International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 3, May-June 2025

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Application of Graph and Network Theory in Game Analysis

Robert Woźniak

Email: patapon456@gmail.com

ORCID: 0009-0000-9153-4195

-------------------------------------*************************-------------------------------

Abstract:

This paper uses graph theory and networks to model the behavior and strategies of a computer character in a game. The study was conducted on the character Hwoarang from the game TEKKEN 8. Based on his game information, a multigraph maps the transitions between different combat positions. The graph's vertices represent combat positions, while the directed, weighted edges correspond to the sequences of blows enabling the transitions between them. Combat positions are vertices, and the edges correspond to the sequences of blows, enabling the transitions between them. Cycles, paths, and Hamiltonian paths were found, and fundamental graph analysis was performed. The study showed that graph-based models can effectively describe and support the development of character movement and combat strategies in computer game environments.

Keywords — **Graph Theory, Game analysis, Strategy, Multigraphs.**

-------------------------------------*************************-------------------------------

I. INTRODUCTION

Researchers often use simple graphs from the theory of graphs and networks. Using simple graphs in many cases can result in the loss of important information such as the quantity of relations between vertices. An example is modelling an airline network: cities are vertices, and airline connections are edges. A simple graph will only consider the existence of connections between cities.

In reality, several daily flights may be operated by different carries at different times. Such information is important in analyzing passenger transport availability; using only single relations will result in an erroneous network analysis. In analyzing energy infrastructure using multiple relations, it is possible to assess whether there are independent transmission lines between two transformer stations. Very similar problems occur in water or gas networks, where simple graphs do not capture the differentiation of functions or transmitted media types. Another example is the problem of public transport in cities. There may be more than one route with many stops between two points. Using only one edge can lead to inaccurate water flow analyses. In the case of IT, servers can exchange services using multiple channels — one of which can be used for critical data and another for archiving purposes. These examples suggest using more advanced graph models such as weighted graphs or multigraphs. However, can multigraphs be used to analyse and develop characters in computer games?

II. LITERATURE REVIEW

Since it is impossible to find articles directly investigating the application of multigraphs in computer character analysis, one can distinguish research on multigraphs themselves.

In research Sjöstrand [18], the problem of simplifying multigraphs with loops to simple graphs is addressed. It is shown that a multigraph with a graphical degree sequence can be transformed into a simple graph through double-edge changes involving at least one loop or multiple edges. This method finds application in Monte Carlo sampling in graph theory.

A new architecture of neural networks operating on multigraphs is presented, extending the concept

 ©IJSRED: All Rights are Reserved

of convulsive signal processing with various types of edges. This leads to new models of signal processing, filtering, and frequency representation [4]. Butler, Parada-Mayorga, and Ribeiro presented an architecture for resource allocation and hate speech localization.

Expanding the methodological landscape of network modelling, Pelekis [16] introduced a new model of a random binomial multigraph, combining elements of hypergraph network theory and Erdős–Rényi random graphs. This model adds edges between vertices from a fixed family of hypervertices with a certain probability.

Gionfriddo and Ameto [7] studied the problem of strict edge colouring in multigraphs, where each infinite vertex must have at least two edges of the same colour. They introduced the concept of the upper chromatic index and analysed edge-colouring properties.

Boneva et al. [2] proposed a method for transforming multigraph systems into simple graphs, allowing the use of tools that do not natively support multigraphs. They claim applications in modelling the verification of object-oriented systems.

Dankelmann and Oellermann [6] analysed degree sequences in multigraphs with the maximum possible number of edges for a given degree of connectivity.

Magnant [12] studied Gallai structures in multigraphs, analysing their density and estimating the number of edges satisfying Gallai conditions.

Micale et al. [13] proposed a new algorithm for matching multigraph subgraphs using a unique data structure to filter matches and optimize the search order, outperforming methods like SUMgra or Memgraph.

Cranton [5] proved that every multigraph with a maximum vertex degree of at least three can be leaf-colored entirely with no more than 2Δ−12 colours.

Kandesamy, Kandesamy, and Smarandache [11] discussed applications of multigraphs in modelling multinetworks and provided a literature review focusing on computer science and social sciences.

Oziewicz and Velázquez-Quesada [15] introduced the concept of "graphs of graphs," treating multigraphs as higher-order structures — important for modelling megastructures and hierarchical systems.

Kapoor, Saxena, and van Leeuwen [10] proposed a method for finding subjective multigraph patterns based on maximum entropy tested in aerial network analyses.

Multigraphs have also been used for community analysis, where different types of edges represent various relationships between community members. Shafie [17] presented case studies of complex social structures.

Bryant et al. [3] discussed the necessary conditions for decomposing a complete multigraph into cycles, extending previous results on cyclic graph decomposition.

Kohn [8] demonstrated that the chromatic index of multigraphs asymptotically coincides with the fractional chromatic index, impacting the theory of graph colourings.

Kaplan et al. [9] proposed approximation algorithms for the asymmetric convoyeur problem in multigraph systems directed to cycles with bioinformatics and data compression applications.

Nakano, Nishizeki, and Saito [14] studied the f-colouring of multigraphs, where a function f limits the number of vertices. They presented a polynomial algorithm for estimating the f-chromatic number.

Finally, Bernstein, Kostochka, and Pron [1] extended the concept of dp-coloring as a generalization of list colouring and discussed the properties of dp-critical graphs.

III. METHODOLOGY

Definition 1. Multidigraph

A multidigraph is a triple $D = (V, A, \psi)$, where:

- V is a finite set of vertices,
- A is a finite set of arcs (directed edges),
- $\psi: A \to V \times V$ is a function assigning to each arc $a \in A$ its source and target vertices, i.e., $\psi(a) = (u,v)$.

