Delay Tolerant Networks in underground mines using Machine Learning Techniques

Chandrashekar Akkenapally

September 6, 2024

1 Introduction

Location prediction has been a well researched problem in the realm of traffic forecasting and traffic prediction. Although there has been a lot of interest in how traffic can be forecasted in shopping malls or over the land, there has been little research as to how can location be predicted for someone who is present underground. The methods for this can be manifold. One such use case is for optimal resource allocation in case of disasters. In specific cases, such as a disaster that might occur in underground mines, there are multiple cases in which a single fracture or failure can lead to landfalls in different parts of the mine, the area in a mine is spread can be wide and hence spending resources over the whole mine is not optimal or efficient. To get a better understanding of which locations to target in such as disaster, the location prediction method could be vital and of great use. For location prediction in underground scenarios, simply a way in which a solution that has great results over the land can be used for underground too such as in underground mines but with there are multiple issues with the same.

1.1 Wired vs Wireless systems

Location prediction using wireless systems such as handheld devices which can register GPS signals has been a well though topic, but these solution do not work in case of underground mines or underground settings owing to the inability of a set method for using locations and registering coordinates. The (0, 0) coordinate for a coal mine present in Tenesse would be different than that of the another metal mine present in Colorado.

Another issue that is present is the communication architecture in itself. In underground communication scenarios, cables such as leaky feeder systems have been used widely to communicate between different people or equipments, but in case of any disaster, the whole wired system would breakdown and hence cannot be used. This makes the case for using a system which can be used for communication even when there are different disasters that occur and to know specific locations.

Wireless devices such as DTN devices have shown to do great work incase of turbulent scenarios. They have found to be of great using in space station, space sateliets and battlefields too. They work on the "store and forward" framework. Delay/Disruption Tolerant Networking (DTN) is a suite of standard protocols that use information within the data stream (headers attached to data units) to accomplish end-to-end data delivery through network nodes. DTN enables data delivery in situations that involve

- Disconnections (e.g., end-to-end link unavailability)
- Delays (e.g., Deep Space missions)
- Data rate mismatches (e.g., high data rate Science downlinks but lower rate terrestrial connections)

A DTN architecture is a store-and-forward communications architecture in which source nodes send DTN bundles through a network to destination nodes.

The DTN protocol suite can operate in tandem with the terrestrial IP suite or it can operate independently. DTN provides assured delivery of data using automatic store-and-forward mechanisms. Each data packet that is received is forwarded immediately if possible, but stored for future transmission if forwarding is not currently possible but is expected to be possible in the future. As a result, only the next hop needs to be available when using DTN.

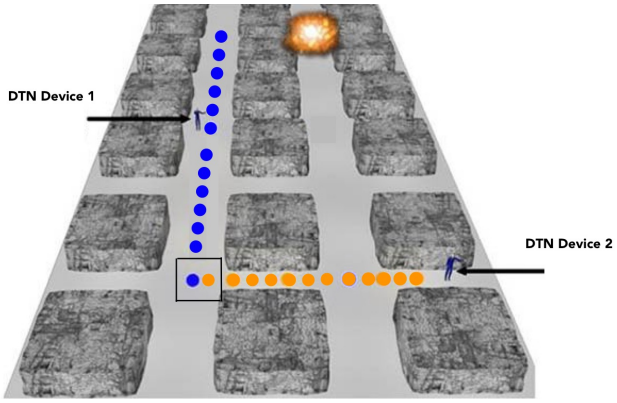

Figure 1: Communication between 2 miners

Fig 1, shows the communication between 2 different miners and how they connect with each other. When the miners are moving in underground mines, in wireless systems or connections, they are not always connected but moves and come in connection when they are at a specific location. When the nodes are connected, they would share messages and data that is senses, which can later be used for different kinds of activites.

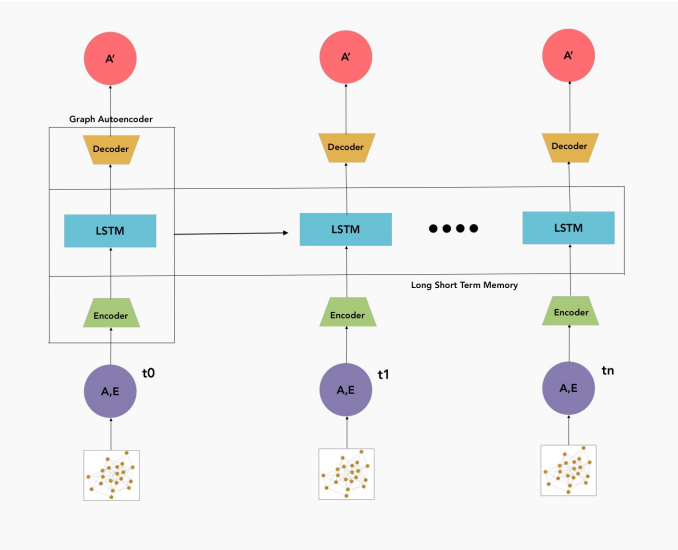

Figure 2: Architecture

1.2 Delay Tolerant Network and Problems

Delay tolerant networks are a research subject on their own; however, they are quite important in many of challenges related to energy efficiency and green communication. Since many of the networks are power constrained, rely on batteries, and often involve mobility, there is a high chance that at some point the network will become partitioned. When part of the network cannot communicate with the rest, a subset of the nodes may be cut off from communicating with the Internet or other intended destinations within a subnet. Let's consider a network where sensor nodes were dropped randomly from a plane. Eventually the nodes on the routing path will use their battery faster than the rest of the nodes and a partition in the network may occur if nodes on each side of the path are outside of communicating range of one another. Another problem delay tolerance tries to address is high error rates within communication networks. It has typically been taken for granted that communication will occur without errors, however with wireless networks and increasing use of the spectrum, this will become more of a problem all the time. The last problem is that delay tolerance can address is long or variable delay within networks. It has traditionally been assumed that propagation delays are relatively homogeneous within networks, however with long haul wireless links, intermittent connections, and high error rates this is not always the case. It may take some time for a successful communication to occur. In this project we are focusing on the last problem that is to reduce the delay within the network. In this we should be able to predict the future position of the node precisely which helps us to transfer the data to selective nodes and to transfer the data in a dynamically varying ef-

ficient path. This eventually will decrease the amount of power consumed by each node immensely, reduces the error in communication, and data loss.

