# Structure based Similarity Label Propagation on Electric Vehicle data

***Abstract*—The Electric Vehicle (EV) is taking over the market step by step. People choose to replace fuel vehicles with EVs for multiple reasons. With the increasing retention rate of EVs, Electric Vehicle Supplement Equipments (EVSE) are also being deployed in a wider area and a charging point is one of the most important equipment among them. Our research tries to propose a method that can help in the deployment of charging points in a semi-supervised learning scenario. The method is based on a graph whose node represents a single charging poin, and the graph captures the structural similarity between nodes. We noticed that label propagation is an efficient method to fulfill semi-supervised learning tasks. The traditional label propagation method extracts the weights of edges to identify the similarity between nodes, so a node's label tends to be decided by the label distribution of its neighbors. However, in realistic problems, distant nodes sometimes share similar structural identities and belong to the same community. For this reason, we propose a structure-based label propagation method that identifies structural similarities between distant nodes. Our experiment shows that this method successfully extracts structural similarity from even distant nodes.**

***Index Terms*—Label Propagation, structural identity, semi-supervised learning**

## I. Introduction

Many people have been turning to electric vehicles in recent years as their primary mode of transportation. The low fuel cost and high-tech driver assistant system make the EV competitive in the market. Along with the popularity in the market, the corresponding technologies also developed rapidly. Some autopilot systems enhanced by computer vision techniques can park themselves in the parking lot. Meanwhile, motivated by battery technology, electric vehicles keep getting more mileage. Also, EV can deeply incorporate the concept called 'digital twin' [1]. However, compared to refueling a traditional fuel vehicle, it takes a much longer time to charge an electric vehicle. The situation makes it essential to design an efficient charging network. Several previous works have worked on the problem of building an efficient network [2], [3], [12].

We address the problem of using a similar power scheme for similar chargers. Geometrically, the location of the chargers can be easily abstracted into a graph. As shown in fig 1, we picked one of our datasets, the NCR data, as an example. All the blue and red points marked on the figure specify the location of EV chargers. We separate them into two categories according to whether they are inside London city or only in the Great London area but not in London city. Moreover, we zoom in on the top-right corner of the figure with constructed graph built by the $k$NN method. We can see that the chargers in the two categories show apparent differences in characteristics. The distribution of chargers inside London city is much denser than those in the countryside. Along with the increasing density, the degree of the nodes also increases. That is an interesting observation that can be identified on the structural level. Once we extract the topology of the geometric information to build the graph, the nodes belonging to the blue category will have much more neighbors. Our proposed method then captures the structure information and separates the two categories in a semi-supervised learning scenario.

Label propagation is an efficient and powerful clustering method based on graph theory and been used for semi-supervised learning tasks []. The traditional label propagation method [16] values the distance between nodes. The label information will be first transformed to its closest neighbors. So, generally speaking, a node will be labeled according to the most similar nodes to it, which are decided by the ratio and distance of surrounded nodes. However, we noticed that the structure information also plays an essential role in deciding the similarity between nodes [11]. Also, [9] finds that the structure identity helps to identify individuals in a social network, which is similar to our goal. Struc2vec [10] makes good use of the structure information, which inspires us to come up with a new method of label propagation that allows labels to transform into distant neighbors with similarities in structure identity.

[1]

Overall, our contributions are as follows:

- We proposed a new method to build structure sensitive similarity graph for label propagation so that the label can be propagated to a distant neighbor.
- We perform our method on a toy example and several EV datasets—the performance overpass the original label propagation method.

## II. Related Work

### A. Graph Construction

Graph construction would always be the first step to fulfill for a graph-based semi-supervised learning framework. Graph construction aims to build a similarity graph that can describe the relationship between original Euclidean samples for further graph method processing. We usually treat samples from the original dataset as nodes in the similarity graph; meanwhile, the edge of the similarity graph will represent the distance

[1]https://github.com/ReallyMonk/clusterGNN-ev-label-propogation.git

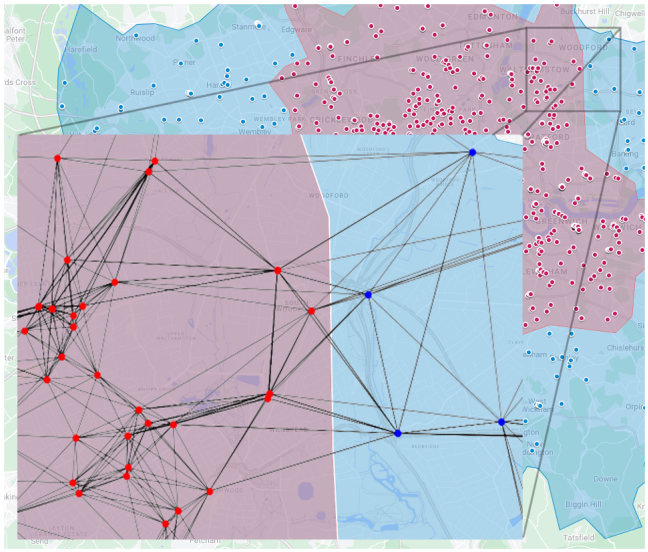


Fig. 1. The figure locates the exact locations of chargers inside the Great London area. Each point in the figure indicates a charger in the real world. And we zoom in to show a part more clearly to illustrate the structure similarity we try to identify. In the graph, we zoom in the top-right corner of the original graph, which we believe shows the differences between the two classes more clearly.

between each pair of corresponding samples. To build a similarity graph, we must carefully pick two important components – a measurement of corresponding nodes and a method to sparsify the graph. L2 distance (also known as Euclidean distance) is the most widely used distance measurement, and it shows excellent stability and performance. Meanwhile, cosine distance and chi distance are good choices for distance measurement. They show good performance on text classification and histograms, respectively. Graph sparsification plays an important role in increasing the robustness and efficiency of further calculation. Neighborhood methods like k-nearest neighbors graph and e-distance graph work great in building the unregular graph. These methods are efficient and simple to use. An alternative method to neighborhood methods is based on the b-matching algorithm. A typical attribute of the b-matching method is that the generated similarity graph is exactly b-regular (each node has an exact b degree). In [6], Jebara and Huang show that regular graphs can achieve better performance on classification graphs. However, our method measures the similarity through a sorted degree list of nodes' neighbors, so it values the differences in the degree between different nodes.

