On the pursuit of Graph Embedding Strategies for Individual Mobility Networks

Omid Isfahani Alamdari
University of Pisa
Pisa, Italy
alamdari@di.unipi.it

Mirco Nanni
ISTI-CNR
Pisa, Italy
mirco.nanni@isti.cnr.it

Agnese Bonavita
Scuola Normale Superiore
Pisa, Italy
agnese.bonavita@sns.it

***Abstract*—An Individual Mobility Network (IMN) is a graph representation of the mobility history of an individual that highlights the relevant locations visited (nodes of the graph) and the movements across them (edges), also providing a rich set of annotations of both nodes and edges. Extracting representative features from an IMN has proven to be a valuable task for enabling various learning applications. However, it is also a demanding operation that does not guarantee the inclusion of all important aspects from the human perspective. A vast recent literature on graph embedding goes in a similar direction, yet typically aims at general-purpose methods that might not suit specific contexts. In this paper, we discuss the existing approaches to graph embedding and the specificities of IMNs, trying to find the best matching solutions. We experiment with representative algorithms and study the results in relation to IMN characteristics. Tests are performed on a large dataset of real vehicle trajectories.**

***Index Terms*—individual mobility, graph embedding**

I. Introduction

The recent development and diffusion of location-aware data collection technologies led to an abundance of spatial and mobility data that allows scientists and domain experts to study human mobility at an unprecedented level of depth. The focus of these studies can be either at the (lower) level of trajectories – namely, the single trips, the route they followed, their speed profile, etc. – or at the (higher) level of mobility demand, thus focusing on the places the users stopped by and the transitions between these places. In the latter case, most works analyze the mobility demand from a collective viewpoint, which aggregates the mobility of a population over a territory, aiming to identify traffic flows characteristics and predicting them [1]–[3]. However, when significant data is available on single users or vehicles, a longitudinal analysis can be developed to build a model of the individual's mobility, which can help in several analytical tasks, from understanding mobility patterns [4] to mobility prediction [5], [6], event prediction [7], simulation [8], etc.

The mobility demand of an individual can be effectively represented by means of networks and graphs, where nodes correspond to spatial locations and edges represent trips connecting pairs of locations. In particular, we are interested in *Individual Mobility Networks* (IMNs), a graph abstraction of individual mobility introduced in [9] and later used in several applications [7], [8]. The locations in IMNs are specific to the user and extracted through clustering on their stop points, instead of being fixed (as, for instance, happens with the cognitive maps introduced in [10], based on predefined points of interest), and both nodes and edges are enriched with information of frequency stop/traversal, temporal distribution and spatial aggregates that help to better describe the way the user moved in the period of observation. Some simplified examples are visually illustrated in Figure 2. By abstracting away the specific spatial position of each location, IMNs also provide a rather intuitive way to compare users belonging to different places, aiming to understand common and discriminating characteristics.

The question that arises, however, is now: *how to effectively (and efficiently) compare IMNs?* Which features, aspects of their structure or combinations of them are significant to characterize an IMN w.r.t. others? Very specific applications and tasks might directly come with an ad hoc answer, yet not providing a general approach to the problem. Similarly, while humans can rationalize some discrimination criteria by visually inspecting IMN examples (e.g. by looking at Figure 2 one might think to measure the frequency dominance of the node labeled as '0', or to count the number of peripheral connections – i.e. not involving node '0' – in each graph), that might suffer from subjectivity of personal views, and miss some important aspects.

The scientific literature has tackled this type of problems for over a decade, the most relevant and interesting approaches belonging to the *graph embedding* field, i.e. the art of translating graphs into fixed-size vector representations that encapsulate the useful information implicitly contained in graphs. However, the landscape is rather complex, with each method proposing to model various kinds of concepts and undergoing validation on benchmarks that possess specific, often under-explored, characteristics. As a result, comprehending whether our graphs genuinely align with the expected requirements of the provided solution becomes very challenging. The purpose of this paper, then, is to study what the current state-of-art in graph embedding can do for the specific case of IMNs, starting from a selection of candidate embedding methods; performing a comparative analysis of IMNs' characteristics to see where they are positioned w.r.t. some reference benchmarks in literature; and finishing with a mixed supervised and unsupervised experimentation aimed to identify promising existing solutions

and lasting issues.

This work provides four types of contribution:
1) we introduce IMNs and their related embedding problem, characterizing them with respect to other popular graph embedding benchmarks, in particular identifying their key differences. This aspect is often not sufficiently discussed in literature, as the diversity of datasets employed is typically very limited – mostly belonging to social networks or molecule data sources – and the characteristics of datasets are mainly provided to show their size. Understanding how much the data we want to embed fit the validated benchmarks is an under-explored aspect of the problem;
2) we discuss several families of embedding methods, some of which are selected for experimental evaluation on IMNs. In particular, we highlight the concepts they want to model, the type of information they handle and how they design their aggregation / propagation within the graph;
3) we provide a detailed empirical study of various methods, aiming to understand which approaches seem to extract useful information for IMNs and also what is the intertwined role of different sets of input features and of inferred graph structure information. To this purpose, we design a supervised classification task to obtain objective performance measures, and complement it with a subjective evaluation based on similarity search and visual inspection;
4) finally, the paper is intended to provide also a prototypical example of the process that analysts need to face when they have to orient in the vast and heterogeneous literature on graph embedding, since research papers often are mostly focused on highlighting the originality of the proposed solutions, while more general guidelines for practitioners are seldom discussed.

The rest of the paper is organized as follows: Section II provides an overview of the state-of-art methods for graph embedding; Section III introduces IMNs with a comparative studies of their characteristics vs. other benchmarks; Section IV describes the features, algorithms and evaluation approach adopted; Section V provides an empirical evaluation of the embedding methods selected; finally, Section VI summarizes the work and provides conclusive remarks.

II. Graph Embedding: state-of-art

Since our main objective is to study the applicability of graph embedding methods to Individual Mobility Networks, we provide here a brief overview of graph embedding literature. We remark that the expression *graph embedding* is often ambiguously adopted to refer both to the strategies that yield an embedding for the whole graph and those that instead assign an embedding to single nodes. We will refer to the first type as *graph-level embedding* and to the second as *node embedding*. They are not mutually exclusive, since several graph-level embeddings are actually derived from a node embedding through aggregation of the output node representations, yet they might have different objectives to optimize.