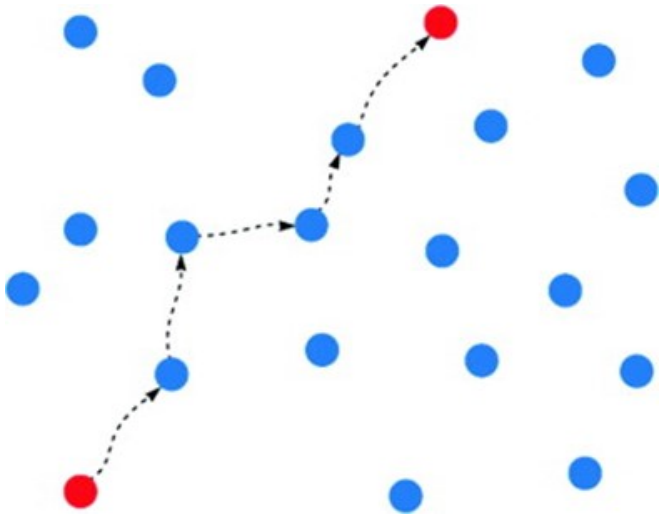

Figure 3: Routing a message from source to destination in DTN

2 Related Work

There are multiple works that have been done to show how GNN or Graph Neural Networks can be used for Traffic Forecasting. Defu Cao et al.[2] used Spectral Theory to forecast the traffic. They used their train data as the image and the current location as inputs, the image was then passed through a VGG16 model which allowed for extraction of features and then this was coupled with the other features to be passed through a Spec-STN layer and then predicted the location. The location they predicted was in the (x, y) format and did not make specific uses of category of location etc. Other such work was also the SAST-GNN by Y Xie et al[1]. In their work, they used attention mechanism, a feature that was first proposed by GAT, to give specific weights to both spatial and temporal features simultaneously and features are extracted from the same.

Another Hierarchical framework was introduced by Zirui et al., where they focused on how the hierarchical relationship would be used to understand and make things better. These works have used the GPS information for the prediction of location. Although they are very well and work well for scenarios which might have GPS, if there are other such areas in which there is not GPS, these methods would falter and would not work.

As a new approach to routing within a network which may adhere to deterministic schedules as well as be subject to uncertain disruptions, we propose a machine learning based routing framework. This solution will provide the

adaptive benefits found in opportunistic routing strategies, while still adhering to user specified constraints which often exist within interplanetary networking. To accomplish this, we have used miner GNN machine learning model. In the past there are other machine learning models which have been used in DTN routing for various applications such as to Reduce Overhead in DTN Vehicular Networks.

A. *Reinforcement Learning*

Reinforcement learning is a commonly used machine learning algorithm in which the learner discovers how to achieve a desired outcome by maximizing a numerical reward [6]. Reinforcement learning systems typically consist of four elements: a policy, a reward function, a value function, and in some cases, a model of the environment [7]. Q-routing is an adaptation of the Q-learning algorithm developed for packet routing [5]. Q-routing uses the estimated end-to-end packet delivery time for the basis of its reward table, or Q-table. The table contains a row for each neighbour that a node has. Each column corresponds to a destination node. The entry for the row-column pairs in the table is the estimated time required for a packet to be received at the destination if it was sent from one of the possible neighbouring node choices.

Figure 4 shows a single node in the network and links to each of its neighbouring nodes. Its Q-table contains a row for each link to a neighbour and a column for every possible destination in the network. The index of each column corresponds to each node address. The entry corresponding to the node's own address is given a value of 0 (or some other indication of an invalid value), since it will not transmit data to itself. There are several approaches that can be taken for the initial estimate. All entries may be initialized to zero or a random value. Alternatively, a method can be developed to try to calculate an initial estimate of the end-to-end delays. The learner will determine what neighbouring node to send a packet to based on which node minimizes the delivery time. Once the packet has been sent to the chosen neighbouring node, the neighbour will reply back with what it believes the remaining time will be to deliver the packet to its final destination. This response will be used by the first node to update its Q-table. Each update should incrementally improve the accuracy of the Q-table, since nodes closer to the destination should have a more accurate idea of the remaining delivery time [5].

Pseudo-code for the Q-routing algorithm is shown in Figure 5. The variable represents the learning rate, t is the transmission delay over the link from node x to node y, and q is the queuing delay at node y. When a packet arrives it enters the node's inbound queue. The Q-routing algorithm will compare the Q-values (delivery time estimates) for transmitting the packet to its destination via each neighboring node and select the neighbor with the smallest Q value. When the packet arrives at its destination, the re-

ceiver responds back with its own estimated delivery time. This is then used to update the Q-table entry corresponding to that destination.

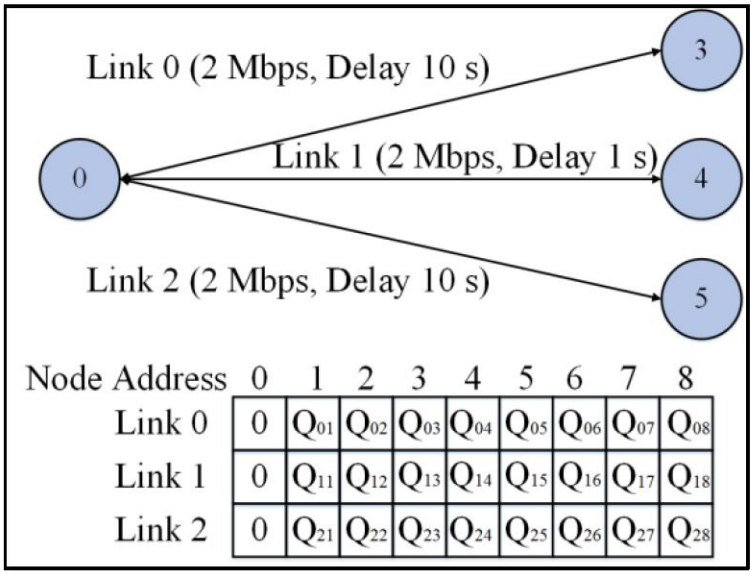

Figure 4: **Node with 3 Links and Corresponding Q-table**

B. *Bayesian Learning*

The concept of Bayesian machine learning is based on the conditional probability that a certain outcome has some likelihood given that it possesses a particular set of attributes. In particular, this paper focuses on the Naïve Bayes classifier. This learning method is used to classify a new instance or occurrence within a set of possible values based on previous training data. The learner will determine the probability that a certain set of attributes most likely correspond to a specific classification within the training data. When a new occurrence is presented to the learner, the training probabilities are used to determine the value v of the new instance from a finite set of values V based on its attribute vector [6]. Bayesian learning is based on calculating the most probable outcome, often called the maximum a posteriori or MAP hypothesis. The Naïve Bayes classifier attempts to find the most probable value for a current instance VMAP given its known attributes [6]. Equation 3 calculates the MAP hypothesis as the probabilities of observing the value vj in conjunction with the attributes. This is easily found by taking the product of the conditional probabilities of observing vj given each individual attribute.