### B. Struc2Vec

Struc2vec [10] is a framework proposed by Leonardo. It learns the latent representation from the structural identity of nodes, which is addressed by the relationship between the node and its neighbors. Struc2vec introduced a hierarchy to identify the structural similarity between nodes. The method will first generate an ordered sequential list that indicates the degree of neighbors in different hops for each node. Then they leverage the Dynamic Time Warping method to measure the distance between two sequences as the weights of edges between corresponding nodes. This process will repeat $d$ times to build a $d$-layer similarity graph, where $d$ is the diameter of the original graph, and the $k$th layer of the graph describes the relationship between nodes on its $k$-hop. After building the similarity graph, the method follows the DeepWalk method to generate walks to learn the structural identity of nodes. In our work, we value the thought of building a multilayer similarity graph to describe the structural information of nodes.

### C. Label Propagation

Label Propagation was proposed in 2003 by Zhu [16]. People noticed its efficient performance in the scenario where only a small part of the data was labeled, so it became widely used in semi-supervised learning tasks. Later works keep digging into the potential of the label propagation algorithm. [13] researched the relationship between label propagation and graph convolutional neural networks (GCN). [5] found that by well-tuning the label propagation algorithm, it can show the comparative performance with GCN.

The label propagation algorithm [16] is designed based on the graph, and many people have used it to work in the semi-supervised learning tasks [4]. We will initialize a graph that includes both labeled and unlabeled data and represents the relationship with edge weight. We believe that higher weights indicate stronger connections between nodes, so the label tends to propagate through those edges. In the algorithm, we will define a transform matrix $P_{ij} = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}}$ to decide which label should be propagate to its neighbors. And a soft label matrix $F$ concats labeled and unlabeled nodes. The soft label can be propagated step by step to its neighbors by matrix multiplication. Then the propagation process will stop when convergence.

## III. Problem Formulation

In this work, we assume there is a list of potential locations to be labeled and only a small part of them have already been labeled so that we work in a semi-supervised learning scenario. Fig 2 shows a complete process of our method. After obtaining the input locations, we extract the useful information to build the original graph with a normal method like $k$NN. The graph here is like the 'digital twin' of the chargers network in the real world. It describes the topology characteristic of the original dataset.

We note the labeled dataset $D_L = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_l, y_l)\}$, where $Y_L = \{y_1, y_2 \dots, y_l\} \in \{1, 2, \dots, C\}$ are the labels. And the unlabeled dataset be noted $D_U = \{(\mathbf{x}_{l+1}, y_{l+1}), (\mathbf{x}_{l+2}, y_{l+2}), \dots (\mathbf{x}_{l+u}, y_{l+u})\}$, where $Y_U = \{y_{l+1}, y_{l+2}, \dots y_{l+u}\} \in \{1, \dots, C\}$ is unknown. Meanwhile, there would be a significant difference between the amount of labeled and unlabeled data, that $l \ll u$. Then the final object is to fulfill a transductive learning process, which is to estimate $Y_U$ from $D_L$.

## IV. Proposed Method

In a traditional label propagation method, the label information will propagate to all nodes' closest neighbors iteratively

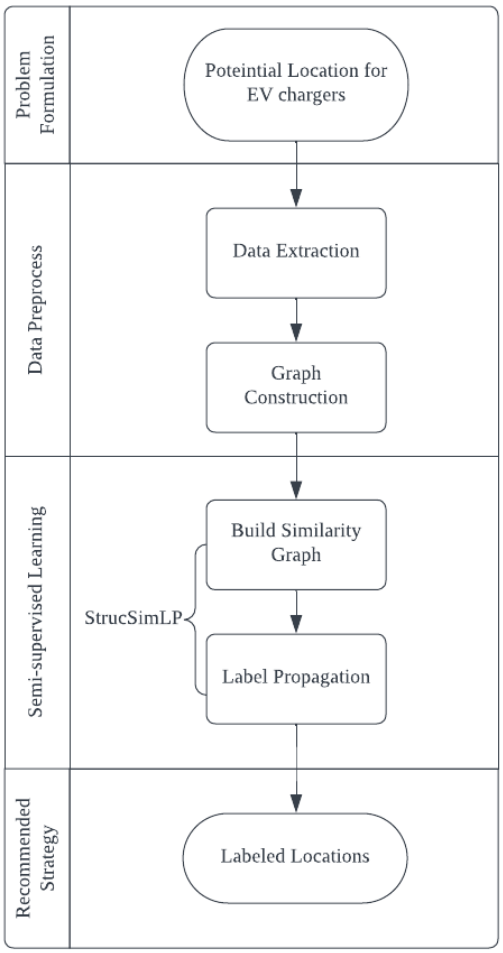



Fig. 2. Process of the structure-based similarity label propagation

and spread to the whole graph like a water wave. This nature indicates that the distance between a pair of nodes plays the most critical role in the propagation process. Correspondingly, the influence of a node on its distant neighbors would be relatively small. Furthermore, this character almost prevents two distant samples been labeled the same. However, a node in a graph does not only obtain feature information but also structure information, which helps to identify similar nodes in a graph.

