICAL: Do Not Use Symbols, Special Characters, or Math in Paper Title or Abstract.*** **(*Abstract*)**

Keywords—component; formatting; style; styling; insert (key words)

I. Introduction (*Heading 1*)

Video games are games that are enacted electronically through various inputs methods, that have an extensive range of genres including action, sports, puzzles, simulations and much more to be seen. It originates from early computer science experiment and has evolved into a multibillion-dollar industry with high production value, deep and intricate gameplay, and big cultural influence. Players can enjoy video games on multiple platforms, including consoles, personal computers, and mobile device making them accessible to a wide audience. The popularity of video games has given rise to organized competitive play that are also known as esports. As of the 2020s, video games remain a dynamic sector of entertainment, continuing to spark discussions about their role in society and culture.

One of such video games is Super Smash Brothers Ultimate. Super Smash Brothers Ultimate is a crossover fighting game developed by Bandai Namco Studios and Sora Ltd. and published by Nintendo at 2018. Super Smash Bros. Ultimate received critical acclaim from both critics and players, with some critics calling it the best installment in the series. It received praise for its large amount of content and fine-tuning of existing Smash gameplay elements, although its online mode was widely criticized. Ultimate is currently the best-selling Super Smash Bros. game and also the best-selling fighting game of all time, beating Super Smash Bros. Brawl and Street Fighter II, which previously held each title respectively. It is also the third best-selling game on the Nintendo Switch. Its massive success has caused it to be nominated for, and win, multiple awards, including winning "Best Fighting Game" at The Game Awards 2019.

Super Smash Brothers Ultimate has a important mechanic of not getting launched off the stage into the border of the map. Getting your character knocked off to the border or fall to the bottom border will result on your character losing a life which may cause you losing the match or put you in a disadvantage. Each character all have a way to return after getting launched offstage depending on each character. The attempt of trying to get back on stage after being launched or falling of is called a recovery.

This paper is trying model some variation of every character recovery with the concept of graph, mainly directed graph and combinatorics. By finding the amount of way a character can try to go back into the stage, it could be inferred a way to check the survivability of the character when they are offstage.

II. Theoritical Framework

A. Graph

A graph is a mathematical structure that is used to represent discrete objects and the relations between those objects. A graph may represent those object using vertices and the relations between those objects as edges. Mathematically graph is defined as G = (V, E) in which V is a not empty set of vertices and E is a set of Edges that may be empty.

A graph could be categorized based on its simplicity or it's direction. If we categorized by simplicity there are two types of graphs, simple graph or non-simple graph. A simple graph is a graph that must not contain any looping edges nor multiple edges between two same vertexes. The opposite is what we call a non-simple graph. A non-simple graph may contain either a looping edge or multiple edges between two same vertexes. The example for simple graph could be seen in Fig A.1 and a non-simple graph on Fig A.2

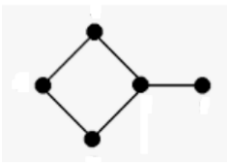

Fig 1 Simple Graph

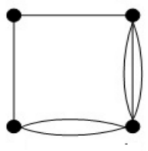

Fig 2 Non-Simple Graph

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Another way to categorized graph is through is through its orientation with direction. There are two types of graphs if we categorized it that way. If the graph doesn't have directed edges it would be called undirected graph, if the graphs has directed edges it would be called directed graphs. Examples for the directed and undirected graphs could be seen in the following Fig A.3 and Fig A.4.

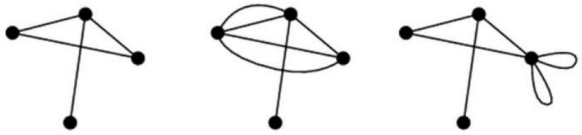

Fig 3 Directed Graphs
Source:

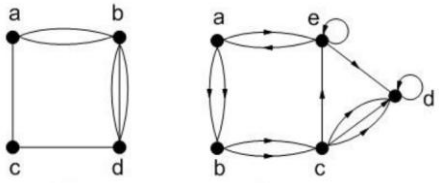

Fig 4 Undirected Graphs
Source:

A.1 Graph Terminology

a) Adjacent

Two vertices in a graph are considered as adjacent if there is and edge connecting the both of them. This indicates a relationship between the vertices. An example that we could take is when a player is playing Super Smash Brothers Ultimate and the possible positions the player character could take after moving while in a recovery.

b) Incidency

The relationship between a vertex and an edge is called an incidency. A vertex could be called incident with an edge if the edge is connected directly with a vertex. For an example vertex A has edge connecting it directly to vertex B, the edge connecting the can be called incident with vertex A or incident vertex B

c) Degree

The degree of a vertex is the amounts of edges that are incident to itself. If a vertex has two edges it has incident in then the amount of degree the vertex has is two. In directed graphs, the degree is further classified as in-degree, the number of edges directed to itself, and an out-degree, the number of edges directed from a vertex to out of itself.

d) Path

A path in a graph is described as a sequence of vertex nnected by edges. It has a start point v_0 and an endpoint v_n and : vertex is can be connected through a sequence of vertex and edges. It must be written so it could made v_0, e_1, v_1, .. e_n, v_n. and be made into $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, … $e_n = (v_{n-1}, v_n)$.