Explanation:

Unlike simple digraphs, a multidigraph allows multiple directed arcs between the same pair of

 ©IJSRED: All Rights are Reserved

vertices. The mapping ψ defines the direction of each arc, while w assigns its weight.

Definition 2. Components of a Multidigraph

Let $D = (V, A, \psi)$ be a multidigraph. Then:

- $V(D) = V$ is the vertex set,
- $A(D) = A$ is the arc set,

For each $a \in A$, if $\psi(a) = (u, v)$, then:

- u is the source of arc a,
- v is the target of arc a.

Visualization:

The multidigraph D can be visualized by drawing points for vertices and arrows for arcs. Each arc a is represented as an arrow from u to v, according to ψ(a).

Definition 3. Walks, Paths, and Cycles in a Multidigraph

Let $D = (V, A, \psi)$ be a multidigraph. Then:

- *A* walk in D is a sequence:

$(v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k)$ for $k \geqslant 0$,

such that $\psi(a_i) = (v_{i-1}, v_i)$ for all $i = 1, \ldots, k$.

- A path is a walk in which all vertices are distinct.
- A closed walk (circuit) satisfies $v_k = v_0$.

A cycle is a closed walk in which $v_0, v_1, \ldots, v_{k-1}$ are all distinct and $k \geqslant 1$.

Explanation:

Walks may revisit both vertices and arcs. Cycles form closed structures with minimal repetition—only the starting and ending vertex are the same.

Definition *4*. In-degree and Out-degree

Let $D = (V, A, \psi)$ be a directed (multi)graph and $v \in V$. Then:

- The out-degree of v, denoted $\deg^{+}(v)$, is the number of arcs for which v is the source,
- The in-degree of v, denoted $\deg^{-}(v)$, is the number of arcs for which v is the target.

Explanation:

These concepts extend classical degree notions from undirected graphs by accounting for arc directionality.

Definition 5. Hamiltonian Structures

Let $D = (V, A)$ be a simple digraph.

- A Hamiltonian path is a path that visits each vertex in V exactly once.
- A Hamiltonian cycle is a cycle $(v_0, v_1, \ldots, v_k)$ such that each vertex appears exactly once among $v_0, \ldots, v_{k-1}$, and $v_k = v_0$.

Explanation:

Even in multidigraphs, Hamiltonian properties refer to vertex visits, not edge multiplicity—each vertex must appear once, regardless of the number of available arcs.

Definition 6. Degree Centrality

In an undirected graph, the degree centrality of a vertex v is:

$$C_D(v) = \deg(v)$$

In a directed graph:

- In-degree centrality: $\deg^{-}(v)$,
- Out-degree centrality: $\deg^{+}(v)$.

Definition 7. Betweenness Centrality

Betweenness centrality quantifies how frequently a vertex appears on the shortest paths between other pairs of vertices in the graph.

Definition 8. Clustering Coefficient

The clustering coefficient measures the local density of connections in a vertex's neighbourhood. It reflects the likelihood that the neighbours of a given vertex are also connected to each other.

The coefficient takes values in the range [0, 1], where:

- 0 indicates no local connections among neighbours,
- 1 indicates a fully connected neighbourhood.

IV. AIM OF RESEARCH

The main goal of this work is to answer whether it is possible to use network and graph theory to describe the behaviour of a computer character. Additionally, an attempt will be made to answer whether the mentioned method can be used to develop a strategy for moving this character.

 ©IJSRED: All Rights are Reserved

V. DATA

Hwoarang from the game TEKKEN 8 will be used to answer the questions posed for the study. The data frame can be found in [20]. We used information about commands and notes. If there is no extra information about changing to another battle stance in the notes, we treat that move as not switching to a battle stance.

Special cases of organising movement

In the article [19], the author presented diagrams of applied kumite techniques, in which he organized kumite techniques in karate as a diagram. Additionally, one can quote his opinion on predicting the opponent's move: "Most participants in kumite tournaments are experts on no more than a handful of strategies, which are - of course - used intelligently. Therefore, anticipating how a future opponent will fight is by no means impossible." Following the cited article, can his idea be used to apply network and graph theory to a programmed character to imitate a real martial art? Can we build a graph in which the vertices are battle positions and the edges are blows changing these battle positions?

Character introduction and analysis

Hwoarang is a fictional character in the game TEKKEN 8. The fighting style he uses is based on absolute Taekwondo - a style based on dynamic kicks and frequent changes of fighting positions. He uses five basic stances: *Left Stance*, *Right Stance*, *Left Flamingo*, *Right Flamingo*, and *Back turn Stance*. Each of these positions defines available techniques and changes in battle positions.

Fig 1. Hwoarang from TEKKEN 8. All rights belongs to Bandai Namco

The tested character has five combat stances*: Left Stance*, *Right Stance*, *Left Flamingo*, *Right Flamingo*, and *Back turn Stance*.

Fig 2. Hwoarang – all stances from left*: Left Stance, Right Flamingo, Back turn Stance, Right Stance, Left Flamingo*. All rights belongs to Bandai Namco

In order to verify whether it is possible to describe a computer character using network and graph theory, it is necessary to explain the notation presented by the developer.

Fig 3. Examples moves with notation.. All rights belongs to Bandai Namco

See TEKKEN notation in appendix.

Using information from the game and this game is available on websites like [20], it can construct a graph. See Appendix for created graph.

