Heterogeneous Graph Dataset with Feature Set Intersection through Game Provenance

Sidney Araujo Melo, Esteban Clua, Aline Paes
{{sidneymelo@id},{esteban,alinepaes}@ic}.uff.br
Institute of Computing, Universidade Federal Fluminense
Brazil

ABSTRACT

Provenance graphs have been adapted for digital games and game analytics and proved to be a powerful tool for capturing game session data for complex games. Due to the own game nature, which is composed by large amount and variety of data, game provenance graphs are highly heterogeneous in terms of node and edge types and their associated feature sets. Furthermore, game provenance graphs are rich from intersections across feature sets from distinct node types. However, most existing heterogeneous graph neural network solutions rely on simple approaches to deal with varying node types, such as projecting each type of node to the same n-dimensional space. They assume that node types, and consequently their composing features, to be independently distributed. In this work, we present the Smoke Squadron Dataset, a game provenance graph dataset containing game session graphs whose node types share several common feature subsets. To address this challenging heterogeneity, we propose a feature set based solution that allows projecting distinct node types that leverage feature set intersection.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; **Neural networks**; • **Applied computing** → Computer games.

KEYWORDS

graph representation learning, graph neural networks, heterogeneous graphs

ACM Reference Format:
Sidney Araujo Melo, Esteban Clua, Aline Paes. 2021. Heterogeneous Graph Dataset with Feature Set Intersection through Game Provenance. In *Workshop on Graph Learning Benchmarks @ theWebConf, 2021, Slovenia.* ACM, New York, NY, USA, 6 pages. https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

Provenance, which stands for the documented history of an object's life cycle[13], has been adapted for digital games and game analytics. Based on the Open Provenance Model, where provenance is represented by annotated graphs, the Provenance in Games (PinG) framework enables game developers to record and track game session data as game provenance graphs[9]. Since then, game provenance graphs have been used for several Game Analytics tasks[10][8][7] that may also leverage AI and ML methods [2][11], including Graph Representation Learning[15][14].

Digital games tend to generate large amounts of heterogeneous data. This is due to characteristics such as the number of game objects (characters, enemies, items, weapons, and many other), events and their respective *features*. Figure 1 presents an example of a provenance graph from a gameplay session of a prototype racing game. Each node represents the state of a player's avatar (a car in the case of a racing game) and contains multiple features associated with the current state. These features may encompass attributes such as position of objects in a 2D or 3D game space, current health points, amount of damage dealt, movement speed, among many others. Additionally, some of these features might be common among all objects (such as a 2D or 3D position) while others might be specific to a group (characters and enemies have health points but items does not) or individual objects (a character with an unique skill). As game provenance graphs are able to capture and track all objects as heterogeneous nodes and their associated features, it is possible to observe distinct levels of intersections across node types.

In Graph Representation Learning, Heterogeneous Graph Neural Networks (HGNN) have been developed to deal with graphs with distinct types of nodes and edges. However, to the best of our knowledge, they usually rely on learning feature transformation matrices, called here as type-specific projection matrices, as they map each node type to a single d-dimensional space R^D[18][17][1][3]. We argue that this approach overlooks any relationship among features of multiple node types, as each projection matrix transforms input features of each type independently. This issue might be aggravated when feature sets are not disjoint, i.e., feature sets from different types of nodes intersect, since each projection matrix might project similar feature subsets differently. Such feature set relationships across node and edge types may occur more frequently as graphs become the standard way of modeling more complex and rich domains, bringing together new challenges.

To that end, we present the Smoke Squadron Dataset[1], a game provenance graph dataset containing several game sessions of Smoke Squadron, a local multiplayer airplane battle arena game. The dataset portrays five node types with intersecting feature sets, as previously mentioned. We also present preliminary results on