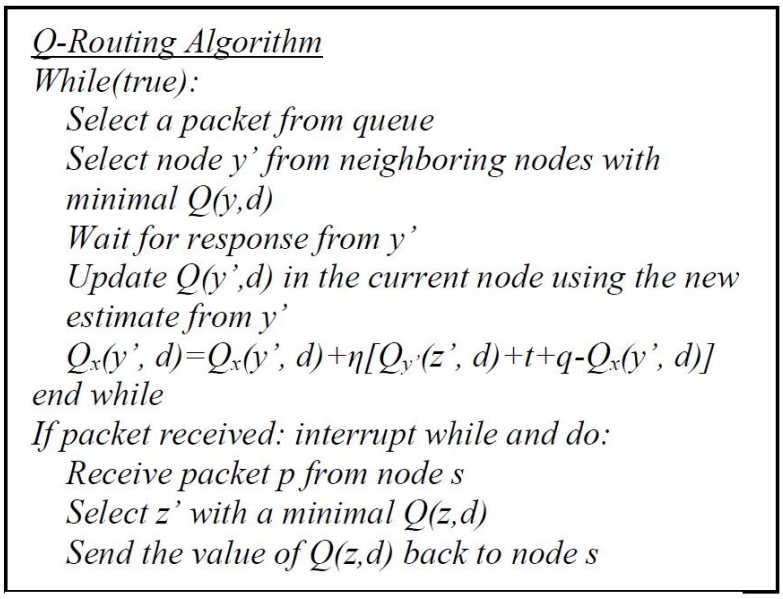

Figure 5: **Q-Routing Algorithm**

Naïve Bayes classification and decision tree learning have been proposed in [**?**] to opportunistically select available communication channels for cognitive radio sensor networks.Context information such as neighbouring nodes, sink nodes, current time slots and the currently available channel set are used to predict link connectivity. An optimal routing path is obtained from the consideration of two classifiers which predict link stability.

C. *Naive multi-copy routing algorithms*
Epidemic is the fundamental multi-copy, high overhead, routing algorithm [9]. If applied without any control, nodes transmit in every opportunity to others that do not have a copy, like an epidemic disease. Despite of that, the overhead can be small and delay / delivery performance better in low density scenarios and sparse networks.

D. *Naive algorithms with flood control*
In order to reduce network overhead, basic flooding control measures can be applied. Some approaches aim at limiting the time to live of messages (TTL), the number of hops, or tuning the message space storage. However, implementing these approaches require intense parameter tuning, not always trivial to implement due to DTNs' dynamic behavior. Some other approaches are inspired in the analogy of vaccines, where once an "anti-packet" is received it induces the node to reject or delete a copy of a specific message [10]. Finally, Spray And Wait (SaW) [8] is another solution which sets a maximum number of copies per original message that can

$$V_{MAP} = \max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n) \quad (1)$$

Bayes rule can be used to write Eq. 1 as:

$$V_{MAP} = \max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \quad (2)$$

This simplifies to:

$$V_{MAP} = P(a_1, a_2, \ldots, a_n | v_j) P(v_j) \quad (3)$$

$$V_{NB} = \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) . \quad (4)$$

exist in the entire network. This value is decreased at every copy transmission. When it equals to one, nodes only transmit directly to the target node. In summary, the goal of all the above solutions is to reduce network overhead without any other consideration about other performance issues. Consequently, the performance of these algorithms is quite dependent on the mobility model and nodes' spatial distribution.

E. *Statistical and context information algorithms*
Other solutions exploit statistical and context information to select the best intermediate nodes. Nodes use utility functions to qualify intermediate nodes and to decide if they should transmit a copy or not. Commonly, mobility statistics and connectivity information is used since the latter has an influence on performance. The knowledge of network connectivity properties and the qualification of nodes allows the design of more effective routing protocols for specific environments [11]. For example, the Lobby index (Lindx) [12] is a two-hops connectivity metric suggested by [13] to qualify nodes. It detects dense zones where highly connected nodes are located. To compute the Lindx in a network, every neighbour from a node x reports to x its connectivity degree. Then, a node x calculates the maximum integer l such that x has at least l neighbours with a value of degree greater than or equal to l. For Lindx to be applied, an additional and exclusive channel for beaconing transmission is needed.

F. *Node encounter probabilities algorithms*
ProPhet [14] and MaxProp [15] are algorithms that employ utility functions to take advantage of information about node encounter probabilities. However, besides mobility information, nodes can collect other kind of information to find network behavior patterns. For instance, the time interval of the day could be related with the network load. Thus, machine learning algorithms were used with the purpose of detecting convenient opportunities for message transmission according to the learned transmission patterns. Basically, machine learning based approaches consist in

collecting data of transmissions, processing the data conveniently and use a trained classifier to support future transmission decisions.

G. *Classification algorithms.*
Classification consists in attributing objects to one of many categories or classes. Formally, according to [16], classification is the task of learning an objective function f that maps each set of attributes to one of the predefined classes. Thus, a classifier constructs a classification model from a dataset (training dataset) using a machine learning algorithm. Given a set of n attributes, a, b, ..., n, and a set of m classes, C1, C2, ... , Cm, a training dataset is a set of tuples where each tuple corresponds to one class Ci, where 1 i m: avalue, bvalue, ..., nvalue→Ci, where avalue, bvalue, ..., nvalue are attribute values and Ci is the class value.

Using the classification model, the trained classifier assigns a class to an unannotated tuple which has only a set of attribute values, avalue, bvalue, ..., nvalue. Also, the classifier must have the capacity of generalization, this mean that the classifier must assign a class even for unannotated tuples that were not seen in the training dataset.

H. *Routing and broadcasting using classifiers*
There are still few approaches that suggest the use of classifiers to support routing and dissemination decisions. [17] use the Naive Bayes classifier to be trained with information of past broadcast retransmissions for message dissemination. Afterward, the classifiers predict if a new broadcast attempt will be successful or not. Classifiers were also used to find mobility patterns from people movement to construct predictive models to forecast: "where", "how long will stay" and "who people will meet" people in different locations [18]. [19] exploit the repetitive behaviour of public buses movement to develop an extensible Framework and a single-copy DTN solution. As a matter of fact, this is the first work in DTN routing that use classifiers to improve routing decision, and it is the most related work to our proposal. It consists in keep a resident Naive Bayes classifier that uses information of end-toend past deliveries (thus, acknowledgment propagation is required) to determine the most likely nodes that will deliver a message to a specific target node. This scheme outperforms a single-copy version of MaxProp. However, when compared with Epidemic, its performance is lower in delivery probability and delay. The reason for that is that in scenarios of low to moderate density Epidemic flooding has usually the best performance in this metrics, but with high network overhead.