A. Graph-level Embedding

Embedding of a whole graph is typically useful for querying or learning from several graphs of moderate size, where the final objective is to retrieve or classify/cluster graphs. The embedding can in principle be achieved by simply computing a fixed set of user-defined statistics and aggregates, for instance some common properties studied in network science: number of nodes, edges, closed triangles, the diameter of the graph, etc. However, that would not capture well the detailed structure of graphs. In particular, several methods try to consider properties of both the *local structure*, i.e. the connections between each node and its neighborhood, and the *global structure*, i.e. the connection/reachability among all nodes in the graph.

Kernel-based Features: Kernel methods derive features for each node from a given neighborhood of other nodes, typically capturing the local properties of the graph around the node. Most solutions in this category are inspired by the basic Weisfeiler-Lehman (WL) algorithm [11], which starts with initial features associated with each node and iteratively updates them with an aggregation of the neighbors' features, repeating the process a given number of times, and finally aggregating the features of all the nodes in the graph into graph-level features, typically through a distribution histogram for each feature. Properly defining the neighborhood and how values are diffused is fundamental. The LDP method [12] adopts the simplest definition, and is limited to one-step neighbors, computing the distribution of their degrees. Various others follow a wavelet approach, where propagation is performed "jumping" from one node to the others located at exponentially increasing distances, thus capturing also long-range dependencies. For instance, Geometric Scattering [13] computes graph embeddings as statistics of the node values distributions, as obtained at each iteration of the wavelet propagation mechanism, while [14] examines the distributions of node features in subgraphs based on diffusion wavelets.

Neural embeddings: This family of methods treats the neighborhood of a node as its *context*, following techniques like skip-grams inherited from language models. For instance, Graph2Vec [15] builds subgraphs for each node and applies a doc2vec approach [16] where the subgraph represents the document and the nodes' labels the words in it. GL2Vec [17] further improves it by also exploiting labels on edges. We remark that these methods need node labels, which provide a common reference vocabulary among different graphs and make their embeddings comparable.

Random Walk Approaches: These are basically a specific (and very popular) variant of kernel methods, where the neighborhood is identified through several random explorations of the links in the graph starting from the node and performing a given number of steps. Beside the neighborhood, the process assigns a weight or probability to the relation between the explored nodes and the starting one, which are used to propagate information or features values from the former to the latter. The Feather methods [18], for instance, use the weights as probabilities in computing the characteristic function of node features (basically, translating features into complex

numbers and then computing their expected values). Finally, Anonymous Walk Embeddings (AWE) [19] renames nodes with their order of visit in the random walk, which highlights the presence of loops. The basic version of AWE, then, simply computes the frequency of each possible anonymous sequence, and associates the frequency distribution to the whole graph.

Spectral Features: As for random walks, these methods consider transition probabilities between nodes (usually uniform, though external weights might be applied in some cases) and study the information diffusion among nodes, now adopting spectral analysis tools, such as the Laplacian matrix and its eigen-vectors and -values. For instance, NetLSD [20] and IGE [21] adopt a global *heat trace* signature of the graph derived from eigenvalues, achieving isomorphism invariance and adaptivity to scale. FGSD [22] defines a distance function between nodes based on their eigenvectors, computing the overall distribution of distances in the graph, with the possibility to emphasize either local proximity or global structure.

B. Node-level Embedding

Methods in this category aim to embed the individual nodes of a single, usually very large, graph. A graph-level embedding can be in principle derived quite easily by pooling the node embeddings in some way, e.g. through averaging or extracting other characteristics of value distributions. However, in order for it to make sense, the node embeddings need to be consistent across the different graphs we want to embed, so that their representations are comparable. Most works on node embedding do not discuss this aspect in detail, therefore it is often unclear if node aggregation is doable and therefore if the proposed methods also apply to graph-level.

Proximity preservation: These methods aim to infer node embeddings that capture the relations between nodes and their neighbors, possibly at various distances.

Several approaches rely on neural embeddings (already mentioned above), such as DeepWalk [23] (and its improvement Walklets [24]), which applies the skip-gram approach by defining contexts through random walks. Node2vec [25] does something similar, with various node sampling strategies to define neighborhoods, improved by Diff2vec [26] through the use of diffusion techniques. GraRep [27] also extends the concept of Deepwalk, considering k-step neighborhoods, and implementing it through Laplacian matrix factorization.

GLEE [28] applies Laplacian Eigenmaps to directly encode the graph structure (in particular, the proximity) through the geometry of the embedding space, yielding node embeddings from which we can estimate, for instance, the number of common neighbors of two nodes or of short paths.

Various papers study the problem in terms of the Laplacian matrix factorization, such as NetMF [29], which basically provides a reformulation of Deepwalk; BoostNE [30], that learns multiple graph representations at different granularities from coarse to fine; or the simple approach in [31], which takes k lowest-frequency Laplacian eigenvectors as embeddings.

Finally, HOPE [32] introduces the idea of assigning to each node two separate embeddings, a *source* and a *target* one, which fit the contexts with oriented graphs, also considering various node proximity measures to capture with the embeddings, such as Katz's distance, Rooted Page Rank and others.

Embedding Nodes with Attributes: Most of the works described so far look at the network properties of graphs, not considering additional existing features of its components. In particular, very frequently the nodes of the graph have either categorical features (e.g. the tags of posts in social networks) or numerical ones (frequency of visits to a web page, the length of a document, etc.) that could strongly help the embedding.

MUSAE [33] adopts a neural embedding on top of random walks (thus similar to Deepwalk), where the sequences of nodes (actually, node values, since they are attributed) are built with random walks sampling one node every r along the path. Different r values are used, pooling (AE) or simply joining (MUSAE) the corresponding results of each node into a larger embedding. Also inspired by Deepwalk, SINE [34] formulates a probabilistic learning framework that separately models pairs of node-context and node-attribute relationships, where each node learns its representation by considering context nodes and observable attributes of the node. BANE [35] adopts a WL diffusion (thus a kernel method) over the nodes attributes, and aims to obtain embeddings in a (binary) Hamming space. TENE [36] considers the case where nodes have a text annotation, and thus devise a matrix factorization schema to create separate embeddings for network structure and text similarity, later joined and optimized together. An analogous process is followed by FSCNMF [37], which outputs two regularized embeddings of the network corresponding to structure and content, later combined as the final representation.