The graph created presents the structure of transitions between battle positions for the character Hwoarang from the game TEKKEN 8. The graph's vertices are the leading combat positions, such as *Left Stance*, *Right Stance*, *Left Flamingo*, *Right Flamingo*, and *Back turn Stance*. The edges connecting the vertices correspond to possible transitions between moves within one position or between different positions. Each edge has its own name, which means a combination of buttons (e.g., *DF+2, 1+4, 3 4*) to perform a move in the character. It should be emphasized that this is a directed graph, which means that each transition has a specific direction from one position to the next move. The graph also shows loops (self-loops), which mean performing subsequent techniques without changing the combat position.

Short visualization for directed edge between vertexes.

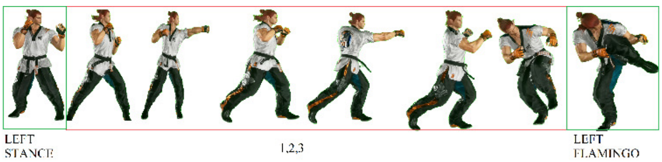

Fig 5. Representation *1,2,3* edge connecting *Left Stance* and *Left Flamingo*

Fig 6. Representation *1+2* edge connecting *Left Stance* and *Back turn Stance*

The attached image shows Hwoarang going from *Left Stance* to *Left Flamingo*. This transition consists of a series of three punches labelled *1, 2, 3*, where *1 Left Punch, Right Punch*, and *3* is the transition to *Left Flamingo*. In this graph, sequences *1, 2,* and *3* are the edges between the nodes' *Left Stance* and *Left Flamingo*. Executing this combination of moves changes the stance from one stance to another. Each execution of the sequence of moves is represented in the graph as an edge connecting the corresponding nodes describing the character's states. Similarly, there can be presented transitions such as *Left Stance 1+2*, or *Right Flamingo 1*.

Fig 6. Representation *1* edge connecting *Right Flamingo* and *Left Stance*.

Due to the presented graph having many directed edges and autoloops, for analysing this graph, there can be calculated next steps such as In, Out, and Total Degree, number of autoloops for every vertex, and statistics such as Clustering Coefficient, Closeness Centrality, and Betweenness Centrality.

The table 2 presented above shows the out-degree and in-degree statistics for the vertices appearing in the graph. In both cases, the vertex under the largest number of edges is *Left Stance*, and in terms of the smallest number of edges, it is *Back turn Stance*. Analysing the table in terms of Total-degree, the dominant vertex is *Left Stance* – it contains the largest number of blows and changes to this battle position. In the same context, *Back turn Stance* has the smallest number of blows entering and leaving this battle position. Therefore, we can say *that Back turn Stance* is very situational in terms of attacks.

Table 3 shows information about the number of auto edges for each vertex in the created directed multigraph. The *Left Stance* vertex has the most auto edges, at 50, and the *Right Flamingo* has the least. The *Back turn Stance* does not have any auto edges at all. Regarding the execution of blows by the fictional character, we can say that the *Left Stance* has the most possibility of dealing blows to stay in this combat position, and the *Back turned Stance* is a transitional combat position - it is not possible to stay in this combat position. The above conclusions are consistent with the information obtained from the calculations. Measures for the created graph are in the following table.

 ©IJSRED: All Rights are Reserved

In Table 4, For the clustering coefficient, we interpret it as an indicator of the local connectivity of the neighbour*s* of the vertex. A value of one means all neighbouring vertices are connected, while zero means no connections between neighbour*s*. Closeness centrality is interpreted as the proximity of the vertex to other vertices - the higher the value, the more dominant the vertex is in the network. Betweenness centrality is interpreted as the importance of a vertex in mediating shortest paths - a value of 0 means no mediation, while a non-zero value indicates that the vertex plays an important role in connecting paths.

There can be found paths, cycles, and Hamiltonian cycles from the built graph. For example, see the appendix for a list of paths, cycles, and Hamiltonian cycles.

Example interpretation for a path:

(*LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*, ”*4,F*”, *RF*, ”*F+3*”, *LF*)

The path shown above starts at the *Left Stance* position and then passes through vertices such as *Back turn Stance*, *Right Stance*, *Right Flamingo*, and ends at *Left Flamingo*. It passes through edges such as: ”*1+2*”, ”*4*”, ”*4,F,*” and ”*F+3*”. The remaining example paths are interpreted analogously.

Example interpretation for a cycle:

(*LS*, ”*4,4,2*”, *RS*, ”*uf+2,3/F*”, *LF*, ”*3,4*”, *RF*, ”*1*”, *LS*)

The above path is a cycle because it starts and ends at the *Left Stance*. It passes through the following vertices: *Right Stance*, *Left Flamingo*, and *Right Flamingo*, and edges such as: ”*4,4,2*”, ”*UF+2,3/F,*” and ”*1*”. The remaining example cycles are interpreted analogously.

Example interpretation for a Hamiltonian cycle:

(*LS*, ”*1,2,3*”, *LF*, ”*B+3*”, *RF*, ”*D+4*”, *RS*, ”*F+3,B*”, *BTS*, ”*3*”, *LS*)

The above path is a Hamiltonian cycle because it starts at the *Left Stance* and ends at that vertex. It goes through all possible vertices and edges such as: ”*1,2,3*”, ”*B+3*”, ”*D+4*”, ”*F+3,B*”, and ”*3*”.