I. *Decision tree classification algorithms.*
The C4.5 decision tree [20] is a machine learning algorithm used for classification. The C4.5 training phase generates a decision tree using training data. The decision tree is composed by: (i) internal nodes that represent conditional operations on the attributes and (ii) terminal nodes associated to class attributes. Initially, all the tuples are attributed to the root

node of the decision tree. The next step consists in finding a conditional operation based on one of the tuples' attributes. This operation divides the root node into two or more internal nodes. The new internal nodes contain a subset of the root node's tuples. This division operation is repeated recursively until the generated nodes are pure. The purity of a node is a measure of the homogeneity of its tuples' class distribution. In terminal nodes, the most frequent class determines the class attribute. To choose the attribute and the conditional operation applied to it, all the possible options are evaluated and the one that maximizes the purity of the resulted subsets is used. With all in mind, the merit of the C4.5 comes from [16]: (i) it makes a automatic selection of the best attributes to create the structure of the decision tree and (ii) the decision tree is a comprehensible form to visualize the attributes's influence in the classification of tuples. J4.8 is the latest and slightly improved version, called C4.5 revision 8, implemented in Weka [**?**]. J4.8 was the last public version of this family of algorithms before the commercial C5.0 was released. The full complexity of decision tree induction is $O(\alpha N(logN))$, where α is the number of attributes and N is the number of instances [**?**].

3 Preliminaries

3.1 Data Generation

The data that was generated was made using Brinkoff Generator. It is a network based generator of moving objects. To emulate or make it similar to the DTN nodes, pauses were added to it to allow for it to be made in such a way that miners would sense data if they are mining or doing such activities. Similar to this was the different kinds of miners and their activities. 5 kinds of miners were chosen which would include, supervisor, engineers, miners and also some heavy equipments.

3.2 LSTM

Long short Term Memory(LSTM) has been used widely to solve the problem of location prediction and how can we use it for predicting location of groups too. The major drawback of this system is that the need for data is quite large. But with real life scenario as ours, we cannot have so much and in addition to that, by the time, we do get the data that it reaches the capability of predicting with some accuracy, the data is not relevant anymore. This leads to a large void to be filled by may other algorithms which can do better at prediction tasks using lesser data. This is where the other algorithms step in. There are other methods which have used

3.3 Random Forests

Random Forests are an area of Machine Learning(ML) that has been long been used for various prediction tasks. First proposed by Ho[b7], it was shown that these random forest which internally use tree-based classifiers for training sub-sample spaces would as a whole increase the prediction accuracy. This has shown to be a very good method for prediction. It was shown that with use of multiple trees, the problem of poor generalization could be overcomed and could be used for multiple other uses.

Random forests follows specific rules related to tree growing, the tree depth and because of this is also robust to overfitting and is stable in the presence of outliers in a very high dimensional parameter space. The concept of feature importance is an implicit feature selection method which is what the Gini Index does. The Gini Index measure the power of the variables in regression or classification and then based on the principle of impurity, splits the data into 2 parts.

$$E = -\sum_{i=1}^{n} p * log(p)$$

For splitting a node, the best way is to make the samples such that you get the best partition possible. This can be done by parameter tuning and understanding how to fix the different categories present so that the estimator can draw a line between the multiple classes available.

3.4 Markov Chain

Markov Chains have been used a lot for understanding patterns and how can one even lead to another. [3] use Markov Chains on continuous Time Series data to show how can it be used for prediction of locations. This way, it can be seen that, there are multiple different ways in which can be used. We also used it for our experiments to see, how the multiple different locations that the particular DTN device has been earlier leads to a better prediction accuracy and lower loss.

3.5 Spatial Temporal Information

The spatial and temporal information is encoded in a form of matrix which will be used as a channel later. The spatial information of how and where the node was at a particular given time, would allow us to encode that behaviour where the node has been and in addition to that it will allow us to see how the nodes move.

3.6 Graphs

3.6.1 Graph Environment

The environment present can be represented as a graph in itself. The environment has been made from the Brinkoff generator which itself used the graph

of Oldenburg to generate the nodes and edges which can be used as the environment. We denote the graph by $\mathcal{G}_\urcorner = (\mathcal{V}_\urcorner, \mathcal{E}_\urcorner)$. The $\mathcal{V}$ represents the vertices which represent the segment which the node visited during the day. The connections to it $\mathcal{E}$ denote the connections to it and how one of the places led to another. These would include self-loops too. To add the spatial temporal component, we add another set of matrices which allow us to encode the information in it.

3.6.2 Agents/DTN nodes

The nodes when move, transmit information between each other which can be then used for predicting the location of the miners after sometime. The idea behind this is that when the nodes move, the graph made by their motion can be overlapped with the environment map that has been created which would allow for learning and understanding what previous trajectories or path the miner might have taken which would allow for the model to predict the way the miner would take after this.

3.7 ONE simulator

The ONE is an Opportunistic Network Environment simulator which provides a powerful tool for generating mobility traces, running DTN messaging simulations with different routing protocols, and visualizing both simulations interactively in real-time and results after their completion. The ONE is a simulation environment that is capable of generating node movement using different movement models and routing messages between nodes with various DTN routing algorithms and sender and receiver types. In addition to that ONE has ability to visualize both mobility and message passing in real time in its graphical user interface.

4 Architecture

Fig 2, shows the architecture. In this model, the data that is passed is the adjacency matrix and also the edge features that are present. Both of them together make the train data. The representation that is learnt through the auto-encoder is then passed to the LSTM layer which then passes it on to the next layer. The ability of this model allows us to learn the representation and then move it forward with the RNN layers. There are GRU, RNNCell with which the experimentation has been done too. Each component is explained in further detail below

4.1 GAE

GAE or Graph Autoencoders learn the representation of the input data which can then be passed through a decoder to predict or reconstruct the, in this case, adjacency matrix. The idea origianlly generated by Kipf et al.[4] in 2016, learns

the representation of the features and also of the nodes that are present. They are together passed through a series of layers which allows for the encoder to learn the representation.

$$\hat{A} = \sigma(ZZ^T)Z = GCN(X, A) \tag{1}$$

Eq 1, shows the encoder and decoder learning and latent variables that are learnt from the adjacency matrix A and the features X. Both of them are together used to learn a latent variable Z which can then be used for generation.

