Palestine Journal of Mathematics

Vol 15(Special Issue II)(2026) , 70–84

The Implementation of a Spatial Temporal Graph Neural Network and Rainbow Antimagic Coloring for Traffic Flow Time Series Forecasting

Ika Hesti Agustin, M. Venkatachalam, Indah Lutfiyatul Mursyidah, Rifki Ilham Baihaki and Marsidi

MSC 2010 Classifications: Primary 33C20; Secondary 33C65.

Keywords and phrases: Spatial temporal graph neural network, rainbow antimagic coloring, traffic flow time series forecasting.

The authors would like to thank the reviewers and editor for their constructive comments and valuable suggestions that improved the quality of our paper. .

Corresponding Author:Ika Hesti Agustin

Abstract Accurate traffic flow forecasting is essential for urban traffic control management. Due to the nonlinearity and complexity of traffic flow, traditional methods cannot meet the requirements of medium and long-term forecasting tasks and often ignore spatial and temporal dependencies. Instead of applying regular convolutions and repeating units, we formulate the problem as a graph and build the model using a spatial-temporal graph neural network. It is considered to enable faster training using certain variables. In this paper, we use Spatial-Temporal Graph Neural Network (STGNN) to solve the time-series forecasting problem in the urban transportation area of the Jember district. Then, we relate the problem to one of the graph theories, namely rainbow antimagic coloring, to place the ETLE CCTV and analyze the traffic flow based on vehicle speed, number of accesses to the road, rain intensity, and crowds. The results show that the Spatial Temporal Graph Neural Network and Rainbow Antimagic Coloring are effective tools for Traffic Flow Time Series Forecasting. The stage of the analysis consists of three steps, namely vertex embedding step, training step, testing step, and forecasting step. We used two STGNN architectures, namely STGNN-476 and STGNN-656. The results showed that the best STGNN model is Cascadeforwardnet-656 with MSE testing of $7,3730 \times 10^{-13}$ and regression correlation of 1.

1 Introduction

Everybody's daily life is significantly impacted by transportation. The rising traffic volume on the city's main roadways is a direct result of the increasing activity of residents in the town of Jember from year to year, both from activities of residents outside the city and from residents of the town. The main roads that feel the impact of the increasing traffic flow in the town of Jember are Ahmad Yani Street, Trunojoyo Street, Hoscokroaminoto Street, Gajahmada Street, Hayamwuruk Street, Brawijaya Street, Sultan Agung Street, P.B. Sudirman Street, Seruji Street, and Slamet Riyadi Street.

Population growth, private property ownership, and numerous issues like traffic flow management exacerbating traffic are all intimately tied to traffic flow [23]. Science and technology advance in tandem with traffic flow. One of the advancements in science and technology is the ability to forecast traffic flow through traffic flow forecasting. The Intelligence Transport System (ITS), a system that integrates information, communication, transportation infrastructure, and vehicles, is one of the applications that help predict traffic. As a result, accurate real-time traffic forecasts are crucial for government, business, and road users [21].

Fundamental traffic flow characteristics, such as speed, rainfall intensity, road length, crowd level, and the number of accesses to a road, are frequently used in traffic studies as indicators

Figure 1. The examples of applications of ANN [9]

to track present traffic conditions and forecast future ones [24]. Traffic forecasts are typically divided into two scales based on the length of the prediction: short-term ($5 \sim 30$ min), medium-term, and long-term (over 30 min). Most widely used statistical methods, such as linear regression, can accurately predict short time intervals. However, those techniques are less useful for reasonably long-term projections because of the complexity and ambiguity of traffic flow [25]. Dynamic modeling and data-driven techniques are two categories that it can use to categorize previous work on mid- and long-term traffic prediction. Dynamical modeling uses mathematical tools (e.g. differential equations) and physical knowledge to formulate traffic problems by computational simulation [20].

Artificial neural networks (ANNs) have recently gained popularity and have proven to be helpful models for classification, clustering, pattern recognition, and prediction in a various fields [26]. The classification utilizes provided class labels to arrange the articles in the information assortment [13]. Besides classification, machine learning is also used for the clustering process [27]. For instance, Batyha used a clustering algorithm to extract indicators of the social security system for disabled people by exploring the historical dimension of social security for the disabled [1].

Regarding usefulness, ANNs, one type of machine learning (ML) model, are now competitive with traditional statistical and regression models [6]. ANNs can be created and used for various tasks, including image recognition and natural language processing. Because of their outstanding characteristics of self-learning, adaptivity, fault tolerance, nonlinearity, and advancement in input to output mapping, ANNs are now primarily employed for universal function approximation in numerical paradigms [16]. The examples of applications of ANN can be seen in Figure 1.

The application of ANN is an effective way to analyze traffic flow as a preventive action. The traffic flow is considered a non-Euclidean data source from the ANN input data perspective. To handle this type of data source, the use of graph neural network (GNN) is considered to be a breakthrough in machine learning. By a graph, we mean a structure $S = (V(G), E(G))$, where $V(G)$ is a finite nonempty set of elements called vertices, and $E(G)$ is a set (possibly empty) of unordered pairs $\{u, v\}$ of vertices $u, v \in V(G)$ called edges [7, 8]. The number of vertices of a graph G is the order of G, denoted by $\mid V(G) \mid$. The number of edges is the size of G, denoted by $\mid E(G) \mid$. The degree of vertex v in a graph G is the number of vertices adjacent to v and denoted by $d(v)$ [15]. Let $u, v \in V(G)$, vertex u said to be adjacent to v if there is an edge e between u and v, that is $e = uv$. Vertex v is then called a neighbor of u. $N(u)$ denotes the set of all neighbors of u. We also say that u and v are incident with edge e. The adjacency matrix of graph G is the $n \times n$ matrix $A = [a_{i,j}]$, where $[a_{i,j}] = \begin{cases} 1 & \text{if } v_i v_j \in E(G) \\ 0 & \text{otherwise.} \end{cases}$ The example of adjacency matrix, graph, and more complicated graph can be seen in Figure 2 and Figure 3.

