DualGRETEL+: Exploiting Dual Hypergraphs for Path Inference Applied to Navigation Data

Anastasia-Sotiria Toufa
Aristotle University of Thessaloniki
Thessaloniki, Greece
toufaanast@csd.auth.gr

Ioannis Tsingalis
Aristotle University of Thessaloniki
Thessaloniki, Greece
tsingalis@csd.auth.gr

Constantine Kotropoulos
Aristotle University of Thessaloniki
Thessaloniki, Greece
costas@csd.auth.gr

ABSTRACT

This paper addresses the problem of path inference in GPS navigation data by enhancing a generative path inference model called GRETEL. The enhanced model, DualGRETEL+, utilizes a dual hypergraph for feature extraction to capture more complex interactions among GPS data. Additionally, a second-order optimizer, the AdaHessian, is employed to enhance the performance of DualGRETEL+. To evaluate the proposed framework, three distinct datasets were used. Experiments indicate that the use of hypergraph features and AdaHessian optimizer contribute to a significant improvement in performance. Consequently, DualGRETEL+ is a promising solution for the path inference problem in GPS navigation data.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

link prediction, path extrapolation, GPS navigation data, graph neural network, GRETEL, dual hypergraph, feature extraction

ACM Reference Format:
Anastasia-Sotiria Toufa, Ioannis Tsingalis, and Constantine Kotropoulos. 2023. DualGRETEL+: Exploiting Dual Hypergraphs for Path Inference Applied to Navigation Data. In *27th Pan-Hellenic Conference on Progress in Computing and Informatics (PCI 2023), November 24–26, 2023, Lamia, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3635059.3635061

1 INTRODUCTION

In recent years, the widespread availability of Global Positioning System (GPS)-enabled devices has led to an explosion in the available location data. GPS data are collected from GPS devices, providing highly accurate information about the movement of vehicles, people, and other objects. They can be used to obtain insights into traffic patterns, route planning, and other applications [5].

One approach to analyzing GPS data is to represent them as a graph, where nodes represent geographic locations, and edges represent connections between them [18]. This graph-based representation allows the application of powerful machine learning techniques such as Graph Convolutional Networks (GCNs) [4, 9, 12] to analyze and predict the movements of objects in space. Path inference, which involves predicting an object's future path or trajectory based on its past movements and other relevant features, is a common technique in analyzing GPS data to predict the movements of vehicles, people, or other objects.

In the context of GPS data, the path inference problem has been addressed by a generative model called GRETEL [3]. The model is designed to accurately capture the directionality of an observed path of ordered GPS locations, referred to as a *prefix*, and generate a suggested path, known as a *suffix*. Candidate suffixes are generated by performing a non-backtracking walk on the modified graph. The ultimate aim is to predict the upcoming roads the driver will likely take based on their travel history.

The Dual Hypergraph Transformation (DHT) algorithm transforms a conventional graph into its dual hypergraph, focusing on edge representation [6]. Hypergraphs are extensions of traditional graphs. They are capable of modeling higher-order interactions [10]. The edges of the original graph are transformed into the hypergraph nodes, and the original graph nodes are transformed into the hyperedges of the hypergraph. The resulting hypergraph can be represented using an incidence matrix that captures all the information. The transformation to dual hypergraph representation allows for more flexible and expressive modeling of complex relationships among GPS data, emphasizing edge characteristics. Here, it is shown that incorporating these new features into the GRETEL model significantly enhances its performance, enabling more accurate predictions of object movements in space. The improved model is called DualGRETEL+ and is tested in three different datasets. The outline of the proposed framework's major steps at a high level is summarized next, and they are visually presented in Fig. 1.

(1) Construct the graph using navigation data.
(2) Apply the DHT to the original graph to obtain its corresponding dual hypergraph
(3) Extract novel features from the dual hypergraph
(4) Utilize both the navigation data and the extracted features within the GRETEL model for path inference

The optimization process greatly influences the performance of machine learning models [2]. Thus, the choice of the optimizer can significantly affect the results. In the context of training DualGRETEL+, two popular optimization algorithms, Adam [8], and AdaHessian [16], are evaluated. Adam uses a combination of the gradient's first- and second-order moments to adapt the learning rate of each weight of the neural network. It is a first-order optimizer that performs well on a diverse set of deep learning tasks. While Adam is a popular optimization algorithm due to its ability to converge to a good solution quickly, some research has shown that in certain cases, it may fail to converge to the optimal solution and

This work is licensed under a Creative Commons Attribution International 4.0 License.

PCI 2023, November 24–26, 2023, Lamia, Greece
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1626-3/23/11.
https://doi.org/10.1145/3635059.3635061

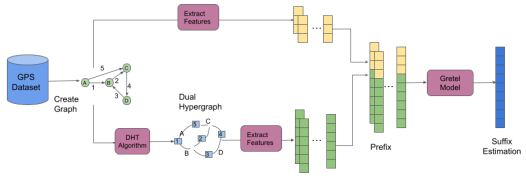


Figure 1: Algorithmic steps of the proposed framework.

even lead to poorer generalization performance [13]. AdaHessian is a second-order optimizer that uses the Hessian matrix, which captures the curvature of the loss function. AdaHessian has been shown to outperform Adam on some tasks, particularly in terms of generalizing performance [16]. However, AdaHessian is computationally more expensive than Adam, as it requires the computation of the Hessian matrix. The computational cost