B. Depth First Search Algoritm

Depth-first search algorithm tries explores edges out of the most recently discovered vertex that is still has not explored edges leaving it. Once all of the edges have been explored, the search "backtracks" to explore other edges leaving the vertex from which it was discovered. This process continues on until it has discovered all the vertex that are can be reached from the original source vertex

C. Super Smash Brothers Ultimate

Super Smash Brothers Ultimate often shorten to "SSBU" or Ultimate is a crossover action fighting game released for the Nintendo switch. The game was first teased at the end of Nintendo Direct on March 8th, 2018, and fully revealed on June 12th at E3 2018. It is the fifth installment in the Super Smash Brothers series. The game was finally released worldwide on December 7th, 2018.

Super Smash Brothers has a unique gameplay compared to other fighting games. Normal fighting games usually has healthbars and we need to reduce the enemy healtbars to win a match, Super Smash Brothers instead has the concept of percentage. When a character got hit in Super Smash Brothers by the enemy, they slowly rack up percentages. The higher the character percentages, the higher the push back received by the character when they get hit. winning a match in Super Smash Brothers is by making the enemy character reach the Border of the map. It could be achieved by either knocking them there or making them fall of the stage. Once they get knocked into the border they lose a stock (stock is what lives are called in this game), if they have another stock they would respawn with 0 percentages. Once their stock reaches one and they get knockout in the border they lose the match.

Most of the map in Super Smash Brothers features a floating platform in the middle and border offscreen. Character usually fights on the map mid-air or on the middle platform. The platform usually has ledges that a character can hang on to if they comeback from being pushed offstage. If a player got their character knocked offstage the player must put an effort to get back to the stage. The effort is called Recovering

D. Recovery (Super Smash Brothers).

In Super Smash Brothers a recovery is considered an attempt to return to the stage after being launched or falling off. Some tools to achieve this end are the mid-air jump and the recovery move (usually an up special move), also known a "third jump". For some character recovery can be augmented by other moves other than one mid-air jump. Some character may contain multiple mid-air jump.

III. Super Smash Brother Terminologies

This section is trying to provide some relevant terminologies that are commonly used by the community of Super Smash Brothers. Before proceeding, It is needed to be mentioned that some of the terminologies are not official and aren't universally used by the whole community nor Nintendo.

1) Mid-air Jump

Mid-air jump also called a double jump or air jump is a jump that can be used once every time the user is airborne. Every character in the game has at least one double jump in the game. Some character has multiple of this and can use up to 5 mid-air jump.

2) Up Special Move

Up special move, commonly referred to as Up B, is a special attack perfomed by pressing the special move button with the control stick aimed upward. Up special moves are usually the main aspect of a character recovery. Using them typically covers most of the vertical distance out of the character ability, but also usually render them helpless or as we call in freefall.

3) Freefall.

Freefall or also called special fall and is called "fallspecial" in the game data is a state in which a character are unable to take action, with the exception of grabbing a ledge, maneuvering left/right, fastfalling or climbing a ladder, until they land on the ground or pass the map border. It is usually caused by using Up special.

4) Directional Air Dodge

In Super Smash Brothers Ultimate by clicking the button for shield in mid-air you can perform something called an air dodge. After committing an air dodge the character will receive a time where they are invincible. If the player flick the control stick towards a direction. The character will be propelled toward that direction but it would cause a delay on the next move.

IV. Mathemttical Modelling Graphs From Recovery

Mathematical Modeling is the process of describing a real-world problem in mathematical terms, usually in the form of equation, and using this model to help us understand the original problem and also discover new features about the problem.

To make the problem more bearable the type of way a character could recover will be categorized into 6 different categories and then will be modeled into a directed graph.

1. One Jump and Freefall Up-b
2. Multiple Jump, Freefall Up-b
3. Multiple Jump, No Freefall
4. One Jump, No Freefall.
5. One Jump, Tether.
6. One Jump, Tether, and Freefall Up-b.