The structural similarity of nodes in the graph has been noticed for a long time. [11] defined the concept of structural equivalence as two nodes are related in the same way to the same other points. Following this concept, Rebeiro proposed a method called struc2vec [10] that extracts the structure information of a graph. Rebeiro believes that a latent representation of structure identity should not depend on any node or edge attributes. This invoked the author to represent structure information with node degree information. The team uses dynamic time warping (DTW) to calculate the distance between two nodes' degree sequences in the $k$-hop neighborhood as the similarity of this pair of nodes on the $k$-th layer of the similarity graph. Inspired by this work, we try to make good use of structural information to rebuild the similarity graph and perform label propagation on it.

**Algorithm 1:** We Propose a structure sensitive method for label propagation.

**Input:** original kNN Graph $G = (V, E)$, the corresponding adjacency matrix $A$, node labels $Y$, layers number $L$,
**Output:** Y

**//Calculate Compressed Similarity Graph:**

**for** *each nodes* $v \in V$ **do**
  **for** *each layer* $k \in L$ **do**
    $R_k(v) \leftarrow$ calculate_k_hop_degrees$(v, k)$
  **end**
**end**
**for** *each nodes i, j* $\in V$ **do**
  **for** *each layer* $k \in L$ **do**
    $f^k_{i,j} \leftarrow g(R_k(i), R_k(j))$
  **end**
  $\bar{A}_{ij} \leftarrow \epsilon_k e^{(-f^k_{i,j})}$
**end**

**//Label Propagation:**

$P \leftarrow \bar{a}_{i,j} / \sum_{l=1}^{n} \bar{a}_{l,j}$
**repeat**
  $Y \leftarrow PY$
  reset propagated labels
**until** *convergence*;

### A. Graph Construction

Assume the original Euclidean dataset is given by $(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, y_{l+1}), \ldots (\mathbf{x}_{l+u}, y_{l+u})$. Then the feature sets can be extracted to $X = \mathbf{x}_1, \ldots \mathbf{x}_{l+u}$. A typical graph construction method consists of distance calculation and garph sparsification. In our method, we choose to use the combination of Euclidean distance and k Nearest Neighbor sparsification. Euclidean distance reflects the length of the line segment of two points in high dimensional Euclidean distance. It is a commonly used distance measurement and can be calculated by performing l2-norm on vector abstraction.

$$dist(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} ({a_i}^2 - {b_i}^2)} \tag{1}$$

Distance measurements operated on each pair of samples provide us with a complete graph whose edge weights represent the distance between nodes. k Nearest Neighbor method connects a node to its k-nearest neighbors. The sparsification can reduce the computation complexity and allow us to create a sparse graph that represents the structure information more clearly. Since we need both degree values and the number of neighbors to calculate structure similarity in further steps,

a complete graph with all nodes sharing the same degrees will erase the information carried by the degree list. Our further experiments showed that graph sparsification creates a different distribution on the length of graph paths, which is important to decide view. And by adjusting the k-value, we can control the distribution of graph paths.

### *B. Rebuild compressed similarity graph*

The first step to rebuilding the similarity graph is to calculate the similarity between each pair of nodes. We have already built the basement similarity graph $G$ through the original Euclidean data. In this process, we will extract the structure information from the basement similarity graph. Our purpose is to generate a single-layer similarity graph representing the structure similarity between nodes, but we will hierarchically calculate the k-hop of neighbors through k=0. $R_k(u)$ collects the degree of all nodes at the exact distance $k$ to $u$ in $G$. When $k = 0$, $R_k(u)$ represents the degree of $u$ itself. And we usually set the upper bound of $k$ to be the diameter of the graph so that we can collect all neighbors of a node in $G$.

To obtain sorted degree sequence of nodes' $k$-th neighbor, $degree()$ calculate the node degree in $G$, and $d(u, v)$ calculate the distance between nodes $u, v$ in $G$

$$R_k(u) = \{degree(v)|v \in V, d(u, v) = k\} \tag{2}$$

The degree sequences of nodes will have different lengths. A normal distance measurement like the l2-norm can not be used in this situation since it requires two vectors of identical length. Struct2vec uses a classic algorithm for calculating the similarity between sequences with different lengths called DTW. It shows excellent performance on the graph embedding task and has been modified to be more efficient for the struc2vec algorithm. However, we notice that DTW is insensitive to the length of a sequence, which means the algorithm will ignore the number of neighbors reflected by the length of degree sequences. Fig 4 shows a situation that confuses the DTW algorithm. Subgraphs on four branches of the graph are a 3-regular graph, so the degrees of nodes on those subgraphs are all 3. The only difference is the number of two pairs of subgraphs. DTW ignores the difference in nodes number and believes two different subgraphs should have the same label. Based on this observation, we fill the shorter sequence with zeros and then use l2-distance to calculate the similarity. This algorithm rebuilds the sensitivity to the scale of a subgraph.

Here we assume $|a| > |b|$, $g(a, b)$ calcualtes the similarity between node $a$ and $b$

$$g(a, b) = \sum_{i=1}^{|b|} ||a_i - b_i||_2 + \sum_{j=|b|+1}^{|a|} ||a_j||_2 \tag{3}$$

For now, we obtained $k+1$ layers of similarity graphs. The calculation complexity will increase rapidly if we construct a multilayer similarity graph. The label propagation would be forced to execute on a $kN \times kN$ matrix, so we decided to compress the multilayer graph into a single layer. The key point in this process is to decide the weights assigned for different layers. Taking the average of each layer could be harmful in representing the similarity of node pairs. Our further experiment on Fig 5 shows that aggregating all layers with the same weight will erase the information carried by important layers. However, if we carefully choose which layer to focus on, the performance can be improved significantly. We found that the ability to identify a structure of a subgraph is bad with an improper $k$. We find this mechanism works similarly to the 'view' of an eye. A person loses the ability to distinguish the differences between subparts when looking at a range with few differences from other parts. So we introduce a set of parameters and call it 'view' to decide which layer to focus on and how much to care about the adjacent layers.