Graph Neural Networks: The most common neural network architectures adopted in the graph domain are Graph Convolution Networks (GCN), which work in a very similar way to the Weisfeiler-Lehman (WL) algorithm, yet with learnable weights driving the aggregation steps. Being a supervised approach, it can be applied only when nodes are assigned to label values. An alternative approach is provided by *self-supervised learning* (SSL) approaches for graphs [38], which extract informative knowledge through ad hoc pretext tasks without relying on existing labels. SSL is often applied as pre-training of a model, and is then followed by a refinement step with conventional GCNs over a labeled dataset. Similarly, *encoder-decoder* methods (e.g. [39]) learn an encoding function (which produces the embeddings) and a decoding one that minimizes a loss w.r.t. properties of the graph, typically the adjacency matrix representing the graph connections. ASNE [40] can be seen as a representative example, which separately embeds nodes and their features, then fuses them through NN layers used to estimate edge probabilities between node pairs.

III. Individual Mobility Networks

Definition. Given a user u, their associated history H_u can be processed to extract their *individual mobility network* (IMN) G_u. An IMN describes the individual mobility of a user through a graph representation of her locations and

movements, grasping the relevant properties and removing unnecessary details. Its nodes correspond to locations that represent a group of stop points identified through a spatial clustering-based aggregation [41]; and its edges correspond to movements representing groups of similar trips between two locations [42].

Definition 1 (Individual Mobility Network): Given a user u, we indicate with $G_u = (L_u, M_u)$ her *individual mobility network*, where L_u is the set of nodes and M_u is the set of edges. Given an aggregation operator agg, for each node $l \in L_u$ we define the following functions:

- $\omega(l) =$ number of trips in H_u reaching location l;
- $\delta(l) = agg(\{\text{durations of stops in } l\})$;
- $\rho(l) = agg(\{\text{arrival times of trips reaching } l\})$;
- $\pi_t(l) = agg(\{\text{durations of trips reaching } l\})$;
- $\pi_d(l) = agg(\{\text{lengths of trips reaching } l\})$;

Operator agg can return either a single value (e.g. median) or a n-ple (e.g. average and standard deviation, or quartiles). The same functions are also defined on edges (movements) $m = (l_i, l_j) \in M_u$ in a similar way, this time considering only trips that start from l_i and reach l_j.

Comparative study of IMN properties. IMNs are rather specific networks that show properties slightly different from other graph datasets typically used to validate graph-level embedding methods. Table I compares the averages of six statistics across six popular datasets plus IMNs. The values include size of the graphs ($|V|$), diameter, density (fraction of edges over the theoretical maximum), nodes degrees, entropy of node degrees and, finally, a statistic we called *maximal ego coverage*, computed as the fraction of nodes contained in the largest ego-network of the graph. We remark that, while IMNs are directed graphs, the other datasets considered are not. Thus, to have fair comparisons, for these measures also IMNs have been converted to their undirected version.

From this comparison, it emerges that while the average size and diameter of IMNs are similar to other graph datasets, their degree and density are rather low, thus showing a general sparseness of the graphs. At the same time, the entropy is relatively low and the maximal ego coverage is relatively high – much higher than what we would expect given IMNs' low density. This suggests the presence of a few highly-connected nodes in IMNs counterbalanced by many low-degree ones. A clearer picture is given in Figure 1, showing boxplots for the more interesting measures. We can see that in IMNs the density is always very low, with a smaller inter-quartile range, while the maximal ego coverage is significantly higher than the others – excepted three (Deezer, Twitch and IMDB-M) that are however characterized by an extremely small diameter (exactly 2 for the ego-networks Deezer and Twitch, between 1 and 2 for IMDB-M), which makes high ego coverages easily very high. The Reddit graphs are those closer to IMNs characteristics, yet they are significantly smaller and denser.

Sample IMNs. The typical structure of IMNs can be seen in the small sample of graphs shown in Figure 2, where in addition to nodes and edges we also represent the frequency

Dataset	avg $\lvert V\rvert$	avg Diam.	avg Density	avg Deg	avg Deg. Entropy	avg Ego Cov.
DEEZER	23.49	2.00	0.23	5.55	1.64	0.94
TWITCH	29.67	2.00	0.20	5.84	1.93	0.96
GITHUB	113.79	5.86	0.08	4.12	1.51	0.56
Reddit	23.93	4.58	0.12	2.09	0.84	0.71
MUTAG	17.93	8.22	0.14	2.21	1.02	0.18
IMDB-M	13.00	1.47	0.77	10.14	0.48	0.90
IMN	33.96	3.68	0.12	3.26	1.34	0.78

TABLE I
COMPARISON OF STATISTICS OF VARIOUS GRAPH DATASETS.

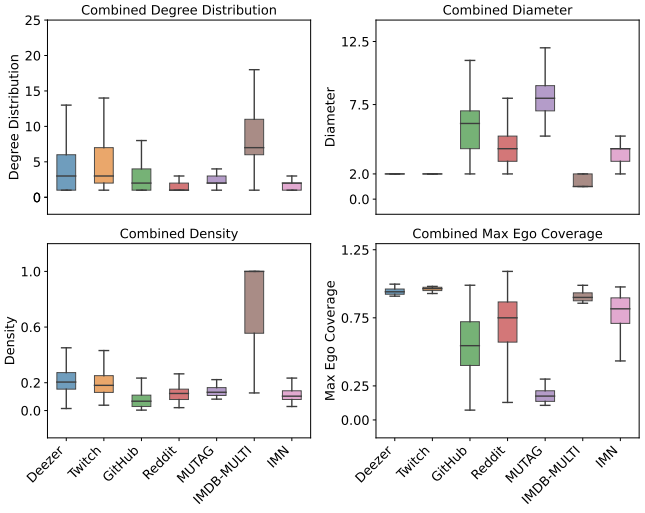

Fig. 1. Distribution of nodes properties in various graph datasets. From top-left: degree distribution, diameter, density and maximal ego coverage.

of stops at locations as nodes' size, and the frequency of trips as thickness of edges. Notice that IMNs are oriented graphs, and edges flow in a clockwise direction. For instance, in the top-left IMN the flow from '0' to '1' is slightly stronger than the flow from '1' to '0'. Finally, nodes are numbered in order of stop frequency and self-loops (i.e. trips that start and end in the same location) are also represented.

As the statistics in the previous section suggested, IMNs are mostly characterized by a central, high-degree node (most likely corresponding to the home location of users) connected to a large number of other nodes. Often this location is connected to a second one (most likely the workplace) with high-frequency edges. The presence of these two strong *poles* drastically reduces the diameter of the graphs, although IMNs generally show a low density of connections.