GLB, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

[1] https://github.com/sidneyaraujomelo/smokesquadrondataset

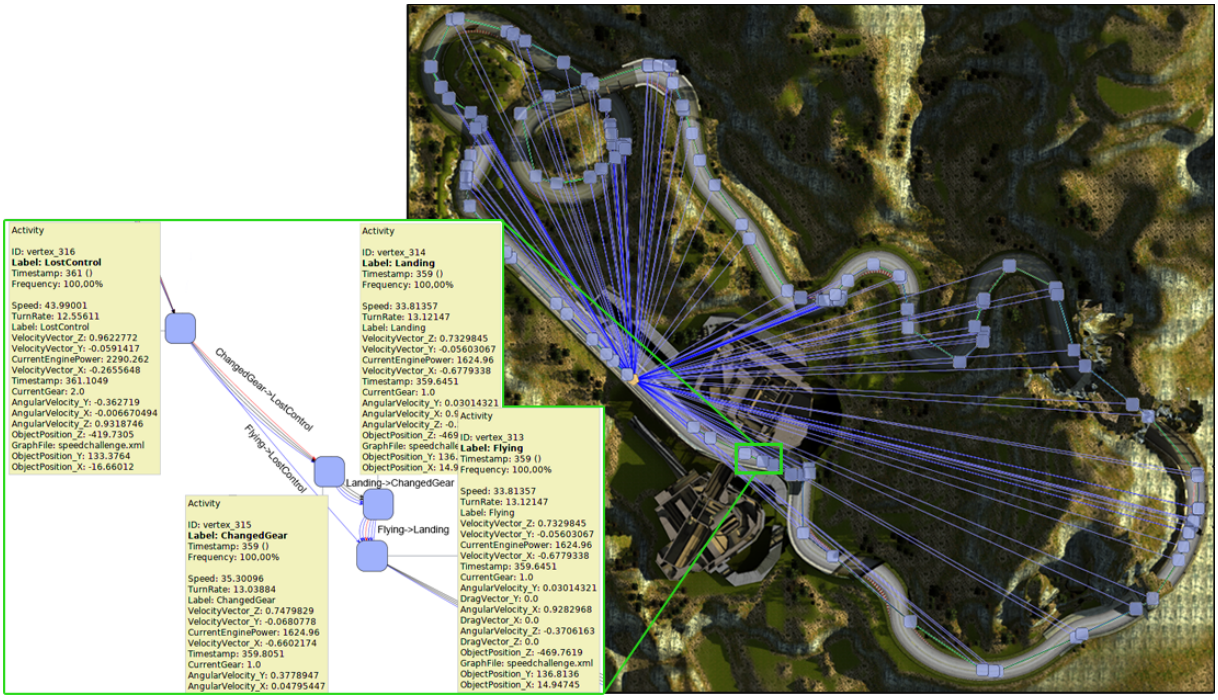

Figure 1: Example of provenance graph from a racing game prototype.

feature set-based solutions for projecting heterogeneous nodes into a single d-dimensional feature space.

2 BACKGROUND

In this section we introduce briefly the Provenance in Games (PinG) framework and its influence on node heterogeneity and address existing HGNN approaches for dealing with node heterogeneity.

2.1 Provenance in Games

The Provenance in Games (PinG) framework adapts the Open Provenance Model in the context of digital games by mapping game objects into three provenance specific type of nodes: *Agent*, *Activity* and *Entity*[9]. *Agent* nodes represent dynamic game objects such as players, non-playable characters (NPCs), enemies, monsters. *Activity* nodes represent actions and events such as jumping, attacking, taking damage or dying. *Entity* nodes represent static elements such as items, weapons, scenario objects. Also, each pair of provenance node type also define a causal relationship through labeled directed edges. For example, an edge between two Activities A and B represent that "Activity B was *triggered by* Activity A". For conciseness sake, we refer to [9] for a complete description on all edge types. Finally, nodes and edges recorded throughout a game session compose a game provenance graph that documents the whole session.

To enable gathering game provenance data during game sessions, a plugin for the game engine Unity called PinGU implemented the PinG framework[12]. Through PinGU, game developers can instrumentalize provenance data capture according to their needs and the game's nature. The game developers can define the granularity of the provenance capture process by several aspects, such as how often an ongoing action or event should generate a new *Activity* node and which features from each game object should be tracked. This choice is made by the game developer, guided by a Game Analytics task of interest.

In this context, we can illustrate how heterogeneous the resulting game provenance graphs can be. Consider a game with three characters A, B, and C, with distinct gameplay mechanics, interacting in a 3D environment through a set of actions. The resulting game provenance graph should contain nodes for characters A, B, C, and their respective features. Considering that each character has a distinct gameplay mechanic, it is fair to assume that each character has an associated feature related to that gameplay mechanic. Thus, nodes from A, B and C are heterogeneous and contain different feature sets. However, since they all interact in the same 3D environment, these nodes should also have 3D positional features in common. If we instantiate this example in industry-level games such as League of Legends, we would have to account for more than 150 characters.