Here we choose to use the probability distribution function of normal distribution to calculate similarity between two nodes on $k$-hop neighbors represent by $\varepsilon$. Two parameters $\tilde{k}$ and $\sigma$ represent the focused layer and view range, respectively.

$$\varepsilon_i = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(i-\tilde{k})^2}{2\sigma^2}}, i = 0, 1, \dots k \tag{4}$$

Then the similarity within each layer $k$ between nodes $u$ and $v$ calculated by $f_k(u, v)$, then aggregate the layer-wise similarity with view configure $\varepsilon$ to obtain the final similarity on compressed similarity graph. Meanwhile, we would like the edge weight to be large when two nodes are similar to each other, so that the label information would be more likely to propagation to similar nodes. To achieve this, we perform exponential operation on the similarity value to calculate edge weights in adjacency matrix.

$$f_k(u, v) = g(R_k(u), R_k(v)) \tag{5}$$

$$\bar{A}_{uv} = \sum_{i=1}^{k} \varepsilon_i \text{exp}(-f_i(u, v)) \tag{6}$$

The above steps fulfill the process of building a similarity graph. It is a complete graph, so the label propagation process will reach stability really fast. And compared to a multilayer graph, this compressed graph can save a lot of computational resources. Meanwhile, by controlling the view size, we can set a smaller $k$ to reduce unnecessary layers. Further experiments also show that layers with large $k$ barely carry important information that helps to identify structure patterns.

### *C. Decide view*

The view is an important setting in our method. It decides which layer of neighbors to focus on and how much we should care about the other layers. View is important to maintain the performance of the whole system. Setting the view in an inappropriate range will force the algorithm to focus on the neighbor's ring without any differences, which confuses the algorithm in deciding the similarity between two nodes. Our experiments found that the choices of view are closely corresponding to the distribution of the length of the graph path. Fig 6 shows that in our toy example, there will always be two peaks in the distribution, and setting view value around

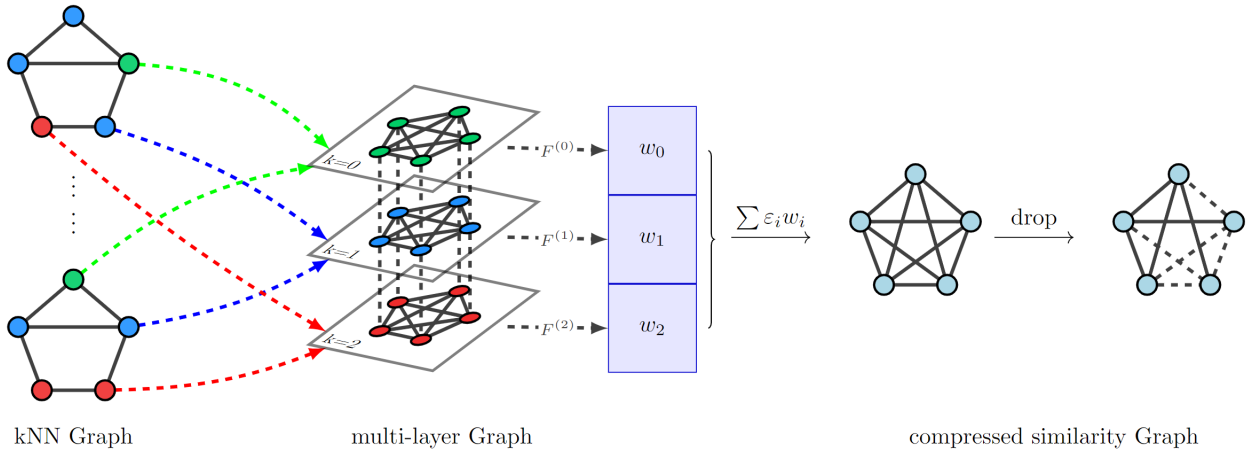
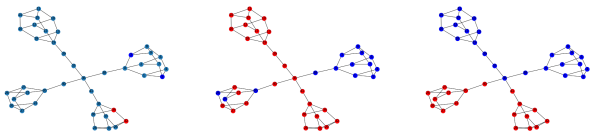
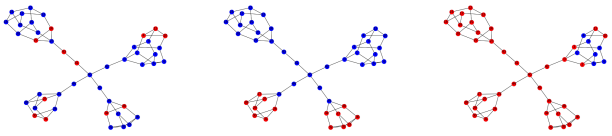



Fig. 3. shows the process of rebuilding the compressed similarity graph from the original kNN similarity graph. The first part of the figure explains how to extract the corresponding degree lists from the kNN similarity graph. The nodes were colored blue, red, and green to represent the node's 1-hop, 2-hop neighbors, and the node itself, respectively. The distance between the degrees of these nodes decides the edge weights of the corresponding layers. Then scaled by the hyperparameter layer weights, these distances will be compressed to generate a single layer as the adjacency matrix of the compressed multi-layer graph, where $\varepsilon$ is a preset hyper parameter. The final part of the graph describes a dropout process to remove the edges with lower importance. Hence we build a similarity graph that extracts the structure information.