Using the built multigraph directed at the Hwoarang character and identifying paths, cycles and Hamilton cycles and graph statistics, there are the foundations for creating three combat strategies: offensive, defensive and surprising. The offensive strategy will consist of quickly moving between positions using short or medium paths. Due to the high out degree (92) and closeness centrality equal to 1.0, *Left Stance* will play an important role. This means that from this *Left Stance* you can move to other combat positions. Example offensive paths, such as (*LS*, "*1+2*", *BTS*) or (*LS*, "*3+4*", *RS*), will allow for quick short attacks of add-ons, putting pressure on the opponent. Longer paths, such as (*LS*, "*1+2*", *BTS*, "*4*", *RS*, "*4,F*", *RF*, "*F+3*", *LF*), allow for dynamic transitions between combat positions. This is to confuse the opponent and cause our attack to be misread. With the help of *Right Flamingo* and *Right Stance*, respectively out-degree 38 and 20 and high betweenness centrality, they will support the creation of long combinations of blows.

The goal of the defensive strategy will be to control the position and rhythm of the fight through repeatable, safe transitions - the use of cycles. The *Left Stance* combat position is a*ls*o distinguished by a very large number of autoloops equal to 50. It will be the central defensive position in the battle. An example defensive cycle (*LS*, "*1,2,3*", *LF*, "*3,4*", *RF*, "*1*", *LS*) will allow you to stay active without having to change positions. Cycles can also be easily interrupted and switched to counter-offensive thanks to the high number of edges coming out of *Left Stance*.

Using Hamilton cycles, an example cycle (*LS*, "*1,2,3*", *LF*, "*B+3*", *RF*, "*D+4*", *RS*, "*F+3,B*", *BTS*, "*3*", *LS*) will allow you to move through all the key positions. This will confuse the opponent by constantly changing the direction and rhythm of the fight. With a high clustering coefficient (1.0) for positions like *LF*, *RF* and *BTS*, it will be difficult for the opponent to predict the actual direction of movement.

 ©IJSRED: All Rights are Reserved

VI. SUMMARY

Based on the analysis of the Hwoarang character's programming, it can be safely stated that network and graph theory can be used to describe the character's behaviour. The representation of the character in the form of a directed multigraph will allow for mapping the transitions between different combat positions. Additionally, cycle paths and Hamilton cycles were identified. On their basis, an offensive, defensive, and surprising opponent strategy was built using, among others, the degrees of vertex entry and exit, the number of loops, and centrality measures. This confirms that the goal of the work has been achieved and that graph methods can be used to describe the character's movement in the computer game environment.

ACKNOWLEDGE

The Author declare that there is no conflict of interest.

APPENDIX

TABLE 1 NOTATION FOR MOVE IN TEKKEN 8.

Notation	Sign
1	Left Punch
2	Right Punch
3	Left Kick
4	Right Kick
U	Up
D	Down
B	Back
F	Forward

Notation sign:

1 - Left Punch
2 - Right Punch
3 - Left Kick
4 - Right Kick
U - Up
D - Down
B - Back
F - Forward

- pressing *1* – perform Left Punch;
- pressing *2* – perform Right Punch;
- pressing *1* and *2* simultaneously is *1+2* – move with both hands;
- pressing *3* and *4* simultaneously is *3+4* – move with both legs;
- pressing *D* and *F* and *2* and *3* simultaneously is written *DF 2+3* – Perform a move

Full multidigraph model:

V = {*Back turn Stance*, *Left Flamingo*, *Left Stance*, *Right Flamingo, Right Stance*}

A multidigraph is a triple $D = (V, A, \psi)$, where:

- ψ is a finite set of vertices,
- A is a finite set of arcs (directed edges),
- $\psi: A \rightarrow V \times V$ is a function assigning to each arc $a \in A$ its source and target vertices, i.e., $\psi(a) = (u,v)$.