A machine learning algorithm known as a graph neural network (GNN) can extract crucial data from networks to make helpful predictions. Graphs are growing more prevalent and information-rich, and artificial neural networks are getting better at what they do. Aside from artificial neural networks, GNNs has developed into potent tools for many crucial applications (ANN). The GNN technique adopts the framework of the Convolutional Neural Network (CNN). Since performing CNN on graphs is particularly challenging due to the arbitrary size of the graph and its complex topology, which implies there is no spatial locality, we must use the Graph Neural Network (GNN). There is also un-fixed vertex ordering. If we first labeled the vertices x_1, x_2, x_3, x_4, x_5, and the second time we labeled the graph with x_3, x_1, x_2, x_5, x_4, then the graph

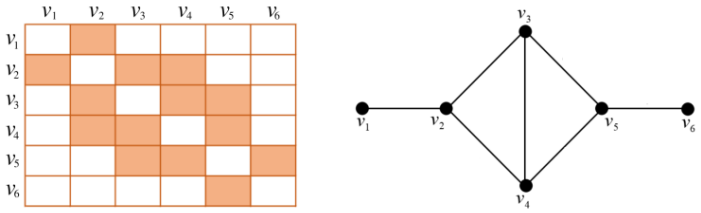

Figure 2. The example of adjacency matrix and graph

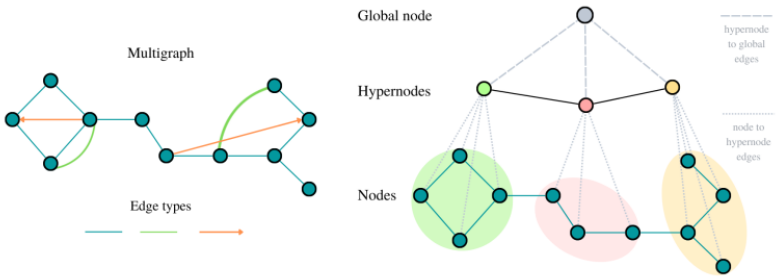

Figure 3. Illustrations of more complicated graphs. An example of a multigraph with three different edge types, including a directed edge, is shown on the left. The intermediate-level nodes of the three-level hierarchical network on the right are called hypernodes.

will not change.

Node-level prediction, edge-level prediction, and graph-level prediction are the objectives of graph neural networks. Figure 4 shows an end-to-end prediction job using a GNN model. Vertex embedding is a concept used in graph theory that entails translating vertex properties to a d-dimensional embedding space (low dimensional space rather than the actual dimension of the graph). To achieve this, vertices must be mapped, so that similarity in the embedding space roughly corresponds to a similarity in the graph. Figure 5 serves as an illustration of node embedding.

There are some other GNN kinds. Graph Autoencoders (GAEs), Convolutional Graph Neural Networks (ConvGNNs), Recurrent Graph Neural Networks (RecGNNs), and Spatial Temporal Graph Neural Network are the four groups that the authors in [22] claim the categorization of GNN. In this research, we discuss STGNN. To solve a dynamic graph problem, STGNN is built on the idea of concurrently modelling spatial and temporal interdependence. The spatial-temporal graph structure in STGNN is dynamic because the properties of the vertex or edge might change over time [3]. A common use of STGNN is the analysis of street traffic flow networks, where a graph structure can characterize street networks. Specifically, each vertex represents a sensor location that monitors vehicle speed, the number of accesses to the street, rain intensity, and crowds, which change over time. Those are the vertex features of each street. Bengio et al. [2] first presented a word embedding approach to developing distributional similarity-based representations of words. Mikolov et al. [16] expanded this method by increasing the effectiveness of word embedding to create Word2vec. Each word may be represented by a real-

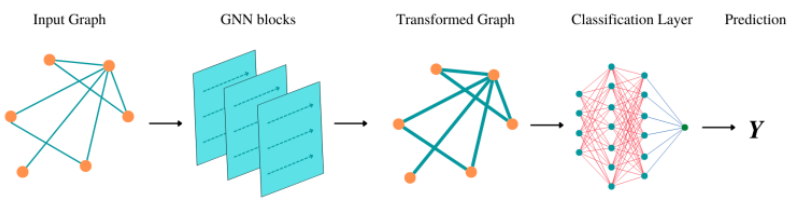

Figure 4. An end-to-end prediction task with a GNN model.

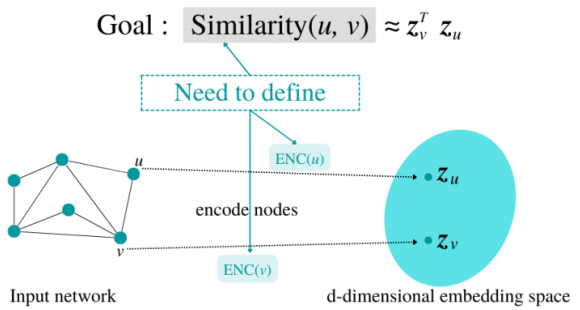

Figure 5. The illustration of node embedding.

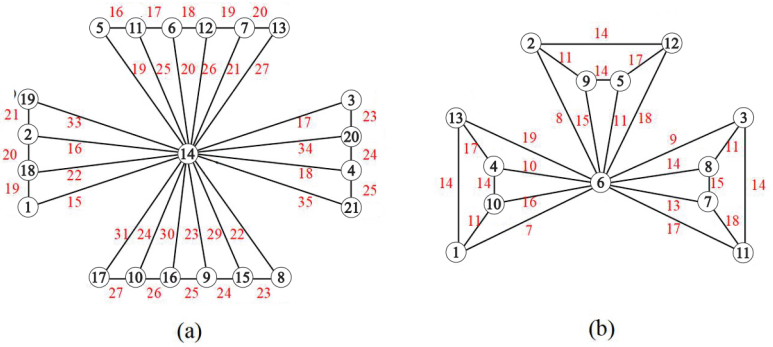

Figure 6. The illustration of the rainbow antimagic coloring of (a) $Amal(B_{4,5}, v, 2)$ and (b) $Amal(W_4, v, 3)$.

valued vector and mapped into a vector space using Word2vec. The abstract semantic content of a word is represented by the word vector in the vector space.

This study aims to apply the STGNN (Spatial-Temporal Graph Neural Network) integrated with the RAC (Rainbow Antimagic Connection Number) to analyze traffic flow and place the ETLE CCTV. Before we explain about rainbow antimagic coloring, we introduce an edge coloring of a graph G. An edge coloring of a graph G is a function $f : E(G) \to C$, where C is a set of distinct colors. For any positive integer k, a $k-$edge coloring is an edge coloring that uses exactly k distinct colors [19]. One of the topics in edge coloring of graphs is rainbow antimagic coloring

A graph G is said to be a rainbow antimagic coloring if there is a bijection function $f : V(G) \to \{1, 2, \ldots, V(G)\}$, and the associated weight of an edge $uv \in E(G)$ under f is $w_f(uv) = f(u) + f(v)$. If there is $w_f(uv) \neq w_f(u'v')$ for every two edges $uv, u'v'$ in $E(P)$, a path P in a vertex-labeled graph G is said to be a rainbow path. Suppose there is a rainbow uv path for each pair of vertices u and v of G. In that case, f is referred to as a rainbow antimagic labelling of G [4]. Graph G admits a rainbow antimagic coloring when we give each edge uv the color of the edge weight $w_f(uv)$. The minimum number of colors taken over all rainbow colorings produced by rainbow antimagic labelings of G is the rainbow antimagic connection number, indicated by $rac(G)$ [14]. Figure 6 [17] is an example of rainbow antimagic coloring.