The complexity of IMNs is rather variable, including small graphs with one node that dominates (top-right of Figure 2, notice the strong self-loop), larger graphs with a quasi-star shape built around node '0' (bottom-left graph), others with more chaotic connections (top-left) and more distributed node frequencies (bottom-right).

IV. FEATURES, ALGORITHMS AND VALIDATION TASK

In this section we introduce the three components of our experimentation: the dataset used for evaluation; the embedding

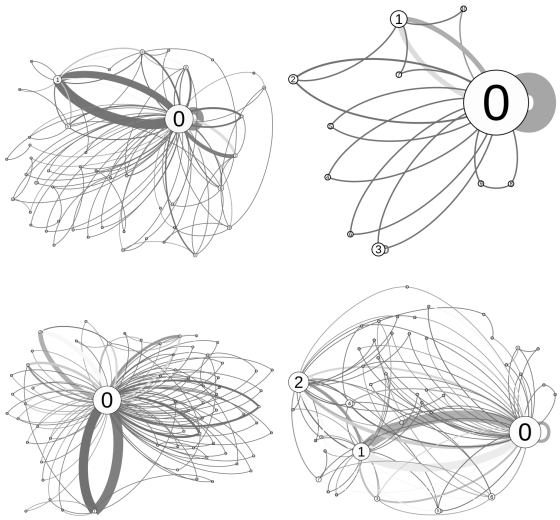

Fig. 2. Sample IMNs. The Individual Mobility Network of four sample users along two months is shown. Size of nodes and thickness/darkness of edges represent their frequency. Nodes are numbered in order of stop frequency.

methods tested; the downstream task and its result evaluation.

A. IMNs and features

Our evaluation of embedding strategies is based on a dataset of 1000 IMNs, each corresponding to a private vehicle moving in the Tuscany region, Italy, over a period of 4 weeks. In particular, each vehicle belongs to one of five provinces, and the corresponding IMN was obtained following the process in [43]. The distribution of vehicles in the five provinces is perfectly balanced, i.e. 200 vehicles per province.

Node features. We enrich IMNs with four classes of features:

- Network Structure features: degree, clustering, node centrality, entropy of next location.
- Location usage features: stop frequency, stay duration, all the above stratified by time-of-day and day-of-week.
- Trajectory features: average length of incoming trips, average incoming trip duration, radius of gyration of the vehicle w.r.t. the node location.
- Geospatial features: Points of Interest (POIs). We remark that these features might be strongly location-specific, and thus directly help identifying places and their characteristics. For this reason, we will experiment both with and without this information source. Alternative information belonging to the same wide category might be integrated, such as distance from city center (or center of gravity, the home location, etc.), land usage, etc.

Edge features. This kind of information is generally not supported by graph embedding methods. One exception is GL2vec, that builds a dual graph where edges are represented as nodes, and thus their features are used in the embedding generation. For the others, some algorithms allow using edge weights or probabilities, therefore some types of edge features might be translated into usable values. In our case, edges are associated with trip frequencies, thus we could in principle transform them into transition probabilities and use them in random walks methods, such as AWE and FEATHER-G, by modifying their usual transition choice mechanisms. However, preliminary tests with simple feature translations showed no improvement in the output quality, thus this option was discarded for the time being, and left as a potential future direction to explore.

B. Embedding algorithms

The embedding is performed adopting several different algorithms, chosen from the state-of-art described in Section II in order to cover the most important general approaches. The selection was also driven by the availability of open-source code or libraries, and in most cases we exploited the Karate Club library [44]. We consider both graph-level methods, which thus directly return the embedding we needed; and node-level methods, where a final aggregation step is performed to obtain a graph representation. We briefly list them and provide details about the selected parameters and the adaptations performed (where needed).

Graph-level methods include the following:

- LDP [12], a simple **kernel method**. The only parameter is the number of bins, which we set to a default value 32;
- GeoScattering [13], another **kernel method**, requiring as parameters the order and moments adopted in creating the graph spectral descriptors, both set to the default 4;
- graph2vec [15] is a **neural embeddings** approach. Also, in this case, default parameters were used, in particular, the output embedding size was set to 128;
- GL2Vec [17] is another **neural embeddings** method, similar to graph2vec, and the same parameters were set;
- FEATHER-G [18] has a **random walk** component and an aggregation of sampled values, which require to set, respectively, an order for adjacenty matrix (set to default 5) and evaluation points (set to the default 25 values uniformly sampled from 0 to 2.5).
- AWE [19] is also a **random walks**-based method, whose main parameter is the length of generated walks. After testing values between 3 and 8, we chose 5 as the best one;
- GCN-self is a **self-supervised** learning approach [38], based on a standard Graph Convolutional Network to learn predicting a set of classes that is similar yet different from the downstream task. The output of the last convolution layer provides an embedding that is used as input for the final classification task. In our case, we test two training objectives: one consists in predicting the radius of gyration (RoG) of the mobility of the user, discretized into 4 equal frequency intervals[1]; the other predicts the fraction of systematic trips of the user (Sys), measured

[1]We remark that a similar measure is provided at the level of single nodes, which however represents a more local view that might miss the overall picture, since the global RoG is computed w.r.t. the overall mobility center, which usually does not correspond to any of the user's locations.

as the ratio of trips between nodes '0' and '1' (usually corresponding to home-work routine trips) over the total, discretized into 4 classes.

Node-level methods adopted include:

- FEATHER-N is the node-level version of FEATHER-G, described above, thus basically a **random walks** method, the main difference being that it also allows using **node features**, whereas FEATHER-G adopts a simple node degree and clustering coefficient. Parameters are set as above;
- MUSAE [33] is a **neural embeddings** method that also uses **node features**. The algorithm employs random walks, for which we set 5 walks per node of length 10, using order 3. MUSAE makes use of categorical node attributes, the original node features have been discretized into 5 equal-frequency bins.
- TENE [36] also uses **node attributes**, yet adopts a **matrix factorization** schema. Also in this case, categorical node features are used (mostly adopted to represent text labels), thus the same discretization as MUSAE was applied. The technical parameters of the algorithm were set to their default values.
- ASNE [40] is a **Graph Neural Network** approach that, as MUSAE and TENE, exploits node labels. Also in this case, we translated original features to discretized values in the same way as above, and the parameters of the algorithm were set to their default values.

The final graph embeddings for node-level methods have been computed through a mean-pooling of nodes' embeddings.

Among the categories of algorithms described in state-of-art, we are omitting *graph-level spectral features* methods, whose same ideas are basically implemented with different technical tools in other approaches; and *node-level proximity preserving* methods, since aligning the embeddings they produce across different graphs can be very problematic.