With those 5 categorization, it will be modeled a graph where each vertex would represent the state of the character and directed edges would be used to convey the next state that can be done. Below could be seen the modeled graph for each archetypes. In each graph the first vertex is a representation of the character position as being offstage

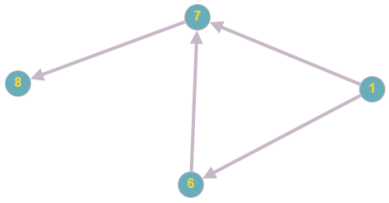

Fig 5. Archetype 1 Graph

In Fig 5, the vertex with the number 6 is used to represent the character state of using mid-air jump and number 7 is used to represent it state after using Up-B. Vertex 8 represent the character going to the ledge. The character chosen to represent this archetype is Ganondorf.

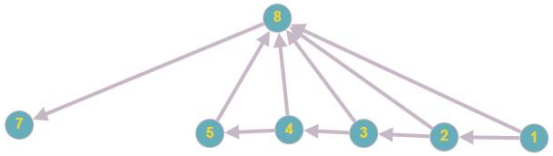

Fig 6. Archetype 2 Graph

Fig 6 represents the second archetypes. Vertex 2 to 5 are used to represent the mid-air jumps of character. Vertex 8 represents Up-B and Vertex 7 represent the character on stage. The character chosen to represent this archetype is meta-knight

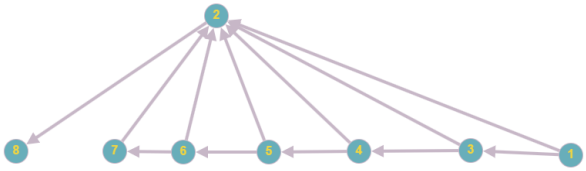

Fig 7. Archetype 3 Graph

Fig 7 represents the third archetypes. Vertex 3 to 7 are used to represent the mid-air jumps of character. Vertex 2 represents charcter using another ability and Vertex 8 represent the character on stage. The character chosen to represent this archetype is jigglypuff.

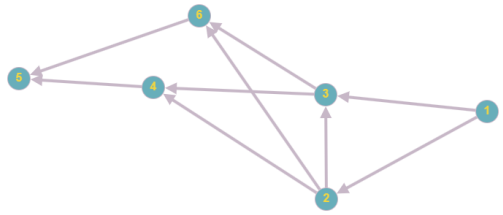

Fig 8. Archetype 4 Graph

Fig 8 represents the fourth archetypes. Vertex 2 represent mid-air jumps, vertex 3 represent Up-B, Vertex 4 represent directional airdodge and vertex 5 represent aerial attack. The character that is chosen for example is mega-man

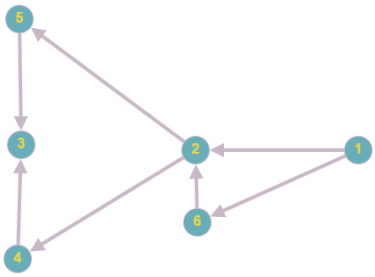

Fig 9. Archetype 5 Graph

Fig 9 represents the fourth archetypes. Vertex 6 represent mid-air jumps, vertex 2 represent Up-B tether, Vertex 4 failed tehtering and vertex 5 represent successful tether. The character that is chosen for example is joker

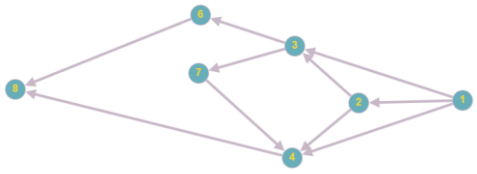

Fig 10. Archetype 6 Graph

Fig 10 represents the sixth archetypes. Vertex 2 represent mid-air jumps, vertex 4 represent Up-B , Vertex 3 represent tether attempt, Vertex 7 failed tehtering and vertex 6 represent successful tether. The character that is chosen for example is Samus