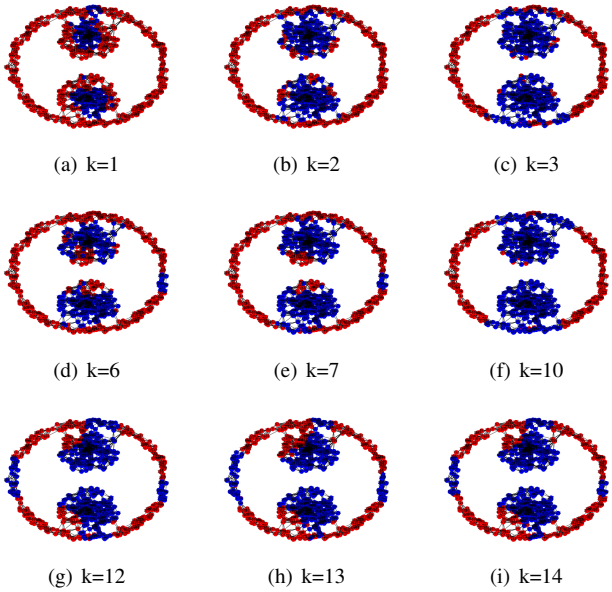
(a) Initialization (b) DTW (c) pad-norm

Fig. 4. left figure shows the initialization of the test graph. This graph consists of two pairs of 3-degree graph with 8 and 10 nodes and we set two different labels to each pair, respectively. The rest two figures show the result of label propagation based on similarity calculated by DTW. Figure in the middle shows the DTW failed to distinguish the differences between node numbers. Meanwhile, we can see that fill-zero strategy managed to obtain this information.



(a) k=2 (b) k=7 (c) k=10

Fig. 5. shows the result by setting k=2,7,10 respectively from left to right. It clearly shows that a proper view selection can improve the performance significantly



(a) k=1 (b) k=2 (c) k=3

(d) k=6 (e) k=7 (f) k=10

(g) k=12 (h) k=13 (i) k=14

Fig. 6. The group of figures shows the prediction result of label propagation result. The graph is generated from designed data after graph construction process. The observation shows that prediction shows clearly discrimination when k falls around 7 and the performance reduce rapidly after k is larger than 10

the layer where the first peak occurred can always show better performance. The nodes in two circles generate the peak because those nodes' inner connection is denser than the outer ring. Meanwhile, this peak represents the layer that maintained the most abundant structure information.

### *D. Label Propagation*

The label propagation process consists of performing the label propagation algorithm on two different graphs. The first graph is the compressed similarity graph, and the other is the original distance graph. As mentioned in the last section, the compressed similarity graph extracts the structure

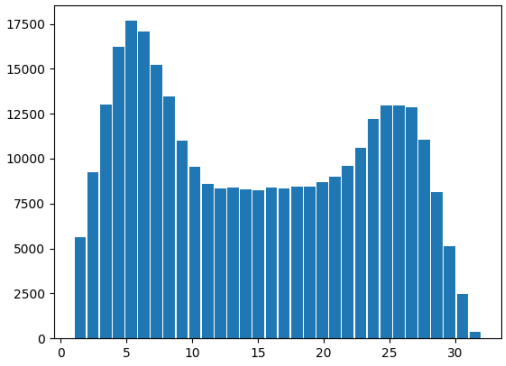



Fig. 7. shows the distribution of graph paths length. For the graph above, we can see that there are two peaks in the graph show that vast paths length accumulated around k from 6 to 10. And this corresponding to the performance of different view value.

information as edge weights so that the distance relationship would be completely different. From the perspective of the original graph, the propagation process on the compressed similarity graph might be like a jump between two distant nodes. This property allows the propagation to execute with ignorance of distance limitation. And another property is that the compressed graph is a complete graph, so the propagation process converges in just a few iterations. The first round of label propagation algorithm helps to set pseudo-labels for unlabeled nodes. However, a weakness in this process is that focusing on the structure information may create some noise in the neighborhood. To solve the noise problem, we perform another label propagation algorithm on the original graph with pseudo-labels generated. This process can further improve the performance of splitting clear communities and strengthen final performance stability. Our further experiments show that label propagation on the compressed similarity graph can provide the expected result for most unlabeled data, and the following label propagation on the original graph with the pseudo labels can extract the distance similarity to correct the mislabeled noise.

$$P_{ij} = \frac{\bar{A}_{ij}}{\sum_{j \in v} \bar{A}_{ij}} \tag{7}$$

$$\bar{Y} = PY \tag{8}$$

## V. Experiment

### A. Dataset

TABLE I
DATASETS PRESETTING

| Dataset | Data Type | Dim($n$) | Samples($N$) | Classes($c$) |
|---|---|---|---|---|
| Toy Example | Artifical | 2 | 2000 | 2 |
| NCR data | Real-world | 2 | 1871 | 2 |
| ACN data | Real-World | 125 | 504 | 3 |

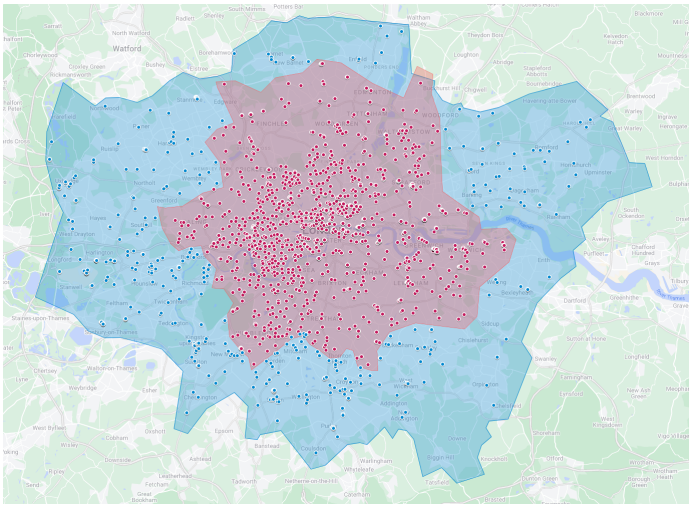


Fig. 8. The figure shows the distribution of the NCR dataset. The blue area shows the Greater London and its included nodes. Meanwhile, the red area shows the distribution of the nodes in London city and its included nodes.