C. Evaluation approach

We evaluate the quality of embeddings in two ways. First, in order to have objective performance measures, we define a downstream graph classification task, using as target label the reference province of each vehicle/graph. We remark that no node and edge feature adopted has a direct geographical reference, such as spatial coordinates, thus the task is indeed challenging. We test this approach by building a simple logistic classification model on top of the embeddings obtained with each of the embedding methods listed above, measuring the performance with the standard Area Under the ROC (AUC).

Second, we simulate a nearest-neighbors query task, where a small number of IMNs are randomly sampled (the queries), and for each of them we select the 10 most similar IMNs in our dataset, based on the cosine similarity computed between the embeddings of the IMNs to compare. Due to space reasons, we perform this task only on a small selection of methods, and visually compare the returned set of IMNs. Clearly, the outcomes of this task are subjective, and different evaluators might draw different conclusions.

D. Baseline and supervised approaches

In addition to the embedding methods listed above, we consider as baseline the usage of raw feature statistics, where, for each node feature, we compute the average value and standard deviation w.r.t. all nodes of a graph.

Finally, while we are not directly interested in supervised graph classification approaches, since the embedding they produce might be too specific for the prediction task, we tested two of them to provide reference performance values. Clearly, our expectation is that they can provide better predictions. The first method considered is a standard GCN (named GCN-PRV, since it is directly built on the target "province"), with three layers of convolution and a mean pooling before a final dense neural network to predict the province (five output neurons, hot-encoding the five provinces). The second method is a simpler dense neural network that takes as input the features of all nodes, stacked according to a node ordering based on nodes' stop frequency (see Section IV-A), padding values for graphs with less nodes. The idea behind this choice is to exploit such ordering (which looks rather intuitive, at least for the top locations, usually associated to home and work location) to bypass the standard pooling step in GCNs, which merges all neighbors losing their identity. The network, named DNN-sorted-nodes, is composed of three layers.

V. Empirical evaluation

In this section we summarize and comment the experimental results obtained in a downstream classification task, studying the impact of different sets of input features, and then on an unsupervised nearest-neighbors retrieval task, where the results are visually evaluated through a subjective validation.

A. Classification task

Table II reports a comparison of results obtained using the different embedding methods discussed above. On top of each embedding a classifier is built through a simple logistic regression. The choice of such simple classifier was intentional, in order to appreciate how well the embeddings could highlight the characteristics that can discriminate among the classes (in our case, the province where the user was moving). The performance measure adopted is the area under the ROC curve (AUC), and, as mentioned in Section IV-A, we report it separately on two columns, corresponding to the case where only non-geospatial features are available ("w/out POI") and that where all features are used ("with POI"). For each column, we highlight the top three results (resp. bold, italic underlined, and just underlined). Embedding methods are grouped based on whether they exploit features or not – in the latter case, clearly, the distinction based on POIs does not apply. Among them, we report results for self-supervised approaches that train the embedding for different tasks, namely predicting the radius of gyration of the user's mobility (GCN-RoG) and predicting the ratio of their trips that are systematic (GCN-Sys); and for the two supervised methods.

First of all, results on the top group of the table clearly show that not using features yields very poor results, basically

TABLE II
AUC PERFORMANCES OF EMBEDDING METHODS ON THE PROVINCE CLASSIFICATION PROBLEM, BASED ON A LOGISTIC CLASSIFIER.

	Method	AUC	
no features	LDP	0.4997	
	GeoScattering	0.5000	
	Graph2Vec	0.4992	
	GL2Vec	0.4837	
	FEATHER-G	0.5111	
	AWE	0.4693	
		w/out POI	with POI
features	raw features	*0.5603*	**0.7984**
	FEATHER-N	**0.5860**	*0.7759*
	MUSAE	0.5261	0.4940
	TENE	0.4928	0.5316
	ASNE	0.4904	0.5203
	GCN-RoG	0.5558	0.6781
	GCN-Sys	0.5297	0.5787
sup.	GCN-PRV	0.5241	0.9174
	DNN-sorted-nodes	0.6352	0.9388

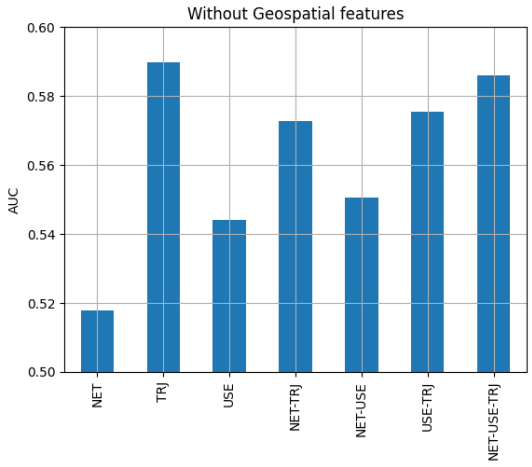

Fig. 3. Features impact: network (NET), mobility (TRJ), location use (USE).

highlighting the fact that the plain structure information that such methods can derive does not capture the discriminating characteristics of the different areas. Their performance is significantly worse than using only aggregate raw features (see the first line of the "features" block).

Embedding methods that use all features excepted POIs are generally better, yet only Feather-N and self-supervised methods improve over using plain features, suggesting that the aggregation and diffusion processes implemented sometimes can destroy the information content in the original data. Apparently, the specific structure of IMNs makes most embedding methods add noise and hide the useful component of the initial node features. The self-supervised approaches perform relatively well, yet still below plain raw features, and significantly worse than Feather-N. Apparently, having a global objective to train for helps the emergence of generally useful characteristics, yet the task difference makes the final balance negative, as the information lost w.r.t. raw features is larger than that gained through the embeddings.

Examining the results using also POI information, we can see that the three top methods remain the same, yet yielding much higher AUC values. At the same time, not even Feather-N can improve over raw features. The other feature-based methods, instead, either do not improve significantly by adding POI information or even worsen slightly. In general, these results suggest that POI data provide a strong predictive power as they are, and any aggregation, propagation, or convolution simply loses useful information along the process.

B. Impact of features

In this section we focus on the best performer among the unsupervised approaches (Feather-N) and investigate the role that different input features play. Figure 3 shows the AUC obtained selecting different subsets of the available features: network structure measures (NET), trajectory features (TRJ), location usage (USE). Geospatial features (POIs) are excluded, since their impact is already known to be large. The effects of other features when POIs are included (omitted for space limits) are very similar to those presented here.