V. Analyzing Recovery Ways

Based on the graph made from modelling, a python program can be made to find out how many ways can a character from one of the categories to return back to the stage. Below is the provided source code for finding the amount of way each categories can return back onstage.

```
def find_recovery_paths(graph, start_node,
success_nodes):

    path_count = 0
    found_paths = []
    stack = [(start_node, [start_node])] # Stack stores
(current_node, path_taken)

    while stack:
        current_node, path = stack.pop()

        if current_node in success_nodes:
            path_str = " -> ".join(path)

            is_subpath = any(path_str in p for p in
found_paths)
            if not is_subpath:
                print(f"Found successful path: {path_str}")
                found_paths.append(path_str)
                path_count += 1
            continue

        for neighbor in graph.get(current_node, []):
            new_path = path + [neighbor]
            stack.append((neighbor, new_path))

    return path_count, found_paths

ganondorf_graph = {
    'Off-Stage': ['Mid-Air Jump Used', 'Up-B (Free
Fall)'],
    'Mid-Air Jump Used': ['Up-B (Free Fall)'],
    'Up-B (Free Fall)': ['Grab Ledge', 'Fall'],
    'Grab Ledge': ['On Stage']
}

meta_knight_graph = {
    'Off-Stage': ['Jump 1', 'Up-B (Free Fall)'],
    'Jump 1': ['Jump 2', 'Up-B (Free Fall)'],
    'Jump 2': ['Jump 3', 'Up-B (Free Fall)'],
    'Jump 3': ['Jump 4', 'Up-B (Free Fall)'],
    'Jump 4': ['Up-B (Free Fall)'],
    'Up-B (Free Fall)': ['Grab Ledge', 'Fall'],
    'Grab Ledge': ['On Stage']
}
```

```
jigglypuff_graph = {
    'Off-Stage': ['Jump 1', 'Pound'],
    'Jump 1': ['Jump 2', 'Pound'],
    'Jump 2': ['Jump 3', 'Pound'],
    'Jump 3': ['Jump 4', 'Pound'],
    'Jump 4': ['Jump 5', 'Pound'],
    'Jump 5': ['Pound'],
    'Pound': ['Grab Ledge'],
    'Grab Ledge': ['On Stage']
}
mega_man_graph = {
    'Off-Stage': ['Mid-Air Jump Used', 'Up-B (Rush
Coil)'],
    'Mid-Air Jump Used': ['Up-B (Rush Coil)'],
    'Up-B (Rush Coil)': ['Air Dodge to Ledge',
'Aerial Attack to Ledge'],
    'Air Dodge to Ledge': ['Grab Ledge'],
    'Aerial Attack to Ledge': ['Grab Ledge'],
    'Grab Ledge': ['On Stage']
}

joker_graph = {
    'Off-Stage': ['Mid-Air Jump Used', 'Use Tether'],
    'Mid-Air Jump Used': ['Use Tether'],
    'Use Tether': ['Tether Connects', 'Tether Misses'],
    'Tether Connects': ['On Stage'],
    'Tether Misses': ['Air Dodge to Ledge'],
    'Air Dodge to Ledge': ['Grab Ledge'],
    'Grab Ledge': ['On Stage']
}
samus_graph = {
    'Off-Stage': ['Mid-Air Jump Used', 'Up-B (Free
Fall)', 'Use Tether'],
    'Mid-Air Jump Used': ['Up-B (Free Fall)', 'Use
Tether'],
    'Up-B (Free Fall)': ['Grab Ledge', 'Fall'],
    'Use Tether': ['Tether Connects', 'Tether Misses'],
    'Tether Connects': ['On Stage'],
    'Tether Misses': ['Up-B (Free Fall)'],
    'Grab Ledge': ['On Stage'],
}
character_graphs = {
    "1. One Jump, Free-Fall Up-B (Ganondorf)":
ganondorf_graph,
    "2. Multiple Jumps, Free-Fall Up-B (Meta
Knight)": meta_knight_graph,
    "3. Multiple Jumps, No Free-Fall (Jigglypuff)":
jigglypuff_graph,
    "4. One Jump, No Free-Fall (Mega Man)":
mega_man_graph,
    "5. One Jump, Tether Up-B (Joker)":
joker_graph,
    "6. One Jump, Tether & Free-Fall Up-B
(Samus)": samus_graph,
}
```

```
SUCCESS_STATES = ['Grab Ledge', 'On Stage']
START_STATE = 'Off-Stage'

for name, graph_data in character_graphs.items():
    print(f"--- Calculating Paths for Archetype: {name} ---
")
    count, _ = find_recovery_paths(graph_data,
START_STATE, SUCCESS_STATES)
    print(f"\n>>> Total successful recovery paths:
{count}\n{'='*60}\n")
```

this program uses the models each possible action as part of a directed graph and then searches through all the paths that begin off-stage and return onstage. This python program Depth-First Search (DFS)

Its starts by setting up two variables: a counter called path_count, which will keep track of the number of successful recovery paths, and a list called found_paths, which will store the actual paths taken. These paths are saved as strings showing the sequence of moves for example, “Off-Stage -> Jump -> Up-B -> Grab Ledge”.