2 Method

This research uses analytical and experimental methods. In the analytical study, we use a mathematical deductive approach to describe the findings, and in the experimental method, we use computer programming to do simulation. We will analyze the traffic flow prediction of ten streets in Jember district, East Java, Indonesia. First, we use a word embedding method to learn distributional similarity based representations of words. Second, we will show the vertex embedding process of a single layer GNN of a given graph with five features data, namely vehicle speed, number of access to the street, rain intensity, crowds, and number of vehicles data for 30 days

observation. Third, we will develop the STGNN programming, train a model using 70% data input obtained from vertex embedding process, testing and finally forecast the traffic flow prediction. The last, we analyze the use of rainbow antimagic coloring on graph representation to determine the number of ETLE administrators.

The following is the algorithm for studying the traffic flow prediction by using STGNN combined by rainbow antimagic coloring [5].

Single Layer GNN Algorithm

Step 0. Given that a graph $G(V, E)$ of order n and feature matrix $H_{p\times q}$ of p vertices and q features, and give a tolerance ϵ.

Step 1. Set a matrix $B = A + I$, where I is an identity matrix, and set A is the matrix adjacency of graph G.

Step 2. Initialize weights W, bias β, learning rate α. (For simplicity, set $W_{m\times 1} = [w_1 w_2 ... w_m]$, where $0 < w_j < 1$, bias $\beta = 0$ and $0 < \alpha < 1$)

Step 3. Multiply weight matrix with vertex features, by setting a message function $m_u^l = MSG^l(h_u^{l-1})$, for linear layer $m_u^l = W^l(h_u^{l-1})$.

Step 4. Aggregate the message from vertex $v's$ neighbors, by setting function $h_v^l = AGG\{m_u^{l-1}\}$, and by applying the sum(.) function $h_v^l = SUM^l\{m_u^{l-1}, u \in N(v)\}$, in regards with matrix B.

Step 5. Determine the error, by setting $error' = \frac{||h_{v_i} - h_{v_j}||^2}{|E|}$, where v_i, v_j are any two adjacent vertices.

Step 6. Observe whether $error \leq \epsilon$ or not. If yes then stop, if not then do Step 7 to update the learning weight matrix W.

Step 7. Update the learning weight matrix by setting $W^{l+1} = W_j^l + \alpha \times z_j \times e^l$ where z_j is the sum of each column in the $H_{v_i}^l$ and divide by the number of nodes.

Step 8. Do Step 3-6 till the $error \leq \epsilon$.

Step 9. Save the embedding results into a vector, by naming the vector file with *embedding_ data.mat.* When the data is a time series data, then do the same process for the next time data observation.

Step 10. Load the *embedding_data.mat* then use the time series machine learning to do forecasting.

Step 11. Have the best training, testing, and forecasting results on two ANN architectures.

Step 12. If the best MSE has attained then STOP.

3 Main results

From now on, we will discuss the research result and describe the following. We first show the nodes embedding. Dafik et.al. analyze the embedding process of vertex features and find Observation 1 to do time series forecasting.

Observation 1. [9] Given that a graph G of order n. Suppose that vertex and edge sets are $V(G) = \{v_1, v_2, ..., v_{n-1}, v_n\}$ and $E(G) = \{v_i v_j | v_i, v_j \in V(G)\}$, respectively. Given that vertex features as follows $H_{v_i} = \begin{bmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,m} \\ s_{2,1} & s_{2,2} & \dots & s_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \dots & s_{n,m} \end{bmatrix}$. The vertex embedding can be determined using the massages passing from vertex $v's$ neighbors $h_v^l = AGG\{m_u^{l-1}, u \in N(v)\}$ under the aggregation **sum**$(\cdot)$, thus $h_v^l = SUM^l\{m_u^{l-1}, u \in N(v)\}$ in regard to the matrix $B = A + I$ where A, I are adjecency and identity matrix, respectively.

To better understand about the graph network, let us give some technical examples of a specific graph with a particular feature of each vertex or node.

Example 1 *Given that a graph G of order four. Suppose that vertex and edge sets are* $V(G) = \{v_1, v_2, v_3, v_4\}$ *and* $E(G) = \{v_1v_2, v_2v_3, v_2v_4, v_3v_4\}$*, respectively. Given that feature node as*

$$H^0_{v_i} = \begin{bmatrix} 0.26 & 0.40 & 0.69 & 0.90 & 0.77 \\ 0.10 & 0.52 & 0.87 & 0.90 & 0.90 \\ 0.90 & 0.35 & 0.20 & 0.10 & 0.39 \\ 0.58 & 0.16 & 0.17 & 0.50 & 0.57 \end{bmatrix}$$

Obtain the nodes embedding with one hidden layer with one neuron, and with minimal loss function.

Solution. Based on the above graph, we can determine the adjecency, identity, and loop-adjecency matrices as follows.

$$A(G) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = A + I = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Learning weight $W^1 = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$

The first iteration can be described as follows:

$$\begin{aligned}
m^l_{v_i} &= W^l \cdot h^{l-1}_{v_i}, \text{ where } i = 1, 2, 3, 4 \\
m^1_{v_1} &= W^1 \cdot h^0_{v_1} \\
&= \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix} \cdot \begin{bmatrix} 0.26 & 0.40 & 0.69 & 0.90 & 0.77 \end{bmatrix} = \begin{bmatrix} 0.1510 \end{bmatrix} \\
m^1_{v_2} &= W^1 \cdot h^0_{v_2} \\
&= \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix} \cdot \begin{bmatrix} 0.10 & 0.52 & 0.87 & 0.90 & 0.90 \end{bmatrix} = \begin{bmatrix} 0.1645 \end{bmatrix} \\
m^1_{v_3} &= W^1 \cdot h^0_{v_3} \\
&= \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix} \cdot \begin{bmatrix} 0.90 & 0.35 & 0.20 & 0.10 & 0.39 \end{bmatrix} = \begin{bmatrix} 0.0970 \end{bmatrix} \\
m^1_{v_4} &= W^1 \cdot h^0_{v_4} \\
&= \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix} \cdot \begin{bmatrix} 0.58 & 0.16 & 0.17 & 0.50 & 0.57 \end{bmatrix} = \begin{bmatrix} 0.0990 \end{bmatrix}
\end{aligned}$$

thus, we have $m^1_{v_i}$:

$$m^1_{v_i} = \begin{bmatrix} 0.1510 \\ 0.1645 \\ 0.0970 \\ 0.0990 \end{bmatrix}$$