**Toy Example** For the toy example, we designed a pattern with three independent parts shown in Fig 9 (a). The pattern consists of two inscribed circles surrounded by a large ring. In our design, the two circles share a similar structure identity, which is different from the structure identity held by the outer ring. So, we labeled the circles and the ring into different clusters according to their structure identities. In our experiments, we randomly generated 800 nodes inside the three areas.

The toy example here was built to have a clear structure identity. The two circles at the top and bottom of the graph have a much more dense node distribution compared to the outer ring. The distribution of path length Fig 7 shows two peaks, which were generated from the two circles inside. We noticed that the whole graph was connected together through the outer ring so that the traditional label propagation method could propagate the label information through the ring. This situation makes it almost impossible to propagate label information from the bottom circle to the top circle for the traditional method without initializing labels inside the top circle.

**National Chargepoint Registry (NCR) data**

NCR is a public dataset collected by the UK government to record the specific information of the registered charger around the UK. Since the dataset collected chargers from all over the country, it includes more than 20000 chargers' information. An entity in this dataset contains the location and specification of a charger. The location information consists of two parts. One is the coordinates indicated by longitude and attitude, and the other is the postal address of the same location from street to the county. Accurate addressing information can benefit the research of location arrangement for chargers or power grid for chargers network. The charger specification indicates the connector output rated watt, voltage, current, and technical configurations. The information can work as a feature

on the charging network, where power information can help to build the power grid network. All the information In our research, we focus on the location information indicated by longitude and latitude and build the graph according to the distance between nodes based on $k$NN method. We can have an overview in Fig 8, which shows all nodes we used for our experiments. On the graph, we can see that the nodes of the full dataset were separated into two categories. We also draw the area into two categories belongs to, where all the red nodes are in the London area and all the blue nodes are inside the Greater London area and outside the London area. To build the original graph that describes the topology, we use the $k$NN method with $k$ value set to be 11.

For the labeling process, we separate two labels according to whether the location is in London or Greater London. As shown in the figure, the chargers located in the Greater London area show a much more sparse structure than those in the London city, which we expected our method to explore.

**Adaptive Charging Network (ACN) data** ACN [7], [8] is a dynamic dataset recording charging sessions from three locations. The first one is Caltech University, which has 54 EVSE( Electric Vehicle Supply Equipment)s. The site is open to the public but still used mainly by faculty and students. The other two sites are JPL and an office building. The former is a national research lab located in Canada, and the latter is an office building located in Silicon Valley. They have 58 EVSEs in total and are only open to employees. In general, the dataset records all charging events at the three locations. Most users can use the charging device as they wish, while only JPL provides a schedule for their employees. We believe we can expect a certain pattern for their charging data. Each entity from the ACN is a single charging session, including useful information like the origin of the EVSE, the connection/disconnection time of the session, the charging rate during the session, the userID of who carried the charging behavior, etc.

ACN dataset collects data by time so that the complete dataset is multivariant time series data. Meanwhile, the charging operation is generated by human wish, so time-series data is usually unevenly spaced on the timeline. To construct the graph with $k$NN method, we will convert the time series data to matrix data with Gramian Angular Field (GAF) [14], which is a time series encoder that extracts the advantages of Gram Matrix. The entries of a Gram matrix are given by the inner product of vectors, which is designed to describe linear dependencies between a set of vectors. In GAF, we first scale a single time series onto $[-1, 1]$ to fulfill the domain of $acrcos(\cdot)$. Then we convert the time series data from Cartesian coordinates to polar coordinates. We represent the angle in polar coordinates with arccos of value in time series and the radius in polar coordinates with timestamps. With the growth of the time axis, the coordinates follow a trace encircling and leaving away from the original point. This conversion maintains both the temporal and value information of the time series. To generate the Gram matrix, we calculate the elements with the addition of the angles under the cosine function.

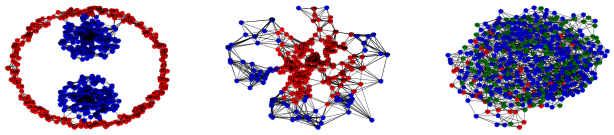
TABLE II
CAPTION

| | Toy | NCR data | ACN data |
|---|---|---|---|
| Compress | 0.8950 | 0.9027 | 0.5873 |
| Traditional | 0.6112 | 0.8620 | 0.4682 |

Based on GAF, [15] proposed a method for multivariant time series. Since GAF converts time series into images, [15] proposes to append multi images together as a large image for the CNN network to capture multivariant information.

However, the GAF method requires the input time-series data to be evenly spaced to maintain the rows and columns index corresponding to the timeline. Based on our observation, we found that the output samples would be evenly spaced if we scaled the time clip up to a week. So we convert the unevenly spaced time series to evenly spaced week-distributed time-series data. During the conversion, we also aggregate some original data to generate new features. Our final well-adjusted time series data preserve five attributes and consider charge station ID as labels. The original dataset contains the exact time of plugging in the charger, disconnecting the charger, and charging. We convert this timestamp information to the time duration of how long the charger has been occupied and how long the charger charges the EV. Then we group the time series data according to the week and station ID to address the total session, and user amounts in a week. With the same grouping, we calculate the sum of the kWh delivered to address the total energy provided by a specific charger in a week.

As mentioned, the ACN dataset has been transformed from time sequence data to image data. Performing label propagation on this dataset is hard because it is a mixture of three communities without a clear boundary. Fig 9 shows that the distribution of the normal method result after propagation compare to our method almost performs a random label. The result shows that our method tries to distinguish the labels from outer to inner areas. Fig 10 shows that the rebuilt similarity graph revealed some latent space patterns and managed to reflect on the original graph.