One goal of autoencoder training is to minimize the generalization error, which reflects the generalization ability of an autoencoder. It is reasonable to assume that the training set can represent the whole input space. We propose a simple training strategy to create new training samples based on the existing training set by adding perturbation to original training samples. As for a given training sample, its adjacency matrix is filled with one and zero. If an element is one, it suggests the corresponding pair of nodes are connected by an edge. Similarly, zero indicates the corresponding pair of nodes are not connected. Note that we only focus on the edges between different nodes, so the elements of the diagonal of the adjacency matrix are ignored. Here, we design a training strategy for sparse adjacency matrixes, while this strategy can also be adjusted for dense adjacency matrix.

4.1.1 Embedding

Each of the nodes are a combination of 3 different parts. The first part is the one hot encoding of the node itself. This is followed by the edge features that are present. The third part is the one-hot encoding of the other node to which this edge connects to. These 3 nodes are concatenated and then embedded to get an embedding which would allow for the nodes to learn the inductive representations. Assuming that there are T different $N \times N$ matrices that are made which encode information for the 3 different features that would lead to the current node. The resultant edge matrix that is formed is $3 \times T$.

$$x_t^0[i] = concat(x_i, f_i, x_j) \tag{2}$$

4.1.2 Encoder

Once, the embedding has been made for the data, the model then encodes or learns the information. This information includes, the nodes or segments this miner has visited before and what are the features of the movement of the miner/DTN node i.e. speed, angle and time which lead to the current location it is at. All of this information is passed and learnt by the model. Multiple layers can be added to learnt much more about the previous information that is present.

4.1.3 Decoder

The encoder is then followed by a bilinear decoder that has been used multiple times before to forecast or predict how the nodes can be found.

5 Implementation and Algorithm

There are various stages in process of deploying the complete environment and implementation of algorithm.

Algorithm 1 Data extraction and aggregation after transfer from miner

: $speed[1, \cdots, n], time[1 \cdots, n], angle[1, \cdots, n] \in D$. : hidden representation$[h_{n+1}]$, cell representation$[c_{n+1}]$.Partition the data into T time horizons selected t in T d in D d is in time horizon t add data to the current time horizon t add data to next time horizon $t + 1$ i = 1 to T $[l_1, ..., l_n]$ extracted locations the miner was at. $[h_1, ..., h_n]$ the previous hidden representations for location for the miner. $[c_1, ..., c_n]$ the cell representations for location for the miner. i = 1 to N Combine the extracted locations i with the data extracted according to time horizon T with the hidden representations of location i using LSTM aggregator. h_{n+1}, c_{n+1}

5.1 Training the GNN model

As mentioned above in the initial state we are trying to train our GNN model with the basic data set consisting of information about various nodes and their parameters such as x coordinate, y coordinate, speed, edge, angle etc. This initial data set is generated from the Brinkhoff generator. The Brinkhoff moving object generator, which simulates mainly physical aspect of object mobility, generates moving objects based on road network. The most important properties of the moving objects in Brinkhoff generator are the speed limit of the object classes, the road speed limit, the maximum capacity of the road segments, the influence of other moving objects on the speed and the routing of the object, the influence of external events and time scheduled traffic. The start and end nodes of the trips are determined by the road network density or region-based approach. These characteristics are the basic specifications for the generation of spatio-temporal data.

An interactive interface is designed for users to control the generation of the features of the resulting dataset which is shown in figure 6. For example, the maximum timestamp to generate objects, number of generated moving objects and external objects per timestamp, classes of moving objects and external objects, report probability, and so on. Users can control the behaviour of the generator by varying the parameters in a control file for the generator, for example, limitation of the minimum and maximum timestamps, definition of the location for road network storage in the computer, and settings of the map

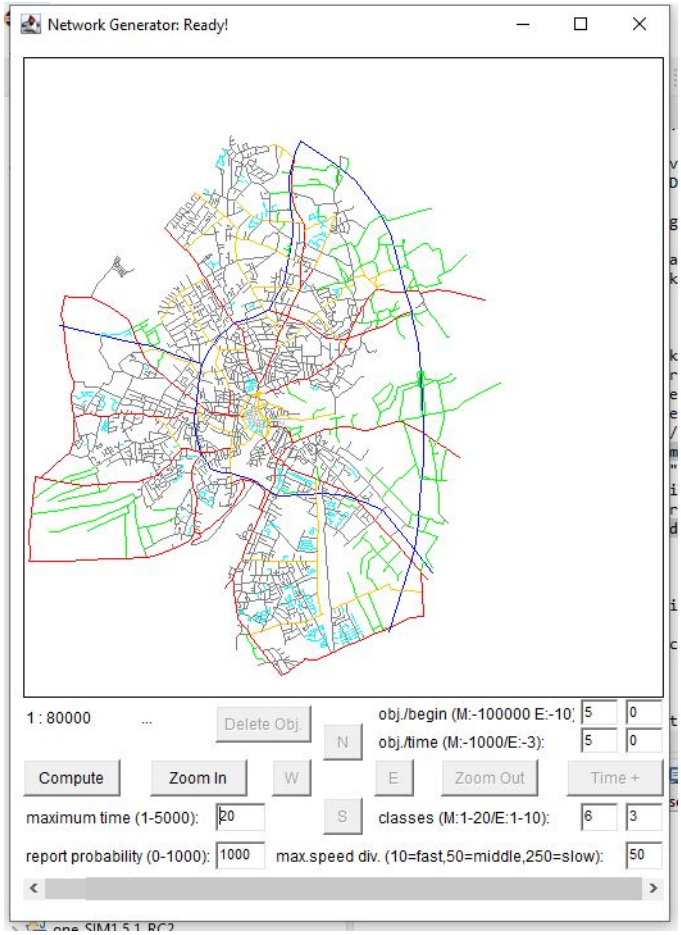

Figure 6: Graphical User Interface of Brinkhoff Generator

viewer. The generation of the resulting dataset is based on real road network data with user-defined properties.