By considering the matrix B, we only include the non zeros element of $m^1_{v_i}$, thus we have

$$m^1_{v_1} = \begin{bmatrix} 0.1510 \\ 0.1645 \end{bmatrix}, m^1_{v_2} = \begin{bmatrix} 0.1510 \\ 0.1645 \\ 0.0970 \\ 0.0990 \end{bmatrix}, m^1_{v_3} = \begin{bmatrix} 0.1645 \\ 0.0970 \\ 0.0990 \end{bmatrix}, m^1_{v_4} = \begin{bmatrix} 0.1645 \\ 0.0970 \\ 0.0990 \end{bmatrix}$$

Take the sum of the elements of each nodes embedding are as follows: $h^1_{v_1} = 0.3155, h^1_{v_2} = 0.5115, h^1_{v_3} = 0.3605, h^1_{v_4} = 0.3605$. Thus, we have the first of aggregation

$$h^1_{v_i} = \begin{bmatrix} 0.3155 \\ 0.5115 \\ 0.3605 \\ 0.3605 \end{bmatrix} \text{ where } i = 1, 2, 3, 4.$$

The loss (e) can be calculated as

$$e^1 = \frac{||h^1_{v_1}-h^1_{v_2}||+||h^1_{v_2}-h^1_{v_3}||+||h^1_{v_3}-h^1_{v_4}||+||h^1_{v_4}-h^1_{v_2}||}{|E(G)|} = 0.0623$$

In the second iteration, we need update $H^{l-1}_{v_i}$ first:

$$\begin{aligned}
H^{l-1}_{v_i} &= \frac{h^{l-2}_{v_i}}{\sum(h^{l-2}_{v_i})} \times h^{l-1}_{v_i}, \text{ where } i = 1,2,3,4 \\
H^1_{v_1} &= \frac{\begin{bmatrix}0.26 & 0.40 & 0.69 & 0.90 & 0.77\end{bmatrix}}{\sum\begin{bmatrix}0.26 & 0.40 & 0.69 & 0.90 & 0.77\end{bmatrix}} \times 0.3155 \\
&= \begin{bmatrix}0.0272 & 0.0418 & 0.0721 & 0.0940 & 0.0804\end{bmatrix} \\
H^1_{v_2} &= \frac{\begin{bmatrix}0.10 & 0.52 & 0.87 & 0.90 & 0.90\end{bmatrix}}{\sum\begin{bmatrix}0.10 & 0.52 & 0.87 & 0.90 & 0.90\end{bmatrix}} \times 0.5115 \\
&= \begin{bmatrix}0.0156 & 0.0809 & 0.1353 & 0.1399 & 0.1399\end{bmatrix} \\
H^1_{v_3} &= \frac{\begin{bmatrix}0.90 & 0.35 & 0.20 & 0.10 & 0.39\end{bmatrix}}{\sum\begin{bmatrix}0.90 & 0.35 & 0.20 & 0.10 & 0.39\end{bmatrix}} \times 0.3605 \\
&= \begin{bmatrix}0.1672 & 0.0650 & 0.0372 & 0.0186 & 0.0725\end{bmatrix} \\
H^1_{v_4} &= \frac{\begin{bmatrix}0.58 & 0.16 & 0.17 & 0.50 & 0.57\end{bmatrix}}{\sum\begin{bmatrix}0.58 & 0.16 & 0.17 & 0.50 & 0.57\end{bmatrix}} \times 0.3605 \\
&= \begin{bmatrix}0.1056 & 0.0291 & 0.0310 & 0.0910 & 0.0104\end{bmatrix}
\end{aligned}$$

thus, we have $H^1_{v_i}$:

$$H^1_{v_i} = \begin{bmatrix}H^1_{v_1}\\H^1_{v_2}\\H^1_{v_4}\\H^1_{v_1}\end{bmatrix} = \begin{bmatrix}0.0272 & 0.0418 & 0.0721 & 0.0940 & 0.0804\\0.0156 & 0.0809 & 0.1353 & 0.1399 & 0.1399\\0.1672 & 0.0650 & 0.0372 & 0.0186 & 0.0725\\0.1056 & 0.0291 & 0.0310 & 0.0910 & 0.0104\end{bmatrix}$$

Now, we need to update the learning weight. Before that, we need take the sum of each column in the $h^1_{v_i}$ and divide them by number of nodes as follows: $z_1 = 0.0789, z_2 = 0.0542, z_3 = 0.0689, z_4 = 0.0859, z_5 = 0.0992$. Thus, we have $z_k = \begin{bmatrix}0.0789 & 0.0542 & 0.0689 & 0.0859 & 0.0992\end{bmatrix}$ where $k = 1,2,3,4,5$. Given that the learning rate α, we can update the weight w as

$$\begin{aligned}
W^{k+1} &= W^k_j + \alpha \times z_k \times e^k, \text{ where } j = 1,2,3,4,5 \\
W^2 &= W^1_j + \alpha \times z_k \times e^k, \text{ for } k = 1 \text{ and } \alpha = 0.1 \\
W^2_1 &= W^1_1 + \alpha \times z_1 \times e^1 = 0.05 + 0.1 \times 0.0789 \times 0.0623 = 0.0505 \\
W^2_2 &= W^1_2 + \alpha \times z_2 \times e^1 = 0.05 + 0.1 \times 0.0542 \times 0.0623 = 0.0503 \\
W^2_3 &= W^1_3 + \alpha \times z_3 \times e^1 = 0.05 + 0.1 \times 0.0689 \times 0.0623 = 0.0504 \\
W^2_4 &= W^1_4 + \alpha \times z_4 \times e^1 = 0.05 + 0.1 \times 0.0859 \times 0.0623 = 0.0505 \\
W^2_5 &= W^1_5 + \alpha \times z_5 \times e^1 = 0.05 + 0.1 \times 0.0992 \times 0.0623 = 0.0506
\end{aligned}$$

Thus, we have W^2 :

$$W^2 = \begin{bmatrix}W^2_1 & W^2_2 & W^2_3 & W^2_4 & W^2_5\end{bmatrix} = \begin{bmatrix}0.0505 & 0.0503 & 0.0504 & 0.0505 & 0.0506\end{bmatrix}$$

