Evaluating event influence on TGN performance in bike sharing station occupancy forecasting

Štefan Horvat, Damjan Strnad, Gregor Horvat, Matej Brumen, Domen Mongus
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, 2000 Maribor, Slovenia
{stefan.horvat, damjan.strnad, g.horvat, matej.brumen, domen.mongus}@um.si

Abstract. *Station occupancy forecasting is an important task in bike sharing networks, particularly for anticipating when the stations are empty or full. Trips and redistribution events form a stream of timestamped interactions used to encode station memories. This paper investigates the influence of events is investigated using a hybrid TGN architecture in conjunction with temporal feature encoders used to produce per-station forecasts. Empirical results indicate that event memory improves model performance, but more complex event structure variants underperform the simpler hybrid TGN model. The study highlights the value of event streams while emphasizing the need for caution when adding model complexity.*

Keywords. machine learning, temporal graph network, bike sharing network, event stream

1 Introduction

Bike sharing system changes state over time and station connections are created by user trips and redistribution operations. Most forecasting models try to capture patterns in station time series events in conjunction with static graph structures, where edges usually represent proximity, historical correlations or feature aggregates. This approach captures spatial dependencies but can miss or lack informational structure that can be pres ent in the interaction events between the nodes (Rossi et al., 2020). Better forecasts can support more sustainable network operations by accurate predictions that could reduce the unnecessary service trips.

In this work we study the role of windowed event memory streams for a bike sharing network in Barcelona. The goal is to produce short term forecasts of available station slots. These could be used to optimise the operational structure for easier scheduling of station maintenance and allow better responses when station operational limits are reached. Instead of treating the problem as a snapshot of fixed graphs over time, we employed timestamped events between station which represent trips and redistribution operations. These events are fed through a Temporal Graph Network (TGN) encoder, allowing each station to maintain a compact representation of historic interactions. Instead of using graph convolution over a static graph, the TGN model maintains and updates station memory from a continuous event stream.

The study combines multiple individual components to examine the role of continuous event flow in bike sharing station occupancy forecasting. Besides the temporal baseline, a hybrid TGN based forecasting model is tested that combines station history and event memory representation. Additionally two model variants are tested in the study, a GraphSAGE based neighbourhood aggregation model over event driven connections, and a dual memory hybrid TGN model that maintains a separate memory based on the event type. The results indicate that event information improves forecasting, and that a compact event memory formulation can outperform more complex hybrid TGN variants.

2 Related Work

Forecasting in bike sharing networks is commonly studied as a station level spatio-temporal prediction problem. Most of the literature emphasizes that station states depend on temporal patterns, spatial relationships between stations, and external factors, such as weather, events, and station imbalance. Station level demand predictions can aid the process of redistribution and produce accurate forecasts for unmet demand while considering external factors. Similarly, in excess demand prediction rebalancing is framed as a response to stations becoming empty or full, which motivates forecasting station occupancy rather than only aggregate demand (Hulot et al., 2018).

Bike sharing networks are usually presented as graphs composed of nodes that represent stations, and edges, which can represent a wide variety of relationships, such as spatial proximity or historical flows. There were several attempts to try and extract graph patterns and train the models to predict node states or flows. One of them uses a graph convolutional model to learn hidden pairwise correlations between the stations (Lin et al., 2018). STG2Vec model learns hetero-

geneous spatial temporal representations from demand, geographic context, historic flow, and external events (Li et al., 2019). TAGCN combines graph convolution and temporal attention to predict station inflow and outflow (Zi et al., 2021). These works show, that graph based structures can improve bike demand modeling, but graphs are usually presented as aggregation of graph features in discrete time intervals referred to as discrete-time dynamics graphs (DTDG), rather than a stream of continuous timestamped events which avoids the aggregation of graph features and possibly produce more accurate predictions (Kazemi et al., 2020).

Spatio-temporal graph neural networks were also widely adopted in other transportation problems. DCRNN and STGCN models combine temporal modeling with graph based spatial propagation, which enables capturing patterns across traffic sensors connected via the transport network (Li et al., 2017; Yu et al., 2018). Other techniques, such as GraphSAGE, allow capturing localised relationships patterns with a model that learns node representations by aggregating features from local neighbourhoods (Hamilton et al., 2017).