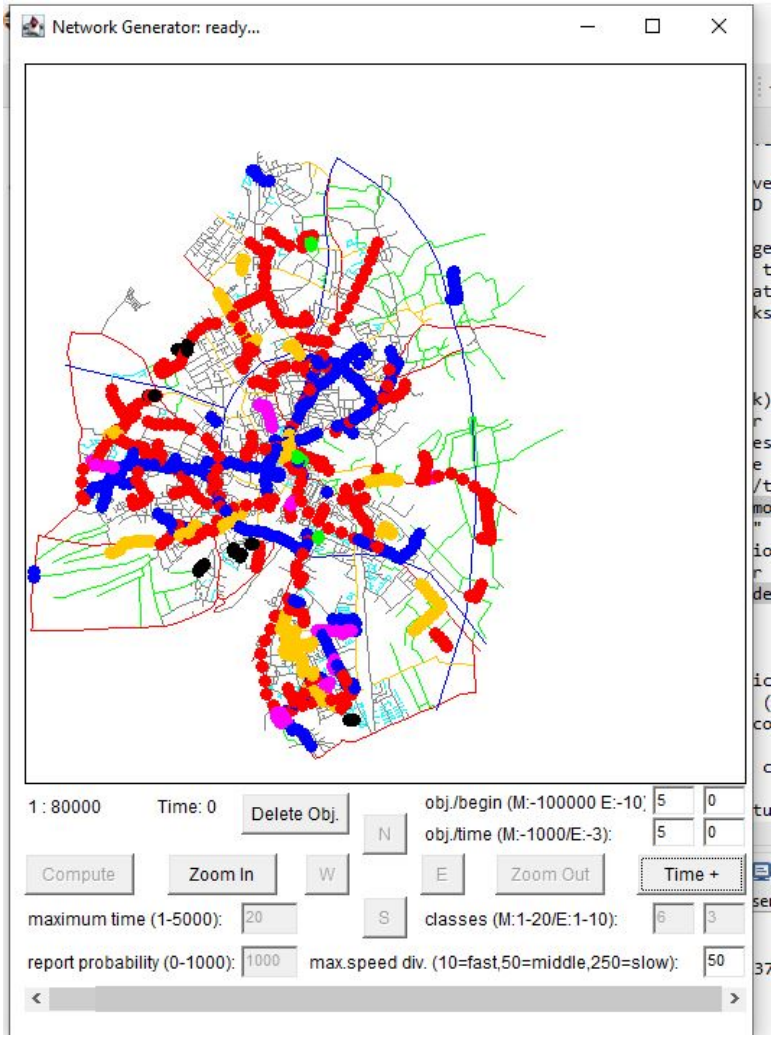

Figure 7: Brinkhoff Generator after the node movement is computed

Once the movement is computed, Fig 7, or triggered in the Brinkhoff generator the output generated in the form of a simple text file which consists of various properties of the node and its movements. As sample output from the Brinkhoff generator is shown in figure 8.

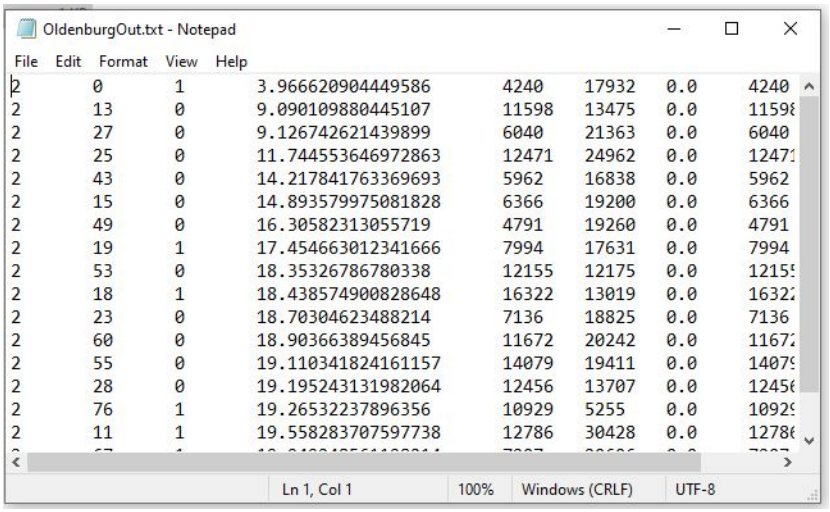

Figure 8: Brinkhoff Generator sample output

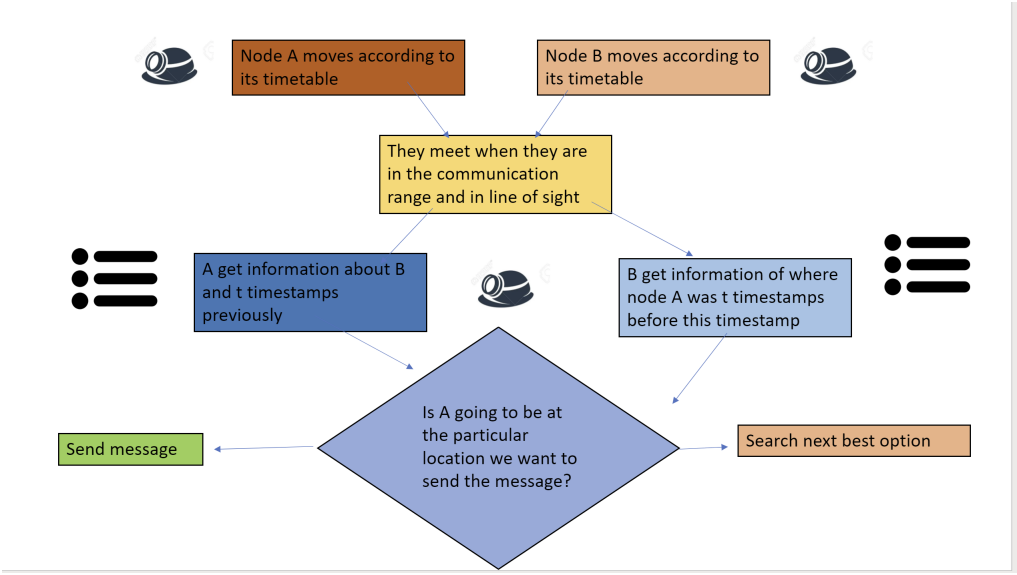

Figure 9: Communication of miners and transmission of information

5.2 Movement model configuration in ONE

The designed machine learning model has been deployed using the version 1.6 of The Opportunistic Networking Environment (ONE) simulator. The ONE simulator has a movement model which decides how the nodes should move while simulation and this model is altered with changes according to our GNN machine learning model. As part of the movement model, we are trying to move the nodes in different directions and in their travel path they meet with other nodes. At this is point we are precisely choosing the DTN agent or node with which we are going to exchange the data, by doing so we can achieve buffer memory management. The decision to choose specific DTN agents is made by the GNN model which decides based on the futuristic path of the DTN agent. This prediction of the future path is made with historic locations data of the DTN agents. This data is fed as input to the model which will be predicting the path and direction in which the DTN agents might travel further. The movement models of ONE simulator govern the way nodes move in the simulation. They provide coordinates, speeds and pause times for the nodes. The basic installation contains, e.g., random waypoint, map-based movement, shortest path map-based movement, map route movement, and external movement. They are considered as parameters by our GNN model for predicting future position of nodes, deciding path to travel and avoiding nodes for information exchange. All these models, except external movement, have configurable speed and pause time distributions. A minimum and maximum values can be given, and the movement model draws uniformly distributed random values that are within the given range. Same applies for pause times. In external movement model the speeds and pause times are interpreted from the given data.