By this new W^2 in hand, we can calculate the second iteration as follows.

$$
\begin{aligned}
m_{v_i}^{l} &= W^{l} \cdot H_{v_i}^{l-1}, \text{ where } i = 1,2,3,4 \\
m_{v_1}^{2} &= W^{2} \cdot H_{v_1}^{1} \\
&= \begin{bmatrix} 0.0505 & 0.0503 & 0.0504 & 0.0505 & 0.0506 \end{bmatrix} \cdot \begin{bmatrix} 0.0272 & 0.0418 & 0.0721 & 0.0940 & 0.0804 \end{bmatrix} \\
&= \begin{bmatrix} 0.0159 \end{bmatrix} \\
m_{v_2}^{2} &= W^{2} \cdot H_{v_2}^{1} \\
&= \begin{bmatrix} 0.0505 & 0.0503 & 0.0504 & 0.0505 & 0.0506 \end{bmatrix} \cdot \begin{bmatrix} 0.0156 & 0.0809 & 0.1353 & 0.1399 & 0.1399 \end{bmatrix} \\
&= \begin{bmatrix} 0.0258 \end{bmatrix} \\
m_{v_3}^{2} &= W^{2} \cdot H_{v_3}^{1} \\
&= \begin{bmatrix} 0.0505 & 0.0503 & 0.0504 & 0.0505 & 0.0506 \end{bmatrix} \cdot \begin{bmatrix} 0.1672 & 0.0650 & 0.0372 & 0.0186 & 0.0725 \end{bmatrix} \\
&= \begin{bmatrix} 0.0182 \end{bmatrix} \\
m_{v_4}^{2} &= W^{2} \cdot H_{v_4}^{1} \\
&= \begin{bmatrix} 0.0505 & 0.0503 & 0.0504 & 0.0505 & 0.0506 \end{bmatrix} \cdot \begin{bmatrix} 0.1056 & 0.0291 & 0.0310 & 0.0910 & 0.0104 \end{bmatrix} \\
&= \begin{bmatrix} 0.0182 \end{bmatrix}
\end{aligned}
$$

thus, we have $m_{v_i}^{2}$:

$$
m_{v_i}^{2} = \begin{bmatrix} 0.0159 \\ 0.0258 \\ 0.0182 \\ 0.0182 \end{bmatrix}
$$

By considering the matrix B, we only include the non zeros element of $m_{v_i}^{2}$, thus we have

$$
m_{v_1}^{2} = \begin{bmatrix} 0.0159 \\ 0.0258 \end{bmatrix}, m_{v_2}^{2} = \begin{bmatrix} 0.0159 \\ 0.0258 \\ 0.0182 \\ 0.0182 \end{bmatrix}, m_{v_3}^{2} = \begin{bmatrix} 0.0258 \\ 0.0182 \\ 0.0182 \end{bmatrix}, m_{v_4}^{2} = \begin{bmatrix} 0.0258 \\ 0.0182 \\ 0.0182 \end{bmatrix}
$$

Take the sum of the elements of each nodes embedding are as follows: $h_{v_1}^{2} = 0.0414, h_{v_2}^{2} = 0.0775, h_{v_3}^{2} = 0.0617,$ and $h_{v_4}^{2} = 0.0617$. Thus, we have the second iteration of aggregation

$$
h_{v_i}^{2} = \begin{bmatrix} 0.0417 \\ 0.0781 \\ 0.0622 \\ 0.0622 \end{bmatrix} \text{ where } i = 1,2,3,4.
$$

The loss (e) can be calculated as

$$
e^{2} = \frac{||h_{v_i}^{l} - h_{v_j}^{l}||_{inf}}{|E(G)|} \text{ where } i,j \in \{1,2,3,4\} = 0.0085
$$

Nodes Embedding

In this paper, we used data without normalization and data with normalization. We use alpha = 0.1, iterations of 3, 6, and 10, and initial weights are 0.5, 0.3, and 0.1. The results of the experiment can be seen in Table 1. Based on the table, it can be seen that the error value in data without normalization increase with each iteration. A comparison of these error values can be seen in Figure 7. The selection of the initial weights used also needs to be precise. Based on the three weight values that we use, the initial weight of 0.1 is the weight that produces the smallest error value.

Table 1. The results of GNN nodes embedding for finding the best loss.

Data Type	Iteration Numbers	Learning Weight	Error Value
Non-Normalization	3	0.5	2.8926×10^{30}
		0.3	2.9151×10^{28}
		0.1	1.4810×10^{24}
	6	0.5	3.2543×10^{45}
		0.3	3.2543×10^{40}
		0.1	3.2543×10^{37}
	10	0.5	2.9324×10^{123}
		0.3	5.1892×10^{120}
		0.1	2.9743×10^{101}
Normalization	3	0.5	77.0926
		0.3	3.5797
		0.1	0.0449
	6	0.5	3.1318×10^{51}
		0.3	1.6087×10^{16}
		0.1	0.0038
	10	0.5	3.4351×10^{23}
		0.3	2.0821
		0.1	$\mathbf{1.3006} \times 10^{-4}$

Time Series Forecasting

Then, we will perform a computer simulation on two neural network architectures to train, test, and forecast traffic flow in Jember. The data on the time series is obtained from the GNN embedding process in the previous stage. We used Matlab programming to perform numerical simulations. The neural network architecture that we use are ANN-476 and ANN-656. At the same time, the ANN models that we use are Feedforwardnet, Paternnet, Fitnet, and Cascade-forwardnet. The parameters we use to train the network are the Lavenberg-Marquadt training function, the sigmoid log transfer function, the sigmoid hyperbolic tangent, the number epochs of 750, and the learning rate of 0.1. The results of this training and testing can be seen in Table 2. Table 2 shows the best model generated by ANN-656 cascadeforwardnet, with a test MSE of 7.3730×10^{-13}.