Trajectory features taken alone are the best setting, while network structure and location usage have a much weaker impact. While the combination of NET and USE yields much better results than the two taken alone – thus, apparently, they complement well each other – combinations of TRJ with other features yield slightly worse results, suggesting that the added useful information is counterbalanced by the noise they introduce in the simple classification model adopted.

C. Supervised approaches

Graph convolutional network (GCN-PRV). Following the standard GCN architecture, nodes are initially represented through their features, and then updated through three convolutional layers, i.e. for each node the mean of its neighbors' features is computed and passed through a linear transformation, then through a non-linear function. The results become the new node representation, and the process is repeated (in our case) three times. The final node representations are then aggregated (in our case, through mean pooling) and fed to a final dense layer with one output node for each class.

The results obtained for GCN-PRV are rather heterogeneous, since with no POI data the performances are slightly worse than the self-supervised approach, while with POI data the AUC increases beyond 90%, largely improving over embedding and raw feature approaches. This suggests that, due to IMNs structure, the GCN mechanism is good at identifying and emphasizing very important basic features (as in the case of POIs) but not to infer useful information from the graph structure and other weaker features.

Fully connected layers (DNN-sorted-nodes). This model is a graph-level classification approach utilizing fully connected layers (dense layers) and a pooling operation. The preprocessing stage involves padding the node features of each graph to ensure consistent input dimensions. The model's architecture comprises three hidden dense layers, each followed by a dropout layer to prevent overfitting. The final dense layer

maps the graph-level features to the desired output dimension, while a softmax activation function is used to derive the predicted class probabilities. The model training optimizes the Sparse Categorical Crossentropy loss function, adopting the Adam optimizer.

The results in Table II show that this less refined approach is actually more robust than the GCN one, improving over all the other methods, also when no POI data is involved. This might suggest that nodes' identities are important to learn how to treat their features, maybe highlighting more central nodes (e.g. home and work locations) and de-emphasizing the others. In order to inspect this hypothesis, we evaluated what happens when we shuffle the order of nodes, independently on each graph, thus losing their identity. However, the results, omitted for space limits, show that the variation is surprisingly small (always less than 0.0105) and practically insignificant, both with and without POI data. This suggests that the most important factor in this prediction task is how different features are combined, and not the node they come from. That is similar to what raw features achieve, in addition to the non-linear combination of features provided by the dense layers that helps to better identify the most discriminant information.

D. kNN Query Evaluation

In this section, we use the embeddings obtained above to perform a kNN queries over a small set of IMNs and then visually review the results. Due to space limits, we will present here the results obtained on two query IMNs, showing the top 10 most similar IMNs returned. For each query IMN, we perform the task both using the embeddings obtained with Feather-N (the best performer in the classification task) and with the self-supervised GCN-RoG, comparing the results. The similarity between two embeddings e_1 and e_2 is computed using the *cosine similarity* $s(e_1, e_2) = \langle e_1, e_2 \rangle / ||e_1|| \cdot ||e_2||$.

An overall comparison of results showed that the two embeddings generally return very different sets of outputs, on average having an overlap smaller than 10%. In the following, we consider a query IMN where the overlap is relatively high (two IMNs over ten) and then one where the overlap is nil.

Figure 4 shows the first query IMN (on top) and the 10 top results returned, ranked by similarity. For the sake of readability, the graphs report the frequency of edges (numerically and as thickness) while nodes only show the frequency rank ('0' is the most frequent one, and so on). The overlaps between the two answer sets are highlighted. A comparison of results shows that Feather-N returns IMNs of similar size and complexity of the query. Also, the query contains a central *open triangle*, namely the most frequent node is strongly connected to the second and third most frequent ones, which are not (significantly) mutually connected. This same pattern seems to emerge in most of the Feather-N answers. On the contrary, GCN-RoG output seems to have fewer open triangles and also tends to include more crowded IMNs and star-like shapes than the query. From this perspective, Feather-N seems to return intuitively better answers.

Figure 5 presents the results with the second query, which show similar characteristics to the previous case: GCN-RoG tends to include much larger and densely connected answers, where the central node is more predominant than in the query, whereas Feather-N's appear more balanced, except very few cases presenting a star-like shape.

VI. Summary and Conclusions

Our exploration started from realizing how much our data, namely IMNs, are different from typical benchmarks used in the graph embedding literature: a semantic difference, since most benchmarks deal with human interactions data or physically connected elements of molecules, whereas IMNs are about movements, thus making recurring concepts in embedding literature like information propagation and bindings not perfectly fit; and a statistical difference, since the empirical exploration of IMNs' properties resulted to be different from many of the others.

The review of existing graph embedding methods highlighted the existence of a limited set of fundamental approaches, plus several variants and improvements. Many of them are general purpose, yet their validity is typically assessed on prediction tasks (which are not our primary objective) over a limited set of application domains. That makes it difficult to identify the promising methods to choose, for which reason we selected approaches from the most representative algorithms in the literature that could be applied to graph-level embedding. This lack of theoretical foundations in the selection of an algorithm is an important gap that can only partially be filled by purely empirical tests.

Empirical evaluations highlighted how the (rich) node features of IMNs are fundamental to achieving acceptable results, yet also suggesting that most methods are not able to handle them properly. Among the methods tested, we could identify only one – a simple adaptation of Feather-N to work at the graph level – that seems to perform better than no-embedding input features, and yet with no large margins. We performed a simple visual check of the embeddings produced through kNN queries by looking at the exterior graph properties (nodes, connections, and weights), which showed results close to what we expected intuitively.

Overall, the claim of our paper is that current graph embedding literature provides several appealing approaches and, yet, very poor support to the development of applications, leaving the analyst alone in orienting within the vast literature, guided almost exclusively by empirical explorations. Our exploration aims to provide a first example of how to develop an analytical process in this context.