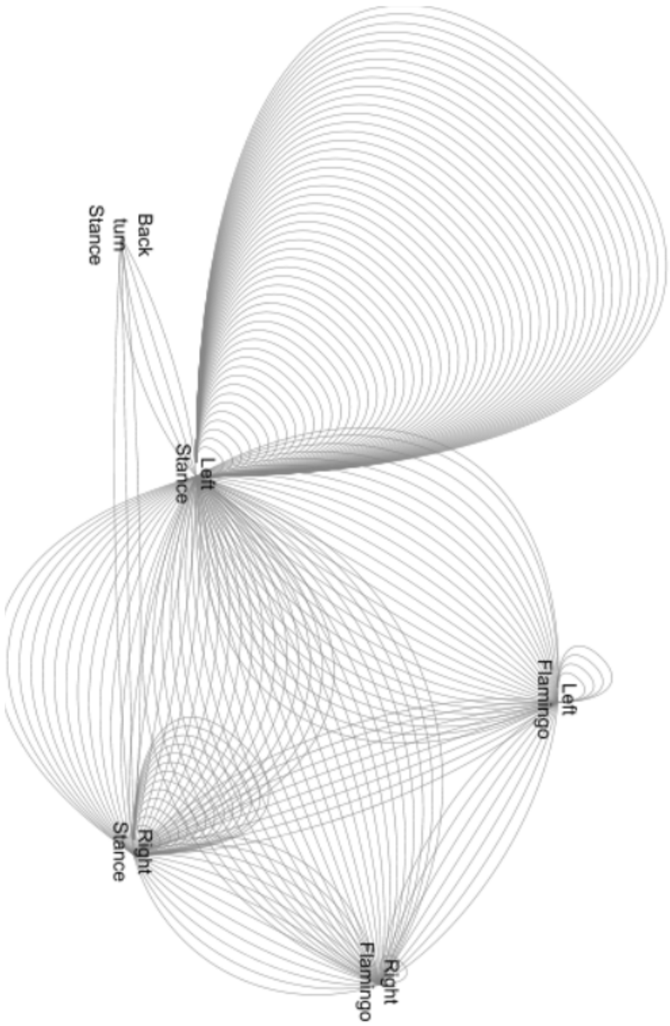

Fig *4*. Graph for Hwoarang Move set

 ©IJSRED: All Rights are Reserved

A = {*1, 1+2, 1+2+3+4, 1+4, 1.1, 1.1.3, 1.1.3.3, 1.2, 1.2.3, 1.2.4, 1.2.F+3, 1.2.F+4, 2, 2.1, 2.1.1, 2.3, 2.4, 2.4.3, 2.4.4, 2.4.F, 2.F+3, 2.F+4, 3, 3+4, 3.2, 3.3, 3.3.2, 3.3.4, 3.3.F+4, 3.4, 3.4F, 3.F+4, 3~4, 4, 4.3, 4.3 F, 4.4, 4.4.3, 4.4.4, 4.4.4 F, 4.4.4.4, 4.F, 4~3, SS.3, SS.3.3, SS.4, B+1, B+1+3/B+2+4, B+2, B+3, B+4, B+4.3, D+3, D+3+4, D+3.3, D+3.4, D+4, D+4.3, D+4.3.4, D+4.4, DB+3, DB+3+4, DB+4, DB+4.4, DF+1, DF+1.3, DF+2, DF+2+3, DF+2+3.4, DF+2+3.F/U/D, DF+3, DF+3+4, DF+3.4, DF+4, DF+4.F, F+1 ,F+1+2, F+1.2, F+2, F+3, F+3.B, F+3~3, F+4, F+4.4, F+4~4, F.F+3, F.F+4, F.F.F+3, F.F.F+4, f.N.4, F.N.D.DF, F.N.DF:4, U+3, U+3+4, U+4, UB+2, UB+4, UF+2, UF+3, UF+3+4, UF+3+4.4, UF+3+4.F, UF+3.4, UF+3.4.3, UF+4, UF+4.4, UF+4.4.4, UF.N.4, WS1, WS2, WS2.3, WS3, WS3+4, WS3+4.3, WS4, WS4.4*}.

List of arcs:

ψ (a) = (u, v) for all $a \in A$:

ψ(1+2) = (Left Stance, Back turn Stance)
ψ(WS3+4) = (Left Stance, Back turn Stance)
ψ(D+3) = (Right Stance, Back turn Stance)
ψ(F+3.B) = (Right Stance, Back turn Stance)
ψ(1.2.3) = (Left Stance, Left Flamingo)
ψ(2.3) = (Left Stance, Left Flamingo)
ψ(4.3) = (Left Stance, Left Flamingo)
ψ(4.3 F) = (Left Stance, Left Flamingo)
ψ(F+3) = (Left Stance, Left Flamingo)
ψ(DF+1.3) = (Left Stance, Left Flamingo)
ψ(DF+3) = (Left Stance, Left Flamingo)
ψ(DB+3+4) = (Left Stance, Left Flamingo)
ψ(U+3) = (Left Stance, Left Flamingo)
ψ(UF+3+4.F) = (Left Stance, Left Flamingo)
ψ(WS2.3) = (Left Stance, Left Flamingo)
ψ(SS.3.3) = (Left Stance, Left Flamingo)
ψ(3) = (Left Stance, Left Flamingo)
ψ(3) = (Left Flamingo, Left Flamingo)
ψ(DF+3) = (Left Flamingo, Left Flamingo)
ψ(F+3) = (Left Flamingo, Left Flamingo)
ψ(UF+3+4) = (Left Flamingo, Left Flamingo)
ψ(DF+2+3.F/U/D) = (Right Stance, Left Flamingo)
ψ(F+3) = (Right Stance, Left Flamingo)
ψ(3) = (Right Flamingo, Left Flamingo)
ψ(3+4) = (Right Flamingo, Left Flamingo)
ψ(1) = (Left Stance, Left Stance)
ψ(1.1) = (Left Stance, Left Stance)
ψ(1.1.3) = (Left Stance, Left Stance)
ψ(1.1.3.3) = (Left Stance, Left Stance)
ψ(1.2) = (Left Stance, Left Stance)
ψ(1.2.F+3) = (Left Stance, Left Stance)
ψ(1.2.F+4) = (Left Stance, Left Stance)
ψ(2) = (Left Stance, Left Stance)
ψ(2.F+3) = (Left Stance, Left Stance)
ψ(2.F+4) = (Left Stance, Left Stance)
ψ(3) = (Left Stance, Left Stance)
ψ(3.3) = (Left Stance, Left Stance)
ψ(3.3.2) = (Left Stance, Left Stance)
ψ(3.3.F+4) = (Left Stance, Left Stance)
ψ(3~4) = (Left Stance, Left Stance)
ψ(4) = (Left Stance, Left Stance)
ψ(4.4.4.4) = (Left Stance, Left Stance)
ψ(F+3~3) = (Left Stance, Left Stance)
ψ(F+1+2) = (Left Stance, Left Stance)
ψ(DF+1) = (Left Stance, Left Stance)
ψ(DF+2) = (Left Stance, Left Stance)
ψ(DF+3.4) = (Left Stance, Left Stance)
ψ(DF+3+4) = (Left Stance, Left Stance)
ψ(D+3) = (Left Stance, Left Stance)
ψ(d+4) = (Left Stance, Left Stance)
ψ(d+4.4) = (Left Stance, Left Stance)
ψ(db+3) = (Left Stance, Left Stance)
ψ(db+4.4) = (Left Stance, Left Stance)
ψ(b+1) = (Left Stance, Left Stance)
ψ(b+1+3/b+2+4) = (Left Stance, Left Stance)
ψ(b+2) = (Left Stance, Left Stance)
ψ(b+3) = (Left Stance, Left Stance)
ψ(ub+2) = (Left Stance, Left Stance)
ψ(ub+4) = (Left Stance, Left Stance)
ψ(u+3+4) = (Left Stance, Left Stance)
ψ(uf+2) = (Left Stance, Left Stance)
ψ(uf+3) = (Left Stance, Left Stance)
ψ(uf+3.4.3) = (Left Stance, Left Stance)
ψ(1+2+3+4) = (Left Stance, Left Stance)
ψ(f.n.d.df) = (Left Stance, Left Stance)
ψ(f.n.df:4) = (Left Stance, Left Stance)
ψ(uf.n.4) = (Left Stance, Left Stance)
ψ(f.f.F+3) = (Left Stance, Left Stance)