Temporal graph learning is an alternative to spatio-temporal graph network modeling which avoids explicit construction of graphs and instead uses event processing. A step towards event processing was first made by JODIE, which models temporal interaction networks by learning dynamic embedding trajectories for interacting entities (Kumar et al., 2019). This was followed by DyREP, which models dynamic graphs through temporal point processes capturing both structural evolution and interaction activity (Trivedi et al., 2019). More general modular framework, based on node memory, message functions, aggregation, and temporal embedding, is the TGN model (Rossi et al., 2020). TGN maintains node memories, which are updated by temporal events, thus allowing each node to summarize its history in a compact way. This maps to the problem in this paper, where trips and redistribution events can be interpreted as timestamped interactions between stations.

3 Method

This section consists of two parts. In the first part the problem and data is described, followed by the description of a hybrid TGN architecture.

3.1 Problem Formulation

The main goal of the experiment is to evaluate whether modeling station to station event structure aids the model knowledge for predicting the number of available docks in the Bicing bike sharing network in Barcelona (Bicing Barcelona, 2026). The events exhibit daily and weekly patterns so the input window was set to $T_{in} = 168$ hours. We were only interested in short term predictions (next hour) so $T_{out} = 1$ hour. Every station is described by 8 features, of which six are temporal and two are static (see Table 1). Hourly features are normalised to interval $[-1, 1]$ using a sine transform, while the remaining features are normalised to $[0, 1]$. To integrate static features, each static attribute was repeated for every time step in the input window. This resulted in a unified time series representation across the participating stations. The objective is to forecast the number of available docks per bike station and is treated as a regression task.

Table 1. Node features with ranges before normalisation. Features marked in italic represent static features.

Feature	Range
Available docks	0 - 50
Holiday	0 or 1
Weekend	0 or 1
Hour of day	1 - 24
Hour of week	1 - 168
Hour of year	1 - 8760
Capacity	2 - 50
Altitude	0 - 175

The stations are interconnected via a continuous stream of events, which represent bike movements. An event can be represented as $e_{ij}(t) = (src, dst, tstmp, feat)$. The src and dst contain information about the participating stations. The $tstmp$ represent the event timestamp and $feat$ represents event features. In our case the event type is classified either as a trip, which represents flow produced by the users or a redistribution, which represents flow caused by human interventions in the system.

3.2 Hybrid TGN Model

The main component of the hybrid model is the TGN encoder that is applied on continuous-time dynamic graphs (CTDG), which are general representations of timed lists of events (Kazemi et al., 2020). They are modeled as a sequence of time-stamped events producing, for each time t, the embeddings of the graph nodes $Z(t) = (z_1(t), ..., z_n(t))$ (Rossi et al., 2020). The concept is based on event streams, which describe the interaction of two nodes at a specified time t. The TGN stores its internal node representations using a vector referred to as memory $s_i(t)$. Node memory is a learned hidden state involving the node in question and keeps track of the nodes history in a compact format. When $e_{ij}(t)$ arrives, two messages are generated, one for source node $m_i(t)$ and the other for the target node $m_j(t)$. A message for the source node is constructed using Equation 1:

$$m_i(t) = msg_i(s_i(t^-), s_j(t^-), \Delta t, e_{ij}(t)) \quad (1)$$

where $s_i(t^-)$ is the memory of node i before time t, Δt represents elapsed time of the event and msg_i is a learnable message function. Processing in batches usually involves multiple events using the same node i. In that case batch message aggregation function is used $\bar{m}_i(t) = agg(m_i(t_1), ..., m_i(t_b))$. Choices for agg can range from simple non-learnable function such as most recent message and mean message or complex learnable functions such as attention or recurrent neural networks. This is followed by an update of the node memory using Equation 2:

$$s_i(t) = mem(\bar{m}_i, s_i(t^-)) \tag{2}$$

To avoid the memory staleness problem, the embedding module is used to generate temporal embedding $z_i(t)$ of node i by sampling nodes reachable within k graph hops before time t using Equation 3:

$$z_i(t) = \sum_{j \in_i^k (0,t)} h(s_i(t), s_j(t), e_{ij}) \tag{3}$$

where h is a learnable function and is implementation defined. In our experiments $h = emb(i, t)$, so we used memory of node i before time t to produce the embeddings (Rossi et al., 2020). An illustration of the TGN encoder can be seen in Fig. 1.