Table 2. The performance indicator of some ANN architectures and models on training and testing traffic flow data set

ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)476	1.1712×10^{-10}	0.99995	1.2858 s	2.0393×10^{-11}
	ANN-(a)656	3.7763×10^{-10}	0.99981	1.2123 s	4.9943×10^{-10}
Fitnet	ANN-(b)476	9.8578×10^{-11}	0.99996	3.8634 s	3.1383×10^{-10}
	ANN-(b)656	6.3841×10^{-11}	0.99998	0.8984 s	7.714×10^{-12}
Patternnet	ANN-(c)476	1.3912×10^{-9}	0.999	9.9946 s	9.6447×10^{-9}
	ANN-(c)656	3.8303×10^{-10}	0.9989	1.0778 s	9.5994×10^{-10}
Cascadeforwardnet	ANN-(d)476	2.0276×10^{-9}	0.99905	1.9033 s	6.9357×10^{-10}
	ANN-(d)656	9.3442×10^{-13}	1	1.3809 s	$\mathbf{7.3730} \times 10^{-13}$

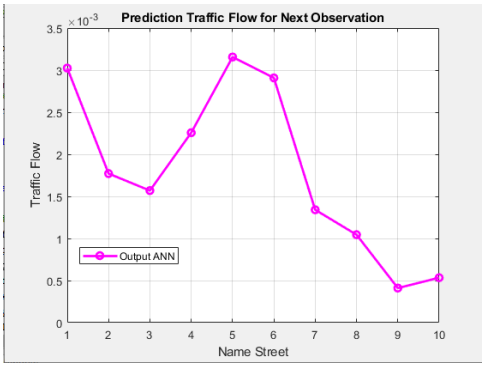

Figure 7. The forecasting traffic flow to determine streets that have the potential to experience traffic jams. The illustration shows that the most street that has the potential to experience traffic jams is street number 5, or in other words, Sultan Agung Street

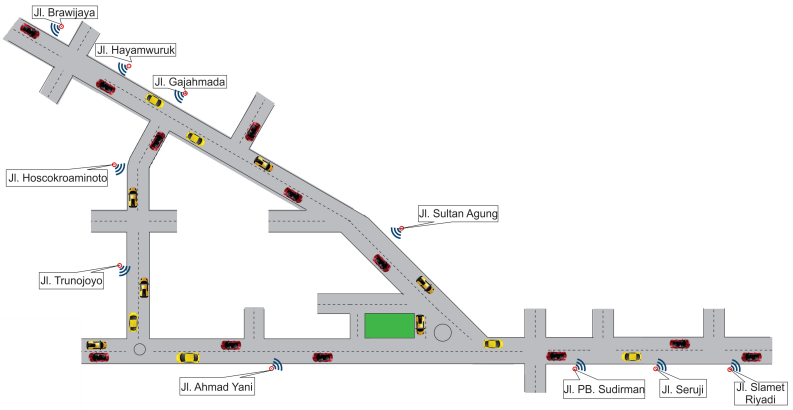

Figure 8. Traffic Flow Visualitation

According to Figure 7, the traffic data for the following day is in the fifth data. The regression that we get is 1.

The relationship between STGNN and Rainbow Antimagic Coloring

ETLE, which stands for "Electronic Traffic Law Enforcement," is the use of information technology to catch traffic law violations electronically. This helps to keep traffic safe, orderly, and safe. This is meant to make sure that the law is applied the same way to everyone involved in traffic. Determining the number of ETLE administrators is an important factor in traffic monitoring. Determining the number of ETLE administrators is kept to a minimum so they can monitor traffic violations on roads that have the potential to be jammed and roads that are not potentially jammed. Congestion that occurs on a road will increase stress and encourage disorderly traffic behavior. In addition, the determination of the ETLE sensor allows admin functions to cross-check the system regarding violations that have occurred. A visualization of traffic flow can be seen in Figure 8.

You can figure out how many ETLE administrators there are by using rainbow antimagic coloring. The number of ETLE administrators is the same as the number of connections for rainbow antimagic coloring. Figure 14 shows the chosen path when it comes to the topic of antimagic rainbow coloring. The application of rainbow antimagic coloring in a graph is as follows: (1) performing antimagic labeling on all vertices of the resulting graph representation so that the vertex labels start from one to many vertices in a graph, (2) calculating the weight of the edges by adding the vertex labels connected to the edge, the edge weights will be calculated, (3) analyzing the rainbow path on the graph, with each two vertex on the graph having at least one rainbow path, and (4) if there is the same color (edge weight) in one rainbow path, then repeating

the labeling activity until every two dots have a rainbow trajectory with a different color.

Next, we will perform rainbow antimagic coloring on the traffic representation graph. For example G_A is a graph representation of traffic flows. In this case, we take each road to the vertex label x_i with $1 \leq i \leq 10$. So, we get x_1 as Ahmad Yani Street, x_2 as Trunojoyo Street, x_3 as Hoscokroaminoto Street, x_4 as Gajahmada Street, x_5 as Sultan Agung Street, x_6 as PB. Sudirman Street, x_7 as Seruji Street, x_8 as Hayamwuruk Street, x_9 as Brawijaya Street, and x_10 as Street Slamet Riyadi. Accordingly, $V(G_A) = \{x_i; 1 \leq i \leq 10\}$ and $E(G_A) = \{x_i x_{i+1}; 1 \leq i \leq 6\} \bigcup \{x_8 x_9\} \bigcup \{x_4 x_8\} \bigcup \{x_1 x_5\} \bigcup \{x_1 x_6\} \bigcup \{x_7 x_{10}\}$.

All that is needed to prove that $rac(G_A)$ is 7. We prove a lower bound and an upper bound. Graph G_A contains Tadpole graph $T_{6,2}$ and path graph P_3. Nisviasari, et al., [18] explained that $rac(T_{\alpha,\beta}) = \lceil \frac{\alpha}{2} \rceil + \beta$ if $\alpha \equiv 1, 2(mod 4)$ so $rac(T_{6,2}) = 5$, while $rac(P_n) = n - 1$, then $rac(P_3) = 2$. Based on this, itt was found that $rac(G_A) \geq 7$.

Next, we will prove that the upper bound of $rac(G_A)$ is $rac(G_A) \leq 7$. Given the vertex label function for graph G_A as follows.

$$l(x_i)=\begin{cases} i+6, & \text{if } i = 2 \\ i+4, & \text{if } i = 3 \text{ or } i = 5 \\ i+1, & \text{if } i = 1 \text{ or } i = 9 \\ i, & \text{if } i = 6 \\ i-2, & \text{if } i = 7 \\ i-3, & \text{if } i = 4 \\ i-5, & \text{if } i = 8 \\ i-6, & \text{if } i = 10 \end{cases}$$

Based on vertex label function, we have the edge weight function as follows.

$$w(x_i x_j)=\begin{cases} 4, & \text{if } i = 4, j = 8 \\ 8, & \text{if } i = 1, j = 6 \text{ or } i = 3, j = 4 \\ 9, & \text{if } i = 7, j = 10 \\ 10, & \text{if } i = 1, j = 2 \text{ or } i = 4, j = 5 \\ 11, & \text{if } i = 6, j = 7 \text{ or } i = 1, j = 5 \\ 13, & \text{if } i = 8, j = 9 \\ 15, & \text{if } i = 2, j = 3 \text{ or } i = 5, j = 6 \end{cases}$$

Based on the edge weight function that has been obtained, we have $rac(G_A) \leq 7$. So, we get $7 \leq rac(G_A) \leq 7$ and it complete the proof that $rac(G_A) = 7$. Table 3 show the rainbow path of G_A.