Next, it will makes a stack with the starting node as Off-Stage" and the path so far, which at this point contains only that single vertex. The stack represent all the paths that the algorithm is currently exploring. Since this is a DFS, the stack is used to make sure that the most recent discovered paths are explored first.

The main loop begins by removing the most recent pushed from the stack. This gives us the current position in the graph and the path taken to get there. it then checks if the current vertex is one of the success states, "Grab Ledge" or "On Stage". If it is, the path is converted into a readable string and checked against all previously recorded successful paths. This is done to avoid counting paths that are just partial versions of longer ones already found—for example, both “Off-Stage -> Up-B -> Grab Ledge” and “Off-Stage -> Up-B -> Grab Ledge -> On Stage” might appear, but only the latter should be counted.

If the current vertex is not a success state, the algorithm continues exploring all the neighbors of the current vertex in the graph. For each neighbor, a new path is created by adding the neighbor to the current path, and this new pair (neighbor node and updated path) is pushed to the stack. This process repeats until the stack is empty and it would until every possible path from the starting vertex has been explored. At the end, it will return the number of unique successful paths, as well as the list of path strings.

The print strings show the amount of path each character archetypes can choose to go back on stage. Below are the ways each archetype can take to go back on stage.

Archetype	Number of Ways
1	2
2	5
3	6
4	4
5	2
6	6

The amounts of ways an archetype could take to return to stage could be use to infer the survivability of the character the player choose while playing the game. The higher amount of way an archetype could choose to return back on stage would make it harder for the enemy to predict your movement and would give you more room for error while playing your archetypes. With easier way to get back on stage it would also make your chances of winning much bigger with the higher chance of you not getting knock to the border and would increase you survivability.

VI. Conclusion

This study presents a graph-theoretic approach to modeling character survivability in Super Smash Brothers Ultimate, inspired by the dynamics of different archetyps of characters in off-stage play. Making a model of a character archetype recovery options as a series of states as a vertex in a directed graph and categorizing fighters based on key ingame mecahnics such as mid-air jumps, freefall, directional air dodge, Up special move and character with tethers, this paper try to represent how a character's survivability can be measured and be visualized.

This paper application of a Depth First Search algorithm, shows that this approach provides a clear measure of the amount of ways that can infer a character archetype recovery potential. The number of unique paths calculated directly correlates to a harder time the enemy could predicit the movement of each archetype, offering an insight into why certain recovery archetypes are safer than others.

These findings show the possibility of combining topics such as discrete mathematics, more focused on graph theory and with fighting game-specific ingame mechanical knowledge to create intelligent and interpretable models of complex gameplay mechanics. This paper can be used as a base for further understanding into character design and the evaluation of character usability in other fighting games contexts.

VII. Appendix

Github Repository for Python Program:

https://github.com/testbored/Discrete-Mathematics

VIII. Thanks And Gratitude

I would like to extend biggest thank you first and foremost to God for His unending kindness. My heartfelt thank yous also goes to my family for their support . I am very thankful for my your lectures and teaching while in the Discrete Mathematics Class (IF1220), and I wish to thank the lecturers, Mr. Rinaldi Munir and Mr. Arrival Dwi, the teaching assistants, and my fellow students for an interesting and unique semester I am confident that the knowledge gained will be beneficial for my future

References

[1] EBSCO, "Video Games: An Overview," *research-starters*. [Online]. Available: https://www.ebsco.com/research-starters/sports-and-leisure/video-games-overview. [Accessed: Jun. 19, 2025].

[2] SmashWiki, "Super Smash Bros. Ultimate," *ssbwiki.com*. [Online]. Available: https://www.ssbwiki.com/Super_Smash_Bros._Ultimate. [Accessed: Jun. 19, 2025].

[3] SmashWiki, "Recovery," *ssbwiki.com*. [Online]. Available: https://www.ssbwiki.com/Recovery. [Accessed: Jun. 19, 2025].

[4] R. Munir, "Graf (Bagian 1)," Lecture Notes for *Matematika Diskrit*, Institut Teknologi Bandung, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. [Accessed: Jun. 20, 2025].

[5] [SmashWiki, "Double jump," *ssbwiki.com*. [Online]. Available: https://www.ssbwiki.com/Double_jump. [Accessed: Jun. 20, 2025].

[6] SmashWiki, "Up special move," *ssbwiki.com*. [Online]. Available: https://www.ssbwiki.com/Up_special_move. [Accessed: Jun. 20, 2025].

[7] SmashWiki, "Helpless," *ssbwiki.com*. [Online]. Available: https://www.ssbwiki.com/Helpless. [Accessed: Jun. 20, 2025].

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Juni 2025

Mahmudia Kimdaro Amin - 13524083