To utilise temporal features of stations a hybrid TGN is proposed. The TGN model is designed to utilise the events, but we also wanted to utilise temporal node features. The hybrid TGN model is extended with a separate TimeConvEncoder for encoding station temporal data, the TGN memory is used for encoding event history representation and a small FutureEncoder is utilised to include known-ahead temporal covariates that describe the forecast timestamp, such as calendar and cyclic time features, and are available before the target station occupancy is observed. The outputs of all three encoders are concatenated to produce station state forecats. An illustration of the pipeline can be seen in Fig. 2.

The TGN encoder can be upgraded to include local node awareness using k-hop expansion, where k specifies the number of local neighbourhoods to be searched. We used GraphSAGE as a neighborhood aggregation layer over the node representations produced from TGNEncoder, TimeConvEncoder, and FutureEncoder. In some cases, the event stream might have a wider local effect, so it makes sense to explore the graph neighbourhood and aggregate the results. GraphSAGE enables learning of node embeddings by aggregating feature information from local graph neighbourhoods producing nonlinear aggregations (Hamilton et al., 2017). A similar effect can be reached by replacing h in Equation 3 with Temporal Graph Attention (ATTN) embedding block (Rossi et al., 2020).

4 Results

There are 548 statios evaluated in the data (Fig. 3), but the RMSE is evaluated only on 501 stations. The remaining 47 stations have special statuses, were closed down, have incomplete data or were reorganised, but can still participate in the flow. The dataset covers the period between January 2023 and March 2025, so there is 2 years and 3 months of data available. The events exhibit daily, weekly and yearly patterns which were confirmed with spikes on corresponding lags when the data was examined using the autocorrelation function (ACF).

After slicing the dataset into windows of size $T_{in} + T_{out}$ we obtained 27147 windows. There were 102410 unqiue origin-destination pairs across the full dataset, including approximately 2536 redistribution origin-destination pairs. The first 80 % of windows were used as a training dataset, the next 10 % was used to represent validation dataset, while the remaining 10 % was used to represent the test datasets. After some preliminary testing on the split dataset, we settled with $T_{in} = 168$ because longer windows did not present any improvements while shorter windows negatively affected the forecasts. We were only interested in next hour predictions, so $T_{out} = 1$. Preliminary feature ablations showed that the retained feature set produced the lowest validation RMSE among the tested configurations so all trained models use the same feature representation - Table 1.

The models were implemented in Python 3.11 using Pytorch Geometric and trained on cuda capable device Nvidia GeForce 3060 12 GB GPU. The hyperparameters were chosen based on experimental results feedback, where we prioritised RMSE and training time.

Each model was trained three times with different random intialisations, and the reported results are the means performance over the three runs. Training was run for a maximum of 10 epochs using the Adam optimiser with a learning rate $\eta = 0.001$. During training, validation performance was monitored and the checkpoint with the lowest RMSE was retained for final evaluation on the test set. Best checkpoint was always reached before the full 10 epochs, so the report results are based on early selected model states.