Table 3. The rainbow path from x_i to x_j on G_A graph.

Case	x_i	x_j	Rainbow Path	Condition
1	x_9	x_8	x_9, x_8	-
2	x_i	x_j	$x_i, x_{i+1}, \cdots, x_{j-1}, x_j$	$i < j, 4 \leq i \leq j-1, 5 \leq j \leq 7$ $8 \leq i \leq 9, 4 \leq j \leq 7, j = 10$ $1 \leq i \leq 4, 8 \leq j \leq 9$ $i > j, 2 \leq i \leq 4, 3 \leq j \leq 5$
3	x_2	x_5	x_2, x_1, x_5	-
4	x_1	x_5	x_1, x_5	-
5	x_i	x_{10}	$x_i, x_{i-1}, \cdots, x_6, x_7, x_{10}$	$1 \leq i \leq 3$

After applying the rainbow antimagic coloring, the next step is that the number of connections for the rainbow antimagic coloring will be used as a reference for determining the number

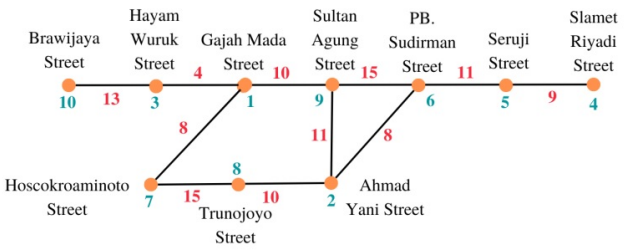

Figure 9. Illustration of Rainbow Antimagic Coloring on the Street Representational Graph

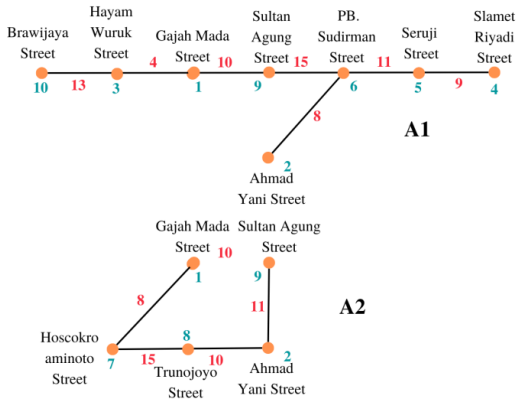

Figure 10. Two Groups of Traffic Flow

of ETLE admins to keep them as minimal as possible. Based on Figure 9, the rainbow antimagic connection number of the graph is 7, so the number of ETLE administrators needed is 7 administrators. Next, a spatial-temporal graph neural network will be applied to determine forecasting regarding congestion in traffic flows.

The graph representation in Figure 9 will be changed to a spanning tree form so that there will be 2 groups of traffic flows, namely the first flow (A1) and the second flow (A2). Each traffic stream will have a different administrator in its path. The function of the administrator is to check the validity of a violation so that violations are not only determined by one party but also cross-checked by other administrators. Figure 10 depicts the two groups of traffic flows.

Based on Figure 10 in group A1, violations that occur on edge weight 13 will be cross-checked by the admin of side weight 4, violations that occur on side weight 4 will be cross-checked by the admin of edge weight 10, and so on. Likewise with the flow of traffic in Group A2. Table 4 is a grouping of two traffic flows according to Figure 15 and the summing elements that make up the edge weights. Table 4 serves to identify sections of the traffic flow that commit violations. For example, in groups A1 and A2, there are the same edge weights, namely 8, 10, 11, and 15. If there is a violation in one of the traffic flows that has the same weight, then to find out the location of the violation, look at the elements that the sum of two vertex labels that makes up the weight of that edge. There are 10 violations that most often occur, namely: running a red light, not wearing a helmet, not turning on the vehicle's lights, not carrying a driving certificate, going against the flow, violating traffic signs, breaking through the busway lane, vehicle users who do not pay attention to safety aspects, not using the mirrors, and driving over the sidewalk.

Discussion graph on Jember's street in this study. This embedding procedure can decrease the initial feature dimension from 5 to 1. Of course, processing message passing and aggregation is necessary to obtain the embedding results. We presume each node holds information or messages that it will forward to other nodes during the message-passing procedure. As a result, we consider the adjacency matrix in this operation. However, we consider more than just a node's connection

Table 4. Traffic Flow Grouping

A_1	A_2
(10, 3)	(1, 7)
(3, 1)	(7, 8)
(1,9)	(8, 2)
(9, 6)	(2, 9)
(6, 2)	
(6, 5)	
(5, 4)	

to nearby vertices.

Additionally, consider how the node is related to itself. Therefore, in this message-passing mechanism, matrix B serves as a reference. We then add up the messages in the aggregation process after they have been sent. We use error value as a benchmark to measure how close two vertices have been processed by aggregation. In this study, the best error value is 1.3006×10^{-4} generated using a weight of 0.1 and 10 iterations.

Furthermore, we forecast the time series data using ANN after obtaining the embedded data. At this stage, there are three parts: training, testing, and forecasting. We used four models and two ANN architectures during the training. In network training, we obtained a model, which was then used in the testing stage. The model that has been built is then tested at the testing stage to measure the performance of the ANN. As a benchmark, we use the mean square error (MSE). Based on the test results, the best ANN model is Cascadeforwardnet with 656 architecture. Furthermore, using the training model, we obtain traffic flow data forecasting on the following day. This forecast is shown in Figure 7.

4 Conclusion remarks

In this paper, we learn about traffic networks as a graph and use spatial temporal graph neural network (STGNN) to forecast traffic flow in Jember city. STGNN is built on the idea of concurrently modelling spatial and temporal interdependence. The spatial-temporal graph structure in STGNN is dynamic because the properties of the vertex or edge might change over time. A common use of STGNN is the analysis of street traffic flow networks where a graph structure can characterize street networks such as vehicle speed, number of vehicles per hour, number of accesses to the street, rain intensity, and crowds which change over time. The results show that the initial weight of 0.1 is the weight that produces the smallest error value, the best model generated by ANN-656 Cascadeforwardnet with a test MSE is 7.3730×10^{-13}, and the forecasting traffic flow shows that most street that has the potential traffic jam is street number 5 or in other words, is Sultan Agung Street.