 ©IJSRED: All Rights are Reserved

ψ(ws1) = (Left Stance, Left Stance)
ψ(ws2) = (Left Stance, Left Stance)
ψ(ws3) = (Left Stance, Left Stance)
ψ(ws4) = (Left Stance, Left Stance)
ψ(WS3+4.3) = (Left Stance, Left Stance)
ψ(SS.3) = (Left Stance, Left Stance)
ψ(3) = (Back turn Stance, Left Stance)
ψ(D+3) = (Back turn Stance, Left Stance)
ψ(1) = (Left Flamingo, Left Stance)
ψ(3.2) = (Left Flamingo, Left Stance)
ψ(3.3) = (Left Flamingo, Left Stance)
ψ(3.f+4) = (Left Flamingo, Left Stance)
ψ(4~3) = (Left Flamingo, Left Stance)
ψ(1+4) = (Left Flamingo, Left Stance)
ψ(b+1) = (Left Flamingo, Left Stance)
ψ(df+3.4) = (Left Flamingo, Left Stance)
ψ(D+3) = (Left Flamingo, Left Stance)
ψ(f+4) = (Left Flamingo, Left Stance)
ψ(uf+3) = (Left Flamingo, Left Stance)
ψ(uf+3.4.3) = (Left Flamingo, Left Stance)
ψ(uf+4) = (Left Flamingo, Left Stance)
ψ(1.1) = (Right Stance, Left Stance)
ψ(2.1.1) = (Right Stance, Left Stance)
ψ(2.3) = (Right Stance, Left Stance)
ψ(2.4.4) = (Right Stance, Left Stance)
ψ(3) = (Right Stance, Left Stance)
ψ(3~4) = (Right Stance, Left Stance)
ψ(4.4) = (Right Stance, Left Stance)
ψ(3+4) = (Right Stance, Left Stance)
ψ(B+3) = (Right Stance, Left Stance)
ψ(B+4) = (Right Stance, Left Stance)
ψ(DF+2+3) = (Right Stance, Left Stance)
ψ(DF+2+3.4) = (Right Stance, Left Stance)
ψ(F.F+3) = (Right Stance, Left Stance)
ψ(F+1) = (Right Stance, Left Stance)
ψ(F+1.2) = (Right Stance, Left Stance)
ψ(1) = (Right Flamingo, Left Stance)
ψ(3~4) = (Right Flamingo, Left Stance)
ψ(b+3) = (Right Flamingo, Left Stance)
ψ(b+4) = (Right Flamingo, Left Stance)
ψ(df+4) = (Right Flamingo, Left Stance)
ψ(D+3) = (Right Flamingo, Left Stance)
ψ(D+3.3) = (Right Flamingo, Left Stance)
ψ(d+4.3) = (Right Flamingo, Left Stance)
ψ(d+4.3.4) = (Right Flamingo, Left Stance)
ψ(f+3) = (Right Flamingo, Left Stance)
ψ(uf+3) = (Right Flamingo, Left Stance)
ψ(1.2.4) = (Left Stance, Right Flamingo)
ψ(2.4) = (Left Stance, Right Flamingo)
ψ(3.3.4) = (Left Stance, Right Flamingo)
ψ(4.4) = (Left Stance, Right Flamingo)
ψ(4.4.4 F) = (Left Stance, Right Flamingo)
ψ(f+4) = (Left Stance, Right Flamingo)
ψ(D+3.4) = (Left Stance, Right Flamingo)
ψ(U+4) = (Left Stance, Right Flamingo)
ψ(UF+3.4) = (Left Stance, Right Flamingo)
ψ(UF+4) = (Left Stance, Right Flamingo)
ψ(UF+4.4) = (Left Stance, Right Flamingo)
ψ(UF+4.4.4) = (Left Stance, Right Flamingo)
ψ(F.N.4) = (Left Stance, Right Flamingo)
ψ(3.4) = (Left Flamingo, Right Flamingo)
ψ(4) = (Left Flamingo, Right Flamingo)
ψ(3+4) = (Left Flamingo, Right Flamingo)
ψ(D+3.4) = (Left Flamingo, Right Flamingo)
ψ(UF+3.4) = (Left Flamingo, Right Flamingo)
ψ(2.4.F) = (Right Stance, Right Flamingo)
ψ(3.4F) = (Right Stance, Right Flamingo)
ψ(4.F) = (Right Stance, Right Flamingo)
ψ(DF+4.F) = (Right Stance, Right Flamingo)
ψ(D+3.4) = (Right Stance, Right Flamingo)
ψ(F+4) = (Right Stance, Right Flamingo)
ψ(F+4) = (Right Flamingo, Right Flamingo)
ψ(4.4.3) = (Left Stance, Right Stance)
ψ(4.4.4) = (Left Stance, Right Stance)
ψ(3+4) = (Left Stance, Right Stance)
ψ(f+2) = (Left Stance, Right Stance)
ψ(DF+4) = (Left Stance, Right Stance)
ψ(DB+4) = (Left Stance, Right Stance)
ψ(B+4) = (Left Stance, Right Stance)
ψ(UF+3+4) = (Left Stance, Right Stance)
ψ(UF+3+4.4) = (Left Stance, Right Stance)
ψ(F.F+3) = (Left Stance, Right Stance)
ψ(F.F+4) = (Left Stance, Right Stance)
ψ(F.F.F+4) = (Left Stance, Right Stance)
ψ(WS4.4) = (Left Stance, Right Stance)
ψ(SS.4) = (Left Stance, Right Stance)
ψ(4) = (Back turn Stance, Right Stance)
ψ(2) = (Left Flamingo, Right Stance)
ψ(B+3) = (Left Flamingo, Right Stance)
ψ(B+4) = (Left Flamingo, Right Stance)