The hybrid TGN model consists of 3 encoders (TimeConvEncoder, TGNEncoder, FutureEncoder) and 1 decoder (OutputDecoder). It consists of two convolutional layers, each followed by a ReLU activation function. After the final activation, the resulting feature representations are passed through mean pooling and a linear layer, which outputs a latent representation with the dimension of 64. The FutureEncoder is used for known future context extraction and consists of a linear layer with a ReLU activation function, followed by another linear layer that produces an embedding of size 16. The function of the TGN encoder was discussed in Section 3.2 and uses events. Finally, the embeddings

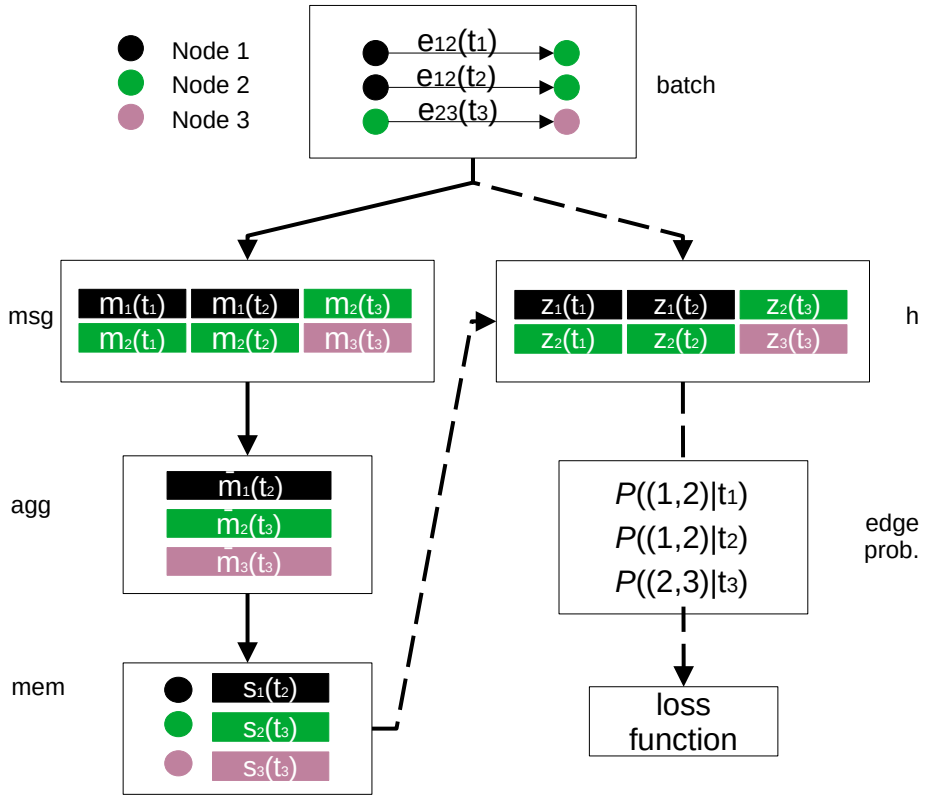

Figure 1. Simplified illustration of a TGN encoder module. Multiple events usually involve the same nodes in the batch. Firstly the left branch (solid black line) gets executed, where event representations are transformed to compact messages. Because a single node can be presented with only one message within the batch, the messages need to be aggregated. In this example the agg always takes the last message. Finally, the node memories get updated. Secondly the right branch gets executed (dashed black line) which creates embeddings using current node events and nodes historic context from memory. The embeddings are used to make edge probability predictions in given t. This is fed to the loss function, which produces gradients that are used to update trainable parameters.

from TimeConvEncoder, FutureEncoder and TGNEncoder are concatenated and passed through the OutputDecoder, which consists of a linear layer followed by a ReLU activation. The last layer is again linear and produces the final prediction of station occupancies in the next hour.

To determine whether using the event processing pipeline improves model performance, a baseline model was designed first that operates solely using per station features. The baseline model consists of TimeConvEncoder, FutureEncoder, and OutputDecoder, and achieved an RMSE of 3.07. In the second experiment, we utilised the hybrid TGN to include the events, but the neighbourhood search was not included is resulted in an improved RMSE of 2.99.

We speculated that including context from local neighbourhoods could have an impact on the model performance, because a flow from one station can influence the flow at the other nearby stations. For instance, if a station was full, the overflow would be redirected to neighbouring stations. We experimented with inclusion of direct, 2-hop and 3-hop neighbourhoods. The best results were achieved by direct neighbourhood integration, that achieved an RMSE of 3.02, but fell short of beating the model from the previous experiment.