There are various movement models which are available in the ONE simulator such as the random waypoint based, map-based and shortest path-based movement model. When a node uses the random waypoint movement model (RandomWaypoint), it is given a random coordinate in the simulation area. Node moves directly to the given destination at constant speed, pauses for a while, and then gets a new destination. This continues throughout the simulations and nodes move along these zig-zag paths. Map-based movement models constrain the node movement to predefined paths. Different types of paths can be defined, and one can define valid paths for all node groups. By this way for example DTN agents can be prevented from entering into the restricted or warning zones within the mines.

The basic map-based movement model (MapBasedMovement) initially distributes the nodes between any two adjacent (i.e., connected by a path) map nodes and then nodes start moving from adjacent map node to another. When node reaches the next map node, it randomly selects the next adjacent map node but chooses the map node where it came from only if that is the only option (i.e., to avoid going back to where it came from). Once node has moved through 10-100 map nodes, it pauses for a while and then starts moving again. The more sophisti-

cated version of the map-based movement model (ShortestPathMapBasedMovement) uses Dijkstra's shortest path algorithm to find its way through the map area. Once a node reaches its destination, and has waited for the pause time, a new random map node is chosen, and node moves there using the shortest path that can be taken using only valid map nodes. For the shortest path-based movement models, map data can also contain Points Of Interest (POIs). Instead of selecting any random map node for the next destination, the movement model can be configured to give a POI belonging to a certain POI group with a configurable probability. There can be unlimited number of POI groups and all groups can contain any number of POIs. All node groups can have different probabilities for all POI groups. Route based movement model (MapRouteMovement) can be used to model nodes that follow certain routes E.g., fixed path between two areas within a mine. Only the stops on the route have to be defined and then the nodes using that route move from stop to stop using shortest paths and stop on the stops for the configured time. All movement models can also decide when the node is active (moves and can be connected to) and when not. For all models, except for the external movement, multiple simulation time intervals can be given and the nodes in that group will be active only during those times.

Routing modules define how the messages are handled in the simulation. Six basic active routing modules (First Contact, Epidemic, Spray and Wait, Direct delivery, PRoPHET and MaxProp) and also a passive router for external routing simulation are included in the package. The active routing modules are implementations of the well known routing algorithms for DTN routing. There are also variants of these models and couple of different models included in the latest versions.

Passive router is made especially for interacting with other (DTN) routing simulators or running simulations that don't need any routing functionality. The router doesn't do anything unless commanded by external events. These external events are provided to the simulator by a class that implements the EventQueue interface.

Once the required movement models in ONE simulator are configured accordingly to the GNN model, we can compile ONE simulator using its Graphical users interface as shown in the figure 9. Once the compilation is started the GUI displays the movement of the nodes and intersection of the nodes at a junctions. When they intersect at a particular junction the data is exchanged among specific nodes or DTN agents. Every node reaching the junction apart from the source node their future path is predicted by the model and then the data is exchanged between specific nodes based on the priority in which they reach the destination soon. The computation is a continuous process and we can determine execution time period. At each and every stage the machine learning GNN model learns from the previous data fed as input. The output can be seen in the report generated or in the GUI of one simulator itself. The output consists of path travelled by various nodes, intersection of nodes, messages routed between the nodes etc as shown in figure 10. The below is algorithm which depicts the computation of entire GNN model.

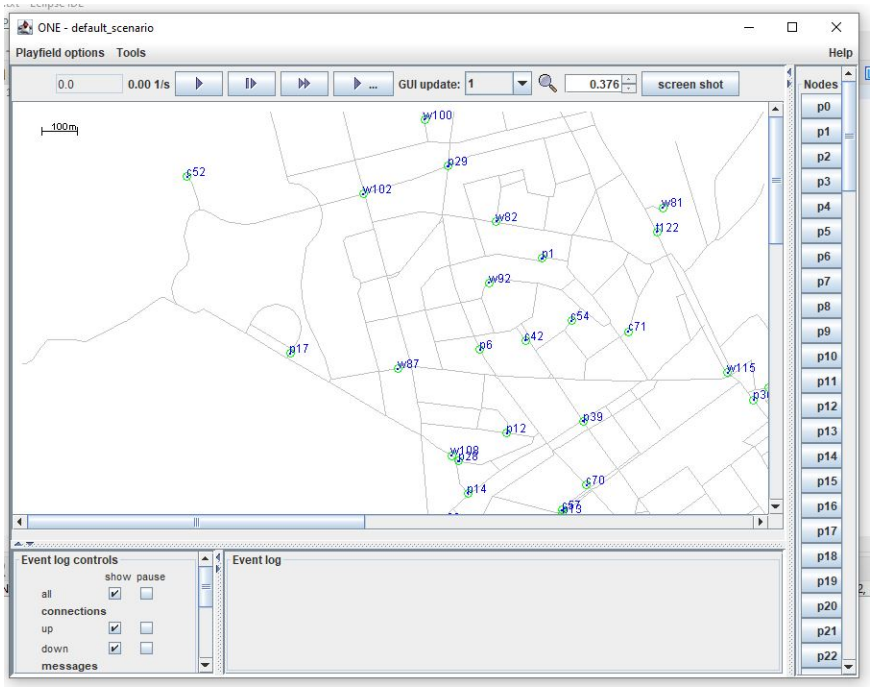

Figure 10: ONE simulator GUI

Algorithm 2 Algorithm for model

1: D ←get data from Brinkoff generator

D ←Make data consistent with mine, make segment and miners stop at regular intervals

D ←Make data DTN consistent, insert break in data, the data cannot be a lot as the memory of the DTN node is small

D ←Clean data(remove NAs etc)

D ←Convert data into combination of 2,3 segments for prediction

Tests :