The natural evolution of this work goes in two directions. The first one is specific for IMNs and includes the development of embedding methods ad hoc for them, aimed to better exploit the abundant features at node and edge level and to cope with their particular graph structure. The second one is more theoretical and aims to classify existing methods based on what kind of information is actually built from

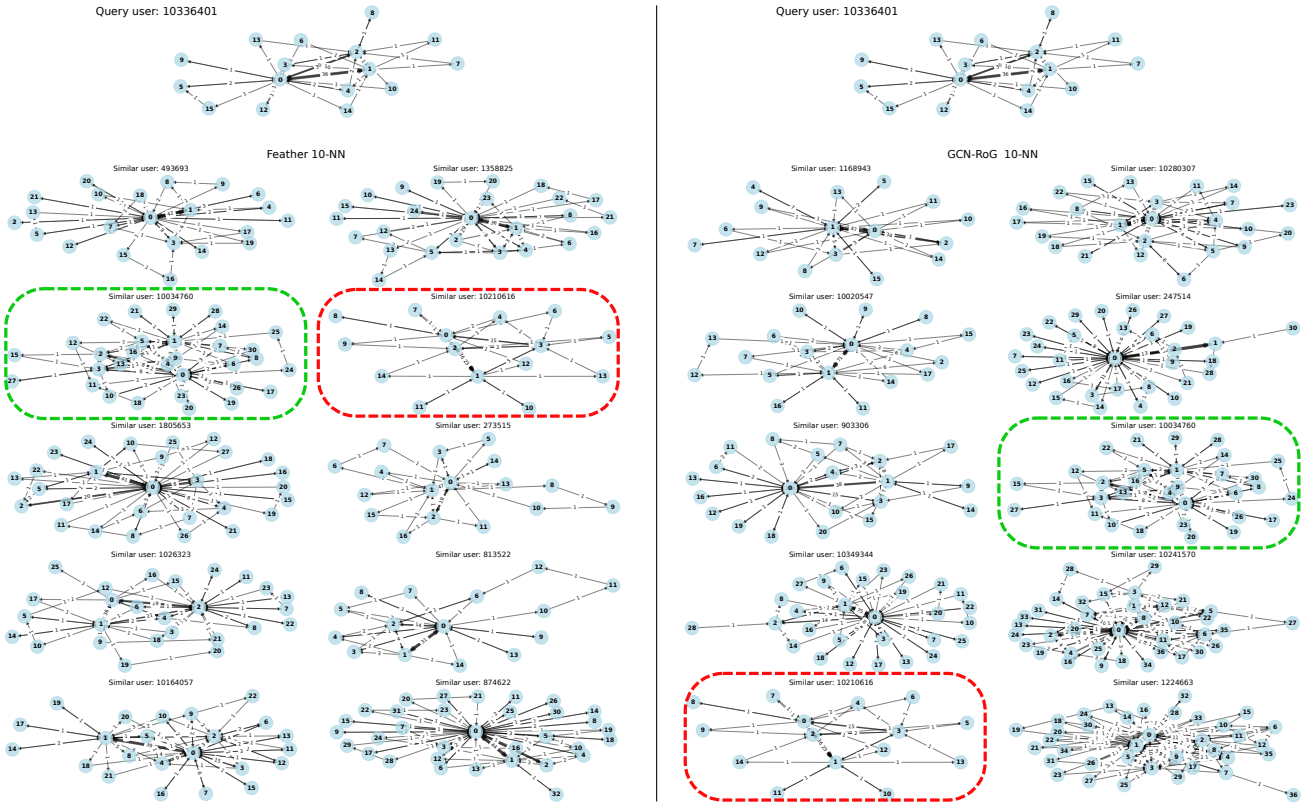

Fig. 4. Query test 1: Top 10 most similar IMNs to a sample user, using Feather-N (left) and GCN-RoG (right) embeddings.

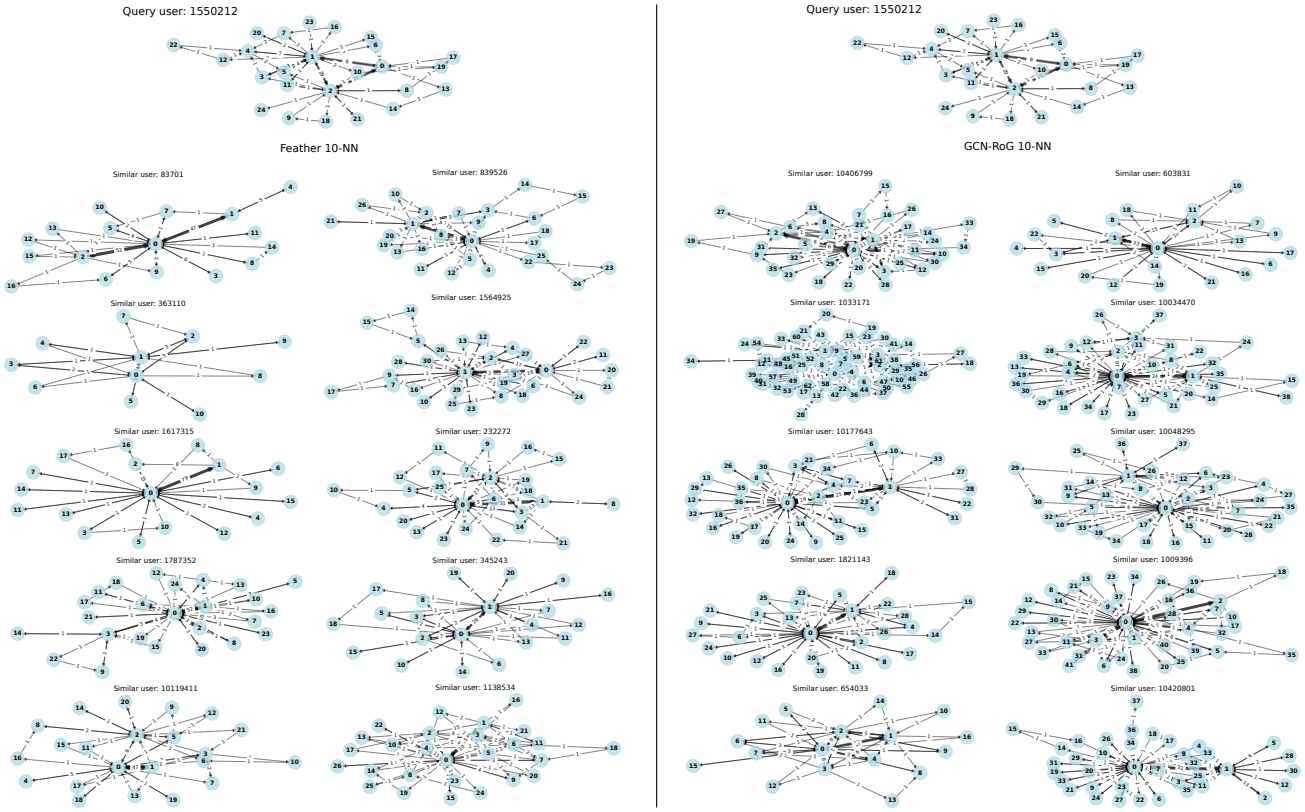

Fig. 5. Query test 2: Top 10 most similar IMNs to a sample user, using Feather-N (left) and GCN-RoG (right) embeddings.

the input graph structure and features, abstracting away from formalization and computational aspects, which often cover very similar concepts behind different covers.