In the final experiment we addressed the problem of exogenous shocks in the form of redistribution events. In the previous experiments a flag was included to indicate whether an event represented a redistribution or a trip. We assumed, that the model might get confused if we feed the events to it in this fashion so the idea of a dual hybrid TGN model emerged. The idea was to group events by their type (trips or redistribution). In the last experiment we therefore trained a dual hybrid TGN model, which maintained two independent memories, one for redistribution and the other for trips. For the final prediction, the two memories were concatenated together. However, the performance degraded further, and the model achieved an RMSE of 3.04. The summary of experiments can be found in Table 2.

The results indicate that stations, own temporal and static features probably do most of the work. Adding the events improves the performance slightly, so TGN provides some context signal learned from the events.

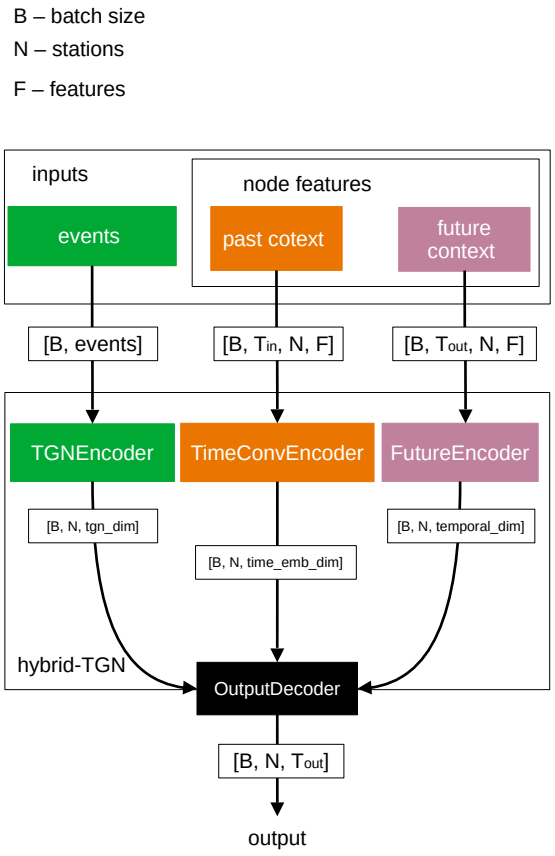

Figure 2. Illustration of a Hybrid TGN model, composed of three encoders. The TimeConvEncoder encodes station history, TGN encodes events, and FutureEncoder encodes known future context. Outputs from encoders are concatenated along the temporal axis and are fed to the OutputDecoder, which produces a forecast for the given horizon T_{out}.

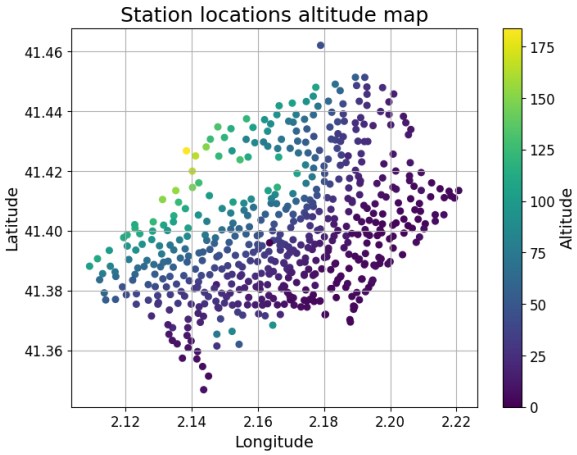

Figure 3. Bike sharing network for the city of Barcelona. The colormap represents station altitudes.

Including the local neighborhood aggregation and dual memory, on the other hand, resulted in worse performance than the simple TGN, but both remained slightly better than the temporal baseline. This indicates, that event information is useful but using more complex models may increase the noise or make optimization harder so extra architectural complexity is currently not helpfull.

Table 2. Model experiments indicate, that all TGN model variants benefited from the additional event information.