Test 1 ←Markov Chain with memory

Test 2 ←LSTM cell

Test 3 ←GRU cell

Test 4 ←Graph Model

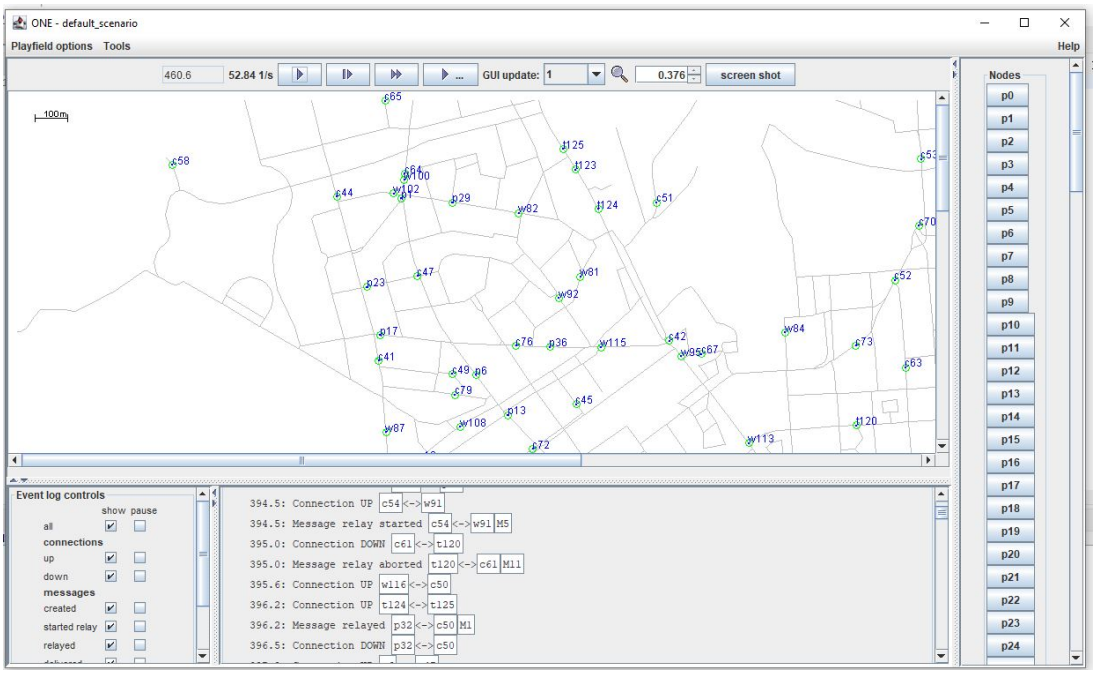

Figure 11: Output after compilation in ONE Simulator

6 Experiments

The experiments we have done have been to compare how previous models have done on the location prediction tasks vs the one we do currently. We are comparing our work with Logistic Regression, LSTM, GRU and other such architecture which would allow us to learn more about how can the different changes be made to learn more about the model and also predict better and with higher accuracy. Some of things we are experimenting are as follows:

- Exp 1 : How much information is good enough for the prediction of location of miner?
- Exp 2 : How many miners should meet each other to learn or get a better understanding of the locations and make the prediction accuracy better?
- Exp 3 : How the pauses differ the kind of predictions made i.e. how accurate is prediction when there are more pauses vs less pauses
- Exp 4 : Does time2vec representation make it better for the learning of timings and positions vs just using spatial information and treating the time as another feature.
- Exp 5 : Can transformers make it better for prediction when there are multiple miners that connect?

References

[1] Xie, Y., Xiong, Y., Zhu, Y. (2020). SAST-GNN: A Self-Attention Based Spatio-Temporal Graph Neural Network for Traffic Prediction. In: Nah, Y., Cui, B., Lee, SW., Yu, J.X., Moon, YS., Whang, S.E. (eds) Database Systems for Advanced Applications. DASFAA 2020. Lecture Notes in Computer Science, vol 12112. Springer, Cham. https://doi.org/10.1007/978-3-030-59410-7_49

[2] Cao, Defu, et al. "Spectral temporal graph neural network for trajectory prediction." 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[3] Du Y, Wang C, Qiao Y, Zhao D, Guo W (2018) A geographical location prediction method based on continuous time series Markov model. PLOS ONE 13(11): e0207063. https://doi.org/10.1371/journal.pone.0207063

[4] Berg, Rianne van den, Thomas N. Kipf, and Max Welling. "Graph convolutional matrix completion." arXiv preprint arXiv:1706.02263 (2017).

[5] M. Littman and J. Boyan, "A Distributed Reinforcement Learning Scheme for Network Routing," Proceedings of the First International Workshop on Applications of Neural Networks to Telecommunications, pp. 45-51, 1993.

[6] T. Mitchell, Machine Learning, C. I. Liu, Ed.: McGraw-Hill, pp. 367-381(Reinforcement Learning) and 157-184 (Bayesian Learning), 1997.

[7] A. Barto and R. Sutton, Reinforcement Learning: An Introduction, Thomas Dietterich, Ed. Cambridge, MA, USA: The MIT Press, 1998.

[8] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in Proc. of ACM SIGCOMM workshop on Delay-tolerant networking, ser. WDTN'05, 2005, pp. 252–259.

[9] A. Vahdat and D. Becker, "Epidemic routing for partiallyconnected ad hoc networks," Tech. Rep., 2000.

[10] Z. J. Haas and T. Small, "A new networking model for biological applications of ad hoc sensor networks," IEEE/ACM Transactions on Networking, vol. 14, pp. 27–40, 2006.

[11] A. C. K. Vendramin, A. Munaretto, M. R. Delgado, and A. Viana, "Grant: Inferring best forwarders from complex networks' dynamics through a greedy ant colony optimization," Computer Networks, vol. 56, pp. 997–1015, 2012.

[12] A. Korn, A. Schubert, and A. Telcs, "Lobby index in networks," Physica A: Statistical Mechanics and its Applications, vol. 388, pp. 2221–2226, Jun. 2009.

[13] G. Pallis, D. Katsaros, M. D. Dikaiakos, N. Loulloudes, and L. Tassiulas, "On the structure and evolution of vehicular networks," in Proc. IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS'09), 2009, pp. 1–10.

[14] A. Lindgren, A. Doria, and O. Schel´en, "Probabilistic routing in intermittently connected networks," SIGMOBILE Mob. Comput. Commun. Rev., vol. 7, pp. 19–20, 2003.

[15] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in Proc. 25th IEEE International Conference on Computer Communications (INFOCOM'06), Apr. 2006, pp. 1–11.

[16] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[17] M. Colagrosso, "A classification approach to broadcasting in mobile ad hoc network," in Proc. IEEE International Conference on Communications (ICC'05), vol. 2, 2005, pp. 1112–1117.

[18] L. Vu, Q. Do, and K. Nahrstedt, "Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace," in IEEE International Conference on Pervasive Computing and Communications (PerCom'11), 2011, pp. 54–62.

[19] S. Ahmed and S. S. Kanhere, "A bayesian routing framework for delay tolerant networks," Apr. 2010, pp. 1–6.

[20] J. R. Quinlan, C4.5: programs for machine learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[21] Z. Jin, D. Guan, J. Cho and B. Lee. "A Routing Algorithm Based on Semi-supervised Learning for Cognitive Radio Sensor Networks," SENSORNETS, pp. 188-194, 2014.