This kind of node heterogeneity, in which nodes represent different objects but share intersecting feature sets, is the problem we intend to address in this work.

2.2 Graph Definitions

This paper follows the following graph definitions and Table 1 brings the adopted notations.

Definition 2.1 (Graph). *A graph is a tuple* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *where* $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ *is the set of nodes and* $\mathcal{E}$ *is the set of edges. An*

Table 1: Table of notations

Notation	Definition
$\mathbb{R}^n$	n-dimensional space
a,**a**,**A**	Scalar, vector, matrix
$\mathbf{x}_v$	Input feature vector of node v
$\mathcal{A}$	Set
$\mathcal{N}(\cdot)$	Neighborhood of a node
$\lvert\cdot\rvert$	Set cardinality

edge $e_{ij} \in \mathcal{E}$ represents the connection between two different adjacent nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$.

Definition 2.2 (Type). *A type t is a set of features $\mathcal{F}_t$ that semantically distinguish nodes or edges. t defines a dimensional feature space $\mathbb{R}^{|\mathcal{F}_t|}$.*

Definition 2.3 (Homogeneous Graph). *A homogeneous graph is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v \in \mathcal{V}$ is associated to at most one type a and each edge $e \in \mathcal{E}$ is associated to at most one type r.*

Definition 2.4 (Heterogeneous Graph). *A heterogeneous graph is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where nodes are associated with a type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and edges are associated with a type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ are sets of node types and edge types where $|\mathcal{A}| + |\mathcal{R}| > 2$.*

Definition 2.5 (Attributed Graph). *An attributed graph is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v \in \mathcal{V}$ of type t is associated to a node attribute vector $\boldsymbol{x}_v \in \mathbb{R}^{|\mathcal{F}_t|}$.*

Definition 2.6 (Node Embedding). *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the node embedding of a node $v \in \mathcal{V}$ is a d-dimensional node representation $\boldsymbol{h}_v \in \mathbb{R}^d$ with $d \ll |\mathcal{V}|$ that captures structural and semantic information of v.*

2.3 Graph Neural Networks

Graph Neural Networks (GNNs) is a general framework for generating node embeddings for nodes $v \in \mathcal{V}$ given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The GNN model relies on a Neural Message Passing (NMP) framework, in which vector messages are exchanged between nodes and updated using neural networks[4]. In general, the NMP framework follows the equation 1:

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^k\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^k\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \text{NEIGHBORS}(u)\right\}\right)\right) \quad (1)$$

in which $\mathbf{h}_u^{(k)}$ represents the embedding of a node u at iteration k, NEIGHBORS is a function over the neighborhood $\mathcal{N}_u$ of node u, and both UPDATE and AGGREGATE are arbitrary differentiable functions (i.e., neural networks)[5]. The message passing framework requires graphs to be attributed since, in the first iteration, the "current" embedding of a node is its input feature vector, i.e., $\mathbf{h}_u^0 = \mathbf{x}_u, \forall u \in \mathcal{V}$

The Heterogeneous Graph Neural Network adapts the GNN framework in order to deal with node and edge heterogeneity. The adaptations mostly encompasses two main concepts: metapaths and feature transformation. Since metapaths are beyond the scope of this work, we focus our attention on feature transformation.

In order to aggregate neighborhood information and compose the "message", HGNNs must deal with the heterogeneity of the multitude of node types. In [16], authors state the fusion problem caused by the heterogeneity of nodes as one of the three challenges of Heterogeneous Graph Embedding. The reason behind this is that AGGREGATE is a function over a set of d-dimensional vectors. Hence, node features must belong to a common vector space. The most common technique applied to realize such feature transformation is projection matrices $\mathbf{W}_a \in \mathbb{R}^{|\mathcal{F}_a| \times d}$ and can be understood as a fully connected neural network layer projecting the input vector of a node $v \in \mathcal{V}$ of type $a \in \mathcal{A}$ into a common d-dimensional space. This approach is observed in several works in literature such as HAN[17], HetGNN[18], GATNE[1], MAGNN[3] and HetSANN[6].

Regardless of how complex are the features associated with each type, projection matrices approaches tend to learn at least $|\mathcal{A}|$ matrices $\mathbf{W}_a \in \mathbb{R}^{|\mathcal{F}_a| \times d}, \forall a \in \mathcal{A}$.