Model	No. of params	RMSE
Baseline	30000	3.07
TGNMemory	78769	**2.99**
TGNKHop	142897	3.02
TGNDualMemory	190161	3.04

The hybrid TGN introduces some additional computational overhead compared to the temporal baseline model because the event stream must be loaded, mapped to station indices, and used to update node memories for each training window. This cost is manageable for the direct event memory formulation, but becomes more pronounced in the TGNKHop variant, where events are expanded into local neighbourhoods before training. This presents a big computational bottleneck and increased the preprocessing time, while not improving RMSE over the simpler memory based TGN.

5 Conclusion

This paper reports the results of a case study on how useful temporal event patterns can be to forecasting of bike station occupancy. The experiments indicate that additional model complexity may not automatically improve performance. The simplest hybrid TGN model without neighbourhood aggregation achieved a lower RMSE compared to the baseline temporal convolutional network, which indicates that trip and redistribution events contain some additional patterns that can positively contribute to model performance, although most of the structure was explained using the temporal and static features of bike stations. The simplest hybrid TGN approach, which combined station temporal encoding, event encoding, future encoding and the output decoder, achieved the lowest RMSE. More complex model variants that added the GraphSAGE neighbourhood aggregation and dual memory redistribution and trips, fared worse.

Overall, the results suggest that event memory is valuable, but the most effective use of events is a TGN memory representation rather than explicit graph neighborhood aggregation or memory separation. Future work should therefore focus less on increasing architectural complexity and more on understanding how temporal memory is updated, how event types con-

tribute to predictions, and under which station or time conditions event information provides the greatest benefit.

Better forecasts could lead to reduction of unnecessary redistribution trips that could affect the emissions, noise, and operational wear. Future work should evaluate whether event-aware forecasts can reduce such trips through improved redistribution optimization, and also strengthen the evaluation protocol in three directions: reporting variability across random seeds, complementing RMSE with MAE and threshold-based empty/full station metrics, and using seasonal or rolling temporal holdouts to test robustness beyond a single final chronological split. These additions would better align the evaluation with the operational goal of identifying critical station states.

Acknowledgments

This research has received funding from the Slovenian Research and Innovation Agency under Research Project J7-70247. We would also like to thank Serveo for providing the Bicing bike sharing network dataset.

References

Bicing Barcelona. (2026). *Bicing, el servei de bicicleta compartida de barcelona*. Retrieved June 10, 2026, from https://www.bicing.barcelona/

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*.

Hulot, P., Aloise, D., & Jena, S. D. (2018). Towards station-level demand prediction for effective rebalancing in bike-sharing systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 378–386. https://doi.org/10.1145/3219819.3219873

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., & Poupart, P. (2020). Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, *21*(1).

Kumar, S., Zhang, X., & Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1269–1278. https://doi.org/10.1145/3292500.3330895

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Graph convolutional recurrent neural network: Data-driven traffic forecasting. *CoRR*, *abs/1707.01926*. http://arxiv.org/abs/1707.01926

Li, Y., Zhu, Z., Kong, D., Xu, M., & Zhao, Y. (2019). Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. https://doi.org/10.1609/aaai.v33i01.33011004

Lin, L., He, Z., & Peeta, S. (2018). Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, *97*, 258–276. https://doi.org/https://doi.org/10.1016/j.trc.2018.10.011

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. M. (2020). Temporal graph networks for deep learning on dynamic graphs. *CoRR*, *abs/2006.10637*. https://arxiv.org/abs/2006.10637

Trivedi, R., Farajtabar, M., Biswal, P., & Zha, H. (2019). Dyrep: Learning representations over dynamic graphs. *International Conference on Learning Representations*. https://openreview.net/forum?id=HyePrhR5KX

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 3634–3640. https://doi.org/10.24963/ijcai.2018/505

Zi, W., Xiong, W., Chen, H., & Chen, L. (2021). Tagcn: Station-level demand prediction for bike-sharing system via a temporal attention graph convolution network. *Information Sciences*, *561*, 274–285. https://doi.org/https://doi.org/10.1016/j.ins.2021.01.065