 ©IJSRED: All Rights are Reserved

ψ(B+4.3) = (Left Flamingo, Right Stance)
ψ(D+4) = (Left Flamingo, Right Stance)
ψ(UF+3+4.4) = (Left Flamingo, Right Stance)
ψ(1) = (Right Stance, Right Stance)
ψ(2) = (Right Stance, Right Stance)
ψ(2.1) = (Right Stance, Right Stance)
ψ(2.4) = (Right Stance, Right Stance)
ψ(2.4.3) = (Right Stance, Right Stance)
ψ(3.4) = (Right Stance, Right Stance)
ψ(4) = (Right Stance, Right Stance)
ψ(4.3) = (Right Stance, Right Stance)
ψ(B+2) = (Right Stance, Right Stance)
ψ(DF+3) = (Right Stance, Right Stance)
ψ(DF+4) = (Right Stance, Right Stance)
ψ(D+3+4) = (Right Stance, Right Stance)
ψ(F+4~4) = (Right Stance, Right Stance)
ψ(2) = (Right Flamingo, Right Stance)
ψ(4) = (Right Flamingo, Right Stance)
ψ(D+4) = (Right Flamingo, Right Stance)
ψ(F+4.4) = (Right Flamingo, Right Stance)
ψ(UF+4) = (Right Flamingo, Right Stance)
ψ(U+4) = (Right Flamingo, Right Stance)

List of example paths, cycles and Hamiltonian cycles

To list paths, cycles and Hamiltonian cycles the following abbreviations for vertex names have been adopted:

- *LS* — *Left Stance*;
- *BTS* — *Back turn Stance*;
- *RS* — *Right Stance*;
- *RF* — *Right Flamingo*;
- *LF* — *Left Flamingo*.

Example Paths:

- (*LS*, ”*1+2*”, *BTS*)
- (*LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*)
- (*LS*, ”*3+4*”, *RS*)
- (*LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*, ”*4*, F”, *RF*, ”F+3”, *LF*)
- (*LS*, ”*3+4*”, *RS*, ”*4*, F”, *RF*, ”F+3”, *LF*)
- (*LS*, ”*4.3*”, *LF*)
- (*LS*, ”*4,4*”, *RF*, ”F+3”, *LF*)
- (*LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*, ”*4*, F”, *RF*)
- (*LS*, ”*3+4*”, *RS*, ”*4*, F”, *RF*)
- (*LS*, ”*4,4*”, *RF*)
- (*BTS*, ”*D+3*”, *LS*)
- (*BTS*, ”*4*”, *RS*, ”*3+4*”, *LS*)
- (*BTS*, ”*4*”, *RS*, ”*4*, F”, *RF*, ”F+3”, *LF*, ”*D+3*”, *LS*)
- (*BTS*, ”*D+3*”, *LS*, ”*3+4*”, *RS*)
- (*BTS*, ”*4*”, *RS*)
- (*BTS*, ”*D+3*”, *LS*, ”*3+4*”, *RS*, ”*4,F*”, *RF*, ”*f+3*”, *LF*)

Example Cycles

- (*LS*, ”*1,2,3*”, *LF*, ”3,*4*”, *RF*, ”1”, *LS*)
- (*LS*, ”*4,4,2*”, *RS*, ”uf+2,3/F”, *LF*, ”3,*4*”, *RF*, ”*1*”, *LS*)
- (*LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*, ”1,1”, *LS*)
- (*LS*, ”*4*,3F”, *LF*, ”B+3”, *RS*, ”2,1,1”, *LS*)
- (*RS*, ”2,4F”, *RF*, ”*D+4*”, *RS*)
- (*RS*, ”*1,1*”, *LS*, ”*1,2,F+3*”, *RS*)
- (*LF*, ”*4*,3”, *LS*, ”1,2,*4*”, *RF*, ”3”, *LF*)
- (*RF*, ”*1,1*”, *LS*, ”*3,3,3,4*”, *RF*)
- (*RF*, ”1”, *LS*, ”2,3”, *LF*, ”3,*4*”, *RF*)
- (*BTS*, ”*3*”, *LS*, ”DF+*4*”, *RS*, ”*F+3,B*”, *BTS*)

Example Hamiltonian Cycles

- (*LS*, ”1,2,3”, *LF*, ”*B+3*”, *RF*, ”D+*4*”, *RS*, ”F+3,B”, *BTS*, ”3”, *LS*)
- (*LS*, ”*F+4*”, *RF*, ”*3+4*”, *LF*, ”d+*4*”, *RS*, ”*D+3*”, *BTS*, ”3”, *LS*)
- (*LS*, ”WS+3,*4*”, *BTS*, ”*4*”, *RS*, ”2,4F”, *RF*, ”3”, *LF*, ”1”, *LS*)
- (*RF*, ”3”, *LF*, ”2”, *RS*, ”*D+3*”, *BTS*, ”3”, *LS*, ”*4,4*”, *RF*)
- (*LF*, ”*D+3,4*”, *RF*, ”d+4,3”, *LS*, ”*1+2*”, *BTS*, ”*4*”, *RS*, ”f+3”, *LF*)
- (*BTS*, ”*4*”, *RS*, ”*4*,F”, *RF*, ”*3+4*”, *LF*, ”*F+4*”, *LS*, ”*1+2*”, *BTS*)