REFERENCES

[1] W. Jiang and J. Luo, "Big data for traffic estimation and prediction: A survey of data and tools," *Applied System Innovation*, vol. 5, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2571-5577/5/1/23

[2] H. Yuan and G. Li, "A survey of traffic prediction: from spatio-temporal data to intelligent transportation," *Data Science and Engineering*, vol. 6, no. 1, pp. 63–85, Mar 2021.

[3] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Syst. Appl.*, vol. 207, no. C, nov 2022.

[4] O. Cats and F. Ferranti, "Unravelling individual mobility temporal patterns using longitudinal smart card data," *Research in Transportation Business & Management*, vol. 43, p. 100816, 2022.

[5] R. Trasarti, R. Guidotti, A. Monreale, and F. Giannotti, "Myway: Location prediction via mobility profiling," *Inf. Syst.*, vol. 64, pp. 350–367, 2017.

[6] Z. Ma and P. Zhang, "Individual mobility prediction review: Data, problem, method and application," *Multimodal Transportation*, vol. 1, no. 1, p. 100002, 2022.

[7] R. Guidotti and M. Nanni, "Crash prediction and risk assessment with individual mobility networks," in *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, 2020, pp. 89–98.

[8] M. Böhm, M. Nanni, and L. Pappalardo, "Gross polluters and vehicle emissions reduction," *Nat. Sustain.*, vol. 5, no. 8, pp. 699–707, 2022.

[9] S. Rinzivillo, L. Gabrielli, M. Nanni, L. Pappalardo, D. Pedreschi, and F. Giannotti, "The purpose of motion: Learning activities from individual mobility networks," in *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, 2014, pp. 312–318.

[10] R. Pérez-Torres, C. Torres-Huitzil, and H. Galeana-Zapién, "A spatio-temporal approach to individual mobility modeling in on-device cognitive computing platforms," *Sensors*, vol. 19, no. 18, 2019.

[11] A. Leman and B. Weisfeiler, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.

[12] C. Cai and Y. Wang, "A simple yet effective baseline for non-attributed graph classification," *arXiv preprint arXiv:1811.03508*, 2018.

[13] F. Gao, G. Wolf, and M. Hirn, "Geometric scattering for graph data analysis," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2122–2131.

[14] L. Wang, C. Huang, W. Ma, X. Cao, and S. Vosoughi, "Graph embedding via diffusion-wavelets-based node feature distribution characterization," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3478–3482.

[15] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[16] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[17] H. Chen and H. Koga, "Gl2vec: Graph embedding enriched by line graphs with edge features," in *International Conference on Neural Information Processing*. Springer, 2019, pp. 3–14.

[18] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1325–1334.

[19] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 2191–2200.

[20] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "Netlsd: Hearing the shape of a graph," in *Procs. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'18)*, 2018.

[21] A. Galland and M. Lelarge, "Invariant embedding for graph classification," in *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, Jun. 2019.

[22] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[23] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014. [Online]. Available: http://arxiv.org/abs/1403.6652

[24] B. Perozzi, V. Kulkarni, and S. Skiena, "Walklets: Multiscale graph embeddings for interpretable network classification," *CoRR*, vol. abs/1605.02115, 2016. [Online]. Available: http://arxiv.org/abs/1605.02115

[25] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[26] B. Rozemberczki and R. Sarkar, "Fast sequence-based embedding with diffusion graphs," *CoRR*, vol. abs/2001.07463, 2020. [Online]. Available: https://arxiv.org/abs/2001.07463

[27] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Procs. of the 24th ACM Int. Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, p. 891–900.

[28] L. Torres, K. S. Chan, and T. Eliassi-Rad, "Geometric laplacian eigenmap embedding," *CoRR*, vol. abs/1905.09763, 2019. [Online]. Available: http://arxiv.org/abs/1905.09763

[29] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization," in *Proceedings of the 11th ACM Int. Conference on Web Search and Data Mining*. ACM, feb 2018.

[30] J. Li, L. Wu, and H. Liu, "Multi-level network embedding with boosted low-rank matrix approximation," *CoRR*, vol. abs/1808.08627, 2018. [Online]. Available: http://arxiv.org/abs/1808.08627

[31] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. MIT Press, 2001.

[32] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Procs. of the 22nd ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, p. 1105–1114.

[33] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *CoRR*, vol. abs/1909.13021, 2019. [Online]. Available: http://arxiv.org/abs/1909.13021

[34] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "SINE: scalable incomplete network embedding," *CoRR*, vol. abs/1810.06768, 2018. [Online]. Available: http://arxiv.org/abs/1810.06768

[35] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *2018 IEEE International Conference on Data Mining (ICDM 2018)*. IEEE, 2018, pp. 1476–1481.

[36] S. Yang and B. Yang, "Enhanced network embedding with text information," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 326–331.

[37] S. Bandyopadhyay, H. Kara, A. Kannan, and M. N. Murty, "FSCNMF: fusing structure and content via non-negative matrix factorization for embedding information networks," *CoRR*, vol. abs/1804.05313, 2018. [Online]. Available: http://arxiv.org/abs/1804.05313

[38] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5879–5900, 2023.

[39] M. Tang, P. Li, and C. Yang, "Graph auto-encoder via neighborhood wasserstein reconstruction," in *International Conference on Learning Representations*, 2022.

[40] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[41] R. Guidotti *et al.*, "Tosca: two-steps clustering algorithm for personal locations detection," in *SIGSPATIAL/GIS*. ACM, 2015, p. 38.

[42] R. Guidotti, R. Trasarti, M. Nanni, F. Giannotti, and D. Pedreschi, "There's a path for everyone: A data-driven personal model reproducing mobility agendas," in *2017 IEEE International conference on data science and advanced analytics (DSAA)*. IEEE, 2017, pp. 303–312.

[43] M. Nanni, R. Guidotti, A. Bonavita, and O. I. Alamdari, "City indicators for geographical transfer learning: an application to crash prediction," *GeoInformatica*, vol. 26, no. 4, pp. 581–612, 2022.

[44] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs," in *Procs. of the 29th ACM Int. Conf. on Information and Knowledge Management (CIKM '20)*. ACM, 2020, p. 3125–3132.