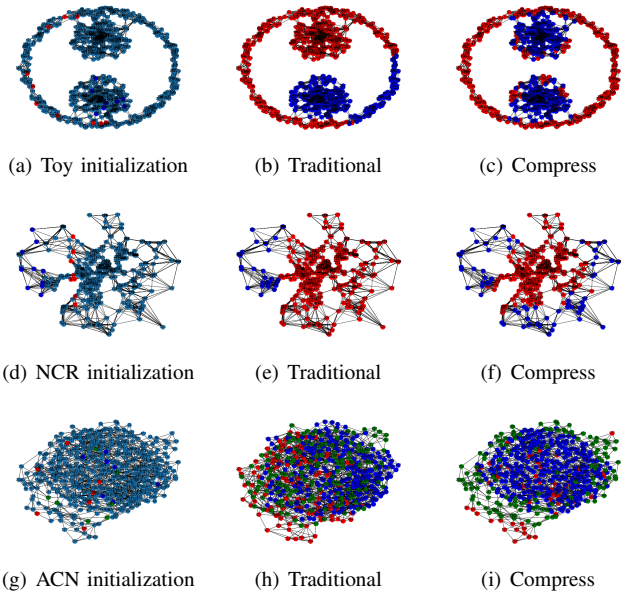


(a) Toy example (b) NCR (c) ACN

Fig. 9. shows the three graphs built on the three datasets with kNN graph construction method.

### B. Results

We can have an overview of the dataset in Table I and the original node distribution in Fig 9. In the experiment setup, we randomly initialize a part from the original datasets. And to reveal the advantage of our method, we pick a group of special initialized nodes that are located unevenly across the whole graph. Like in the toy example, we tend to pick the



(a) Toy initialization (b) Traditional (c) Compress

(d) NCR initialization (e) Traditional (f) Compress

(g) ACN initialization (h) Traditional (i) Compress

Fig. 10. shows the result of the initialization and the final result of the two methods.

initialization that none of the nodes in the top circle were labeled so that only the structural information can help to identify the similarity between two circles. And for NCR data, we picked the initialization where only the left part nodes were labeled so that the dense community inside would separate the outer nodes with sparse structure. But for ACN data, since the nodes do not share a clearly observed structure among communities, we just randomly initialized seven nodes for each community.

The result of our method performed on each dataset is shown in table 10 and in Fig 10. The table shows that our method performs better on every dataset. Furthermore, we can have an overview of how the final results distribute in Fig 10. The left column of the datasets shows the initialized situation for each dataset, and we can have a brief knowledge of how the labeled nodes were distributed in the first place.

In the toy example and the NCR dataset, the blue nodes were geometrically separated by red nodes. The rest two columns show the propagation results from the traditional label propagation method and our propagation method, respectively. We can see that in the toy example, the label information propagates through the right or left path of the ring. However, there is no direct connection between the two circles. The situation blocks the traditional label propagation method from propagating the blue label to the top circle. Also, at the beginning of the propagation process, some nodes on the right corner of the ring were labeled blue and formed a strong blue community. This stable community is hard to be corrected even till convergence. Meanwhile, we set our method to have the view of 2-hop so that the method can identify structural similarities between two nodes. Our method propagates on the compressed similarity graph, so there are strong connections between nodes in the two circles since they share a similar dense structure compared to nodes on the ring. By leveraging the advantage, our method manages to propagate the label information across the two circles. This movement significantly improved the performance of the toy example.

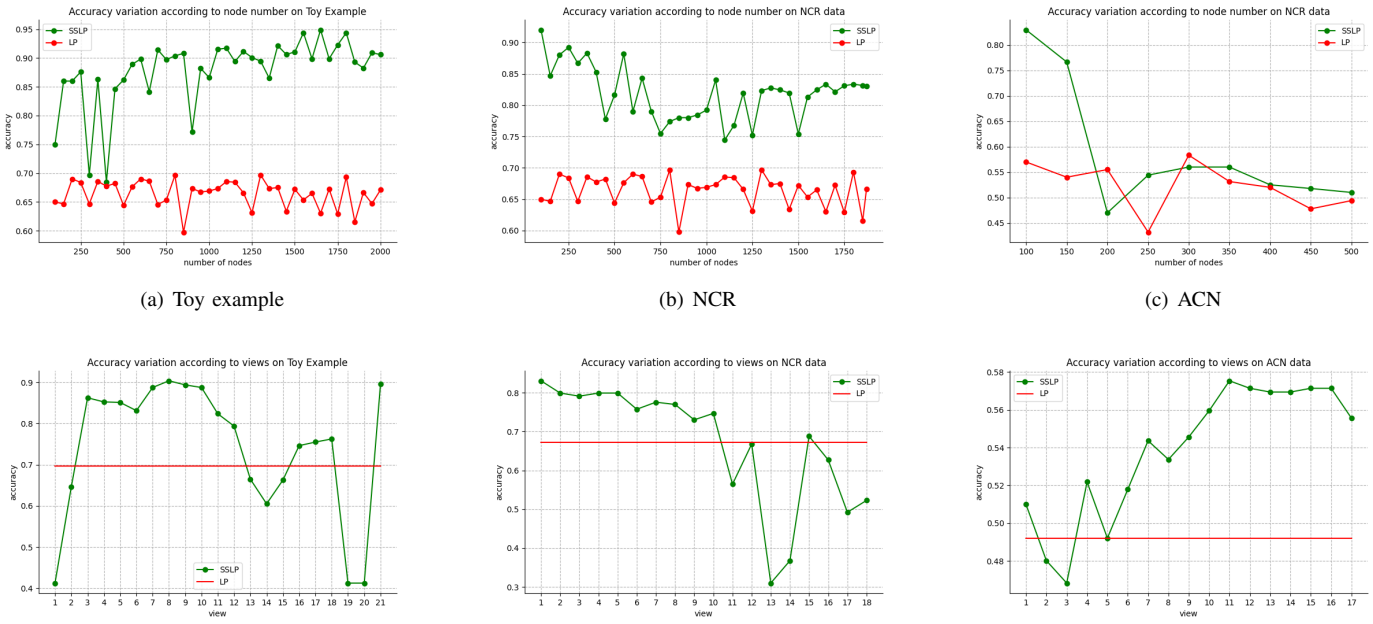
A similar situation also happens in the NCR dataset since we separate the Great London area and London city. The chargers located around the Great London area have a much sparser structure compared to those inside the London city, which is the structural information we expect our method to identify. The traditional propagation process can not overpass the barrier created by the dense nodes inside. We can see in Fig 10 that all blue labels were blocked in the left part of the graph, and red labels take over most of the nodes. Here we can also identify another problem in the traditional label propagation method the unevenly initialized labels will make other communities grow so fast to occupy more nodes, and a strong community will be hard to correct. Otherwise, our method managed to propagate the blue labels around the London area and extract most of the nodes with the sparse label. We set the view to focus on the 2-hop neighbors and support with information from other layers. The 2-hop neighbors' degree list maintains much information in latent space that can work to distinguish the differences between the two communities.