 ©IJSRED: All Rights are Reserved

TABLE 2. STATISTIC FOR CREATED MULTIGRAPH

Statistic/ Node	***Left Stance***	**Back turn Stance**	**Left Flamingo**	**Right Stance**	***Right Flamingo***
In-degree	91	*4*	21	*40*	25
Out-degree	92	3	28	38	20
Total-degree	183	7	*49*	78	*45*

TABLE 3. NUMBER OF AUTOLOOPS FOR EACH VERTEX

Name of Vertex	**Numbers of loops for vertex**
Left Flamingo	*4*
Left Stance	50
Right Flamingo	1
Right Stance	13
Backturn Stance	0

TABLE *4*. CENTRALITY MEASURES FOR EACH VERTEX.

Vertex/Stastitic	**Clustering Coefficient**	**Closeness Coefficient**	**Betweenness Centrality**
Left Stance	0.6	1.0	0.*4*
Backturn Stance	1.0	0.6	0.0
Right Stance	0.6	1.0	0.*4*
Left Flamingo	1.0	0.8	0.0
Right Flamingo	1.0	0.8	0.0

REFERENCES

[1] Bernshteyn, A. Yu., Kostochka, A. V., & Pron, S. P. (2017). On DP-coloring of graphs and multigraphs. Siberian Mathematical Journal, 58(1), 28–36. https://doi.org/10.1134/S0037446617010049

[2] Boneva, I., Hermann, F., Kastenberg, H., & Rensink, A. (2007). Simulating multigraph transformations using simple graphs. Electronic Communications of the EASST, 6. https://doi.org/10.14279/tuj.eceasst.6.62

[3] Bryant, D., HoRSley, D., Maenhaut, B., & Smith, B. R. (2010). Cycle decompositions of complete multigraphs. Journal of Combinatorial Designs, 19(1), 42–69. https://doi.org/10.1002/jcd.20263

[4] Butler, L., Parada-Mayorga, A., & Ribeiro, A. (2023). Convolutional learning on multigraphs. IEEE Transactions on Signal Processing (submitted). arXiv:2209.11354v2. https://arxiv.org/abs/2209.11354

[5] Cranston, D. W. (2009). Multigraphs with Δ ≥ 3 are Totally-(2Δ−1)-Choosable. Graphs and Combinatorics, 25(3), 345–353. https://doi.org/10.1007/s00373-008-0817-5

[6] Dankelmann, P., & Oellermann, O. R. (2005). Degree sequences of optimally edge-connected multigraphs. ARS Combinatoria, 75, 149–160.

[7] Gionfriddo, M., & Amato, A. (2007). An edge colouring of multigraphs. Computer Science Journal of Moldova, 15(2), 230–239. https://www.researchgate.net/publication/220491939

[8] Kahn, J. (1996). Asymptotics of the chromatic index for multigraphs. Journal of Combinatorial Theory, Series B, 68(2), 233–254. https://doi.org/10.1006/jctb.1996.0067

[9] Kaplan, H., Lewenstein, M., Shafrir, N., & Sviridenko, M. (2005). Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. Journal of the ACM, 52(4), 602–626. https://doi.org/10.1145/1070432.1070436

[10] Kapoor, S., Saxena, D. K., & van Leeuwen, M. (2020). Discovering subjectively interesting multigraph patterns. Machine Learning, 109, 1669–1696. https://doi.org/10.1007/s10994-020-05873-9

[11] Kandasamy, W. B. V., Kandasamy, I., & Smarandache, F. (2019). Multigraphs for multi networks. ResearchGate. https://www.researchgate.net/publication/333853530

[12] Magnant, C. (2015). Density of Gallai multigraphs. The Electronic Journal of Combinatorics, 22(1), P1.32. https://doi.org/10.37236/4615

[13] Micale, G., Di Maria, A., Grasso, R., Bonnici, V., Ferro, A., Shasha, D., Giugno, R., & Pulvirenti, A. (2018). MultiGraphMatch: A subgraph matching algorithm for multigraphs. Journal of the ACM, 37(4), Article 111. (DOI pending)

[14] Nakano, S., Nishizeki, T., & Saito, N. (1988). On the f-coloring of multigraphs. IEEE Transactions on Circuits and Systems, 35(3), 345–351. https://doi.org/10.1109/31.2029

[15] Oziewicz, Z., & Velázquez-Quesada, F. R. (2009). Multigraph of multigraphs. Gráficas de Gráficas: Introducción a teoría de categorías. https://www.researchgate.net/publication/254915659

[16] Pelekis, C. (2023). A binomial random multigraph. Preprint. arXiv:2401.00543. https://arxiv.org/abs/2401.00543

[17] Shafie, T. (2015). A multigraph approach to social network analysis. Journal of Social Structure, 16(1). https://doi.org/10.21307/joss-2019-011

[18] Sjöstrand, J. (2021). Making multigraphs simple by a sequence of double edge swaps. Discrete Mathematics, 344, 112328. https://doi.org/10.1016/j.disc.2021.112328

[19] Paz-y-Mino-C, G. (2000). Predicting Kumite strategies: A quantitative approach to Karate. New England Science Public. https://www.researchgate.net/publication/269519279

[20] https://tekkendocs.com/t8/hwoarang

.

 ©IJSRED: All Rights are Reserved