In our understanding, this type-specific feature transformation approach discards the semantics of the node feature sets, assuming that they are disjoint across multiple types of nodes and edges.

3 THE SMOKE SQUADRON DATASET

In this section we present and discuss the Smoke Squadron Dataset, following three aspects: the game, the provenance graph and the dataset itself.

3.1 The Game

Smoke Squadron is a local multiplayer flight game where players fight each other with toy airplanes. Each individual airplane is equipped with a machine gun for continuous damage actions and 3 different types of projectiles (missiles) which kill at collision: Simple Missile, Smoke Missile and Fireworks Missile. Aside from traditional weaponry, planes use a solidifying smoke trail on both themselves and some missiles which kill at collision. Explicitly, the Smoke Missile creates a solid smoke trail just like the player's airplane, while the Fireworks Missile spawns smoke clouds around its explosion position. Since the position of these fatal smoke clouds are mostly defined by the trajectory of players and their projectiles, each match results in a highly dynamic battle space. Each player start with 3 lives and each airplane start with a full health and smoke counter. The health counter is affected by machine gun shots and wall collisions. The smoke counter is depleted every time the player shoots a Smoke Missile or a Fireworks Missile and is recharged by pickup items randomly spawned in the battle space. The match ends when one of the combatants lose all 3 lives.

3.2 The Provenance Graph

Like most games, Smoke Squadron presents several game objects with different feature sets and dimensionalities. In Smoke Squadron's provenance graphs, we have the following five node types corresponding to the following game objects: *Item*, *Player*, *Smoke*, *Missile (Simple & Smoke)* and *Fireworks Missile*.

For the sake of conciseness, we list below the overall features that a node from a Smoke Squadron provenance graph might contain:

- Provenance Type feature: Provenance node type, i.e., Agent, Activity, or Entity.

Table 2: Smoke Squadron node types and feature information

Node Type	Common Features	Player Features	Health	Missile Features	Explosion Time
Item	X				
Player	X	X	X		
Smoke	X		X		
Missile	X			X	
Fireworks Missile	X			X	X

- ObjectTag feature: determines the game object responsible for instantiating that node. The possible values are:
  - Player01, Player02: refers to a Player's Agent and Activity nodes.
  - Smoke: refers to Smoke's Activity and Entity nodes.
  - Rocket: refers to Rocket's Activity and Entity nodes.
  - SmokeItem: refers to SmokeItem's Entity nodes.
  - SmokeItemHalf: refers to SmokeItemHalf's Entity nodes.
  - SmokeSpawner: refers to ItemBox Agent node.
- Label feature: determines a label for the node. The possible values depend on the Type and ObjectTag values. For example, the label value of a node with ObjectTag Missile and Type Entity is the type of the missile; on the other hand, the label value of a node with the same ObjectTag but with Type Activity is the action performed by that missile.
- Health related features: health points and life counter.
- Movement related features: position, direction and rotation in axis X, Y and Z, and Speed.
- Input related features: Input on angle rotation, acceleration and braking triggers.
- Physics related features: Throttle, Rotation effect ratios.
- Weapon related features: Weapons and Smoke's counters, cooldowns and explosion timers.

However, each type of node draws a different subset of the above mentioned features. Table 2 presents an overview of Smoke Squadron's node feature information. We group all the existing features into five feature sets: *Common Features*, *Player Features*, *Health*, *Missile Features*, and *Explosion Time*, where *Health* and *Explosion Time* are single numerical features. *Common Features* encompasses general tags that identify the game object related to that node (type, objectTag, label) and its coordinates in the game world. *Player Features* encompasses input, physics, weapons, and specific movement related features. *Health* represents the life points of a game object. *Missile Features* encompasses specific missile movement and damage related features. *Explosion Time* represents the amount of time it takes for a given game object to explode. As observed in Table 2, these five feature sets are distributed across node types:

- All node types share Common Features;
- Player and Smoke nodes share the Health feature;
- Missile and Fireworks Missile distinguish themselves due to the Explosion Time feature.

These feature sets compose nodes' input vectors $\mathbf{x}_v$. As some features are categorical, they are encoded in the input vector as one-hot vectors. Some examples are the *Label feature* and the *Provenance Type feature*. Such features also increase the dimensionality of each node type's feature set. Table 3 presents the number of features for each node type. Note that even though Smoke and Fireworks Missile nodes have input vectors with the same dimensionality, their feature sets are different.