Compared to the two previous datasets, the nodes in the ACN dataset are almost mixed together and barely show visible structure in low dimensions. Fig 10 shows the final result of the traditional method. We can see that because of the complex edges among nodes, the label information quickly penetrates other communities. The propagating process is also unstable, so we choose an early-stop strategy that we end the propagation process several epochs after all nodes are labeled. Meanwhile, our method takes the same strategy to complete the propagation process. The result in Fig 10 shows that even under the mixture of different communities, our method still tries to identify the structure information among each community. The final result shows a wave pattern from the outer nodes to the core. And this pattern does improve the accuracy.

On the whole dataset, we run several rounds of tests to show the accuracy varies according to several augmentations, as shown in Fig 11. We run experiments on different node numbers and views.

For the experiments about node numbers, we set the interval to be 50, and the number starts with 100 nodes. We will randomly draw specific numbers of nodes from the dataset to perform our method and traditional label propagation for comparison. Since all three datasets have different node numbers, the upper bound of the node number is the number of nodes in the dataset. Then to run our method, we pick the view that performs best on the dataset. For the experiment about views, we run all tests on the full dataset. And have the traditional label propagation as the baseline.

On the toy example and the NCR dataset, which shows more clear structural identity between classes, our method clearly




(a) Toy example (b) NCR (c) ACN

(d) Toy example (e) NCR (f) ACN

Fig. 11. shows the accuracy variations according to node numbers and view. The first row of the figures shows the accuracy varies according to increasing sequential node numbers. The second row shows the accuracy varies according to different views. And each column was run on the same dataset.

overpasses the traditional label propagation on a different number of nodes. And for ACN data, which only shows unseeable structural identity, our method only ahead a little on the final accuracy. But the result view shows that, by moving the focusing range of structural information, our method can manage to locate a more accurate range to improve the overall performance. The toy example and the NCR data also show similar results that if we choose the view carefully, the performance can be improved a lot.

## VI. Conclusion

In this work, we introduced the structure-based label propagation method. It aims to identify the structural information of nodes to fulfill the semi-supervised learning task on graphs. Compared to the traditional label propagation method, which prefers to propagate label information to its closest neighbors, our method can identify structural similarities between distant nodes and propagates label information to disconnected nodes. In the research, we also see the potential of measuring similarity based on structural information. It carries latent information from the relationship between nodes and the structure of the subgraphs. We believe that such measurements contribute massively on revealing latent information in graphs.

## References


[1] G. Bhatti, H. Mohan, and R. R. Singh. Towards the future of smart electric vehicles: Digital twin technology. *Renewable and Sustainable Energy Reviews*, 141:110801, 2021.

[2] N. Chen, C. W. Tan, and T. Q. Quek. Electric vehicle charging in smart grid: Optimality and valley-filling algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 8(6):1073–1083, 2014.

[3] Z. Chen, Z. Xie, and Q. Zhang. Community detection based on local topological information and its application in power grid. *Neurocomputing*, 170:384–392, 12 2015.

[4] S. E. Garza and S. E. Schaeffer. Community detection with the label propagation algorithm: A survey. *Physica A*, 534:122058, 2019.

[5] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson. Combining label propagation and simple models out-performs graph neural networks. 10 2020.

[6] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 441–448, New York, NY, USA, 2009. Association for Computing Machinery.

[7] Z. J. Lee, G. Lee, T. Lee, C. Jin, R. Lee, Z. Low, D. Chang, C. Ortega, and S. H. Low. Adaptive charging networks: A framework for smart electric vehicle charging. *IEEE Transactions on Smart Grid*, 12(5):4339–4350, 2021.

[8] Z. J. Lee, T. Li, and S. H. Low. ACN-Data: Analysis and Applications of an Open EV Charging Dataset. In *Proceedings of the Tenth International Conference on Future Energy Systems*, e-Energy '19, June 2019.

[9] N. Pizarro. Structural identity and equivalence of individuals in social networks: beyond duality. *International Sociology*, 22(6):767–792, 2007.

[10] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[11] L. D. Sailer. Structural equivalence: Meaning and definition, computation and application. *Social networks*, 1(1):73–90, 1978.

[12] C. W. Tan, D. W. Cai, and X. Lou. Resistive network optimal power flow: Uniqueness and algorithms. *IEEE Transactions on Power Systems*, 30:263–273, 1 2015.

[13] H. Wang and J. Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

[14] Z. Wang and T. Oates. Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[15] C.-L. Yang, C.-Y. Yang, Z.-X. Chen, and N.-W. Lo. Multivariate time series data transformation for convolutional neural network. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 188–192. IEEE, 2019.

[16] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.