References

[1] Batyha, R. M. Data Mining-Based Multidimensional Extraction Method for Indicators of Social Security System for eople with Disabilities. *J. Appl. Math. & Informatics* **40** (2022), 289-303.

[2] Bengio Y., R. Ducharme, P. Vincent, and C. Janvin, "A Neural Probabilistic Language Model", Journal of Machine Learning Research, 2003, 3, pp.1137-1155

[3] Bui K. N., J. Cho, and H Yi, "Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues", Applied Intelligence, 2021, pp. 1-12 https://doi.org/10.1007/s10489-021-02587-w

[4] Dafik, F. Susanto, R. Alfarisi, B. J. Septory, I. H. Agustin, and M. Venkatachalam, "On Rainbow Antimagic Coloring of Graphs", Advanced Mathematical Models & Applications **6** 3 (2021), 278-291

[5] Dafik, Q. A'yun, R. I. Baihaki, A. C. Prihandoko, A. I. Kristiana, F. G. Febrinanto, and K. A. Santosa, "The Spatial Temporal Graph Neural Networks and Rainbow Antimagic Coloring for Time Series Forecasting on Flood Flow Anomaly", 2022, In press

[6] Dave V. S. and K. Dutta, "Neural network-based models for software effort estimation: a review", Artificial Intelligence Review, 2014, 42, pp. 293-307 https://doi.org/10.1007/s10462-012-9339-x

[7] Farahani, M. R. A New Vertex-Coloring Edge-Weighting of Complete Graphs. *J. Appl. Math. & Informatics* **32** (2014), 1-6.

[8] Gross J. L., J. Yellen, and P. Zhang, "Handbook of Graph Theory Second Edition", 2014, Florida: CRC Press

[9] https://www.crayon.com/computer-vision (accessed: 9/12/2022)

[10] https://docplayer.info/73251738-Oleh-yakub-dedy-karyawan.html (accessed: 11/12/2022)

[11] https://otomotif.kompas.com/image/2021/03/21/090200015/cara-kerja-etle-mobile-atau-tilang-elektronik-berjalan?page=1 (accessed: 11/12/2022)

[12] https://podcastle.ai/tools/voice-to-text (accessed: 9/12/2022)

[13] Hussain, M. M., and S. K. Devi. Disease Forecast Using Machine Learning Algorithms. *J. Appl. Math. & Informatics* **40** (2022), 1151-1165.

[14] Joedo J. C., Dafik, A. I. Kristiana, I. H. Agustin, and R. Nisviasari, "On the rainbow antimagic coloring of vertex amalgamation of graphs", Journal of Physics: Conference Series, 2022, 2157, pp. 1-12

[15] Kristiana A. I., I. L. Mursyidah, Dafik, R. Adawiyah, R. Alfarisi, "Local irregular vertex coloring of comb product by path graph and star graph", Discreate Mathematics, Algorithms and Applications, (2022), https://doi.org/10.1142/S1793830922501488

[16] Mikolov T., G. Corrado, K. Chen K, and J. Dean, "Efficient Estimation of Word Representations in Vector Space", Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013, pp. 1-12

[17] Mursyidah I. L., Dafik, and A. I. Kristiana, "On Rainbow Antimagic Coloring of Some Classes of Graphs", 2022, pp. 1-15, In press

[18] Nisviasari, R., Dafik, I. H. Agustin, E. Y. Kurniawati, I. N. Maylisa, and B. J. Septory. 2022. Improving the robustness of the affine chipper by using a rainbow antimagic coloring. Journal of Physics: Conference Series - ICCGANT

[19] Saleh, A., Aqeel, A., and H. Alashwali. On Refulmulated Injective Chromatic Index of Graphs. *J. Appl. Math. & Informatics* **39** (2021), 13-29.

[20] Vlahogianni E. I., "Computational Intelligence and optimization for Transportation Big Data: Challenges and Opportunities", Engineering and Applied Sciences Optimization, 2015, pp. 107-128 Springer

[21] Yu B, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting", arXiv:1709.04875v4, 2018, pp. 1-7[1600] D. Wang, H. He, and D. Liu, 2018 "Intelligent optimal control with critic learning for a nonlinear overhead crane system", IEEE Transactions on Industrial Informatics **14** 7 (2018), 2932-2940.

[22] Zhou J., G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph Neural Networks: A review of Methods and Applications", AI Open, 2020, 1, pp. 57-58 https://doi.org/10.1016/j.aiopen.2021.01.001

[23] Alfarisi, R., Prihandini, R.M., Adawiyah, R., Albirri, E.R. and Agustin, I.H., 2019. Graceful chromatic number of unicyclic graphs. *Journal of Physics: Conference Series* (Vol. 1306, No. 1, p. 012039). IOP Publishing.

[24] Dafik, Agustin, I.H., Surahmat, Alfarisi, R., and Sy, S. 2017. ON NON-ISOLATED RESOLVING NUMBER OF SPECIAL GRAPHS AND THEIR OPERATIONS. Far East Journal of Mathematical Sciences, 102, 2473-2492.

[25] Septory, B.J., Utoyo, M.I., Sulistiyono, B. and Agustin, I.H. 2021. On rainbow antimagic coloring of special graphs. In Journal of Physics: Conference Series (Vol. 1836, No. 1, p. 012016). IOP Publishing.

[26] gustin, I.H., Hasan, M., Adawiyah, R., Alfarisi, R. and Wardani, D.A.R. 2018. On the Locating Edge Domination Number of Comb Product of Graphs. Journal of physics: conference series (Vol. 1022, No. 1, p. 012003). IOP Publishing.

[27] Gembong, A.W. and Agustin, I.H. 2017. Bound of distance domination number of graph and edge comb product graph. Journal of Physics: Conference Series (Vol. 855, No. 1, p. 012014). IOP Publishing.

Author information

Ika Hesti Agustin, Department of Mathematics, PUI-PT Combinatorics and Graph, CGANT, University of Jember, Indonesia.
E-mail: ikahesti.fmipa@unej.ac.id

M. Venkatachalam, PG and Research Department of Mathematics, Kongunadu Arts and Science College, Coimbatore-641 029, Tamil Nadu, India.

Indah Lutfiyatul Mursyidah, Rifki Ilham Baihaki, PUI-PT Combinatorics and Graph, CGANT, University of Jember, Indonesia.

Marsidi, Department of Mathematics Education, Universitas PGRI Argopuro Jember, Indonesia.