Table 3: Smoke Squadron node types statistics

Node Type	Item	Player	Smoke	Missile	Fireworks
Dim.	29	53	34	33	34
No. of Nodes	704	70617	87483	7823	1621

3.3 The Dataset

The Smoke Squadron dataset comprises the game sessions produced by four participants, with ages between 20 and 28, with no prior knowledge of the game. The players were recruited locally to participate in the capture process, and the experiment was explained in detail before the game sessions. All game provenance graphs are anonymous since they contain no identification attributes and, therefore, preserve the players' privacy.

The gameplay capture lasted three hours and generated 37 provenance graphs, totaling 168 thousand nodes and 239 thousand edges. The average number of nodes per graph is 4,427, and the average match duration is 185 seconds. Table 3 presents the number of nodes of each type in the dataset.

4 FEATURE SET ENCODING

In this section we present preliminary results of a feature set based projection approach. As observed in the previous section, works in the literature assume that heterogeneous node input vectors are independently distributed and that the feature sets that compose such vectors are disjoint. However, grouping node feature sets may emerge relationships that could be explicitly considered into the representation learning process. To this end, we propose a solution for leveraging feature sets into the HGNN architecture: Feature Set Encoding (FSE).

The objective of FSE is to encode feature set information into the node input vector $\mathbf{x}_v^{\mathcal{FSE}}$. Therefore, we rearrange node input vectors according to a general graph feature superset and concatenate a "bag of features" representation of node type's feature set:

$$\mathbf{x}_v^{\mathcal{FSE}} = \mathbf{W} \cdot \text{CONCAT}\left(\mathbf{x}_v^{\mathcal{C}}, \mathbf{b}_{\mathscr{a}_v}\right) \tag{2}$$

where $\mathbf{x}_v^{\mathcal{C}}$ is the rearranged input vector of node $v \in \mathcal{V}$, $\mathbf{b}_{\mathscr{a}_v}$ is the bag of features representation of node type $\mathscr{a}$, and $\mathbf{W}$ is a projection matrix that learns a latent representation of node input vectors that combines both feature values and feature set

information. To obtain $\mathbf{x}_v^{\mathcal{C}}$, we rearrange input vector $\mathbf{x}_v^{\mathcal{C}}$ according to C, which is an arbitrary permutation of all the features from all node types. In case a feature f does not exist in that node's type, a null value (0) is set. At the end of this process, all input vectors belong to the same dimensional space defined by $\mathcal{C}$, i.e., $\mathbf{x}_v^{\mathcal{C}} \in \mathbb{R}^{|C|}, \forall v \in \mathcal{V}$.

The "bag of features" $\mathbf{b}_{\mathscr{a}_v}$ is a binary vector where each position represents whether a feature is defined for type $\mathscr{a}_v$ (value 1) or not (value 0). Notice that in $\mathbf{x}_v^{\mathcal{C}}$, it is not possible to distinguish an original null value, i.e., an existing feature whose value is 0, from a null value for inexistent feature. This limitation, however, is fixed by the bag of features representation $\mathbf{b}_{\mathscr{a}_v}$.

The resulting FSE input vectors $\mathbf{x}_v^{\mathcal{FSE}}, \forall v \in \mathcal{V}$ preserves original feature values, encodes node feature set information and belong to the same dimensional space $\mathbb{R}^{2|C|}$. Thereby, a single projection matrix $\mathbf{W} \in \mathbb{R}^{2|C| \times d}$, where d is the input dimension of any GNN method, learns a latent representation of node input vectors that leverages both feature values and feature set information, as observed in Equation 2. For clarity's sake, we call the projection procedure defined in Equation 2 Feature Set Projection (FSP).

Table 4: Preliminary results on a Link Prediction task

Method	Precision	Recall
TSP + GNN	**0.53** (6e-4)	0.605 (4e-4)
FSP + GNN	0.514 (7.76e-4)	**0.626** (6.4e-4)

In a preliminary experiment, we performed a Link Prediction task comparing type-specific projection (TSP) matrices and the FSP. In the context of Game Provenance Graphs, the goal of a Link Prediction task is to infer influence between nodes. Since nodes represent both objects and events, we exemplify an application of Link Prediction for this particular dataset: consider that player B is dodging from player A's attack. If player B collides with an obstacle, it is not trivial to analytically determine whether the collision is a direct consequence of player A's attack due to all the interacting elements' complexity. However, a game analyst could benefit from assessing how often that influence occurs during gameplay sessions for game balancing. Thus, we firmly believe that a graph representation learning approach is best suitable for such a task.

The results of our preliminary experiment are shown in Table 4. The GNN used was an unsupervised GraphSAGE implementation in Pytorch-Geometric which learns node embeddings. To perform link prediction, we concatenate node embeddings and pass them through a fully connected layer. Results shown that TSP and FSP achieve similar results. However, while TSP learns 5 different projection matrices for each node type, FSP learns a single projection matrix for all node types.

5 CONCLUSION

In this work we presented the heterogeneous graph dataset Smoke Squadron, containing a node heterogeneity challenge: how to leverage feature set information in heterogeneous graph neural networks. Using game provenance graphs of a multiplayer game, we present graphs in which distinct node types share features subsets among themselves. As HGNNs are prone to be employed in complex and highly heterogeneous domains, such as observed with Game Analytics, we believe that intersection among feature sets might occur more frequently in newer datasets. However, to the best of our knowledge, works in the graph representation literature have always assumed node types and feature sets to be disjoint, hence using type-specific projection matrices. This approach learns each projection matrix independently and disregards relationships among feature sets. We also present preliminary results of a feature set based projection approach which is less costly than type-specific projection approaches and achieves competitive results.

ACKNOWLEDGMENTS

We would like to thank CAPES, CNPq, and FAPERJ for the financial support.

REFERENCES

[1] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1358–1368.

[2] Felipe Machado Figueira, Lucas Nascimento, Jose da Silva Junior, Troy Kohwalter, Leonardo Murta, and Esteban Clua. 2018. Bing: A framework for dynamic game balancing using provenance. In *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 57–5709.

[3] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 1263–1272.

[5] William L. Hamilton. 2020. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, 3 (2020), 1–159.

[6] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4132–4139.

[7] Lidson Jacob, Esteban Clua, and Daniel de Oliveira. 2017. Oh Gosh!! Why is this game so hard? Identifying cycle patterns in 2D platform games using provenance data. *Entertainment Computing* 19 (2017), 65–81.

[8] Lidson Barbosa Jacob, Troy C Kohwalter, Alex FV Machado, Esteban WG Clua, and Daniel De Oliveira. 2014. A non-intrusive approach for 2d platform game design analysis based on provenance data extracted from game streaming. In *2014 Brazilian Symposium on Computer Games and Digital Entertainment*. IEEE, 41–50.

[9] Troy Kohwalter, Esteban Clua, and Leonardo Murta. 2012. Provenance in games. In *Brazilian Symposium on Games and Digital Entertainment (SBGAMES)*. 11.

[10] Troy C Kohwalter, Esteban GW Clua, and Leonardo GP Murta. 2013. Game flux analysis with provenance. In *International Conference on Advances in Computer Entertainment Technology*. Springer, 320–331.

[11] Troy C Kohwalter, Leonardo GP Murta, and Esteban WG Clua. 2020. Provchastic: Understanding and Predicting Game Events Using Provenance. In *International Conference on Entertainment Computing*. Springer, 90–103.

[12] Troy Costa Kohwalter, Leonardo Gresta Paulino Murta, and Esteban Walter Gonzalez Clua. 2017. Capturing game telemetry with provenance. In *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 66–75.

[13] Brian F Lavoie and Richard Gartner. 2005. *Preservation metadata*. OCLC.

[14] Sidney Araujo Melo, Troy C Kohwalter, Esteban Clua, Aline Paes, and Leonardo Murta. 2020. Player Behavior Profiling through Provenance Graphs and Representation Learning. In *International Conference on the Foundations of Digital Games*. 1–11.

[15] Sidney Araujo Melo, Aline Paes, Esteban Walter Gonzalez Clua, Troy Costa Kohwalter, and Leonardo Gresta Paulino Murta. 2019. Detecting long-range cause-effect relationships in game provenance graphs with graph-based representation learning. *Entertainment Computing* 32 (2019), 100318.

[16] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S Yu. 2020. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *arXiv preprint arXiv:2011.14867* (2020).

[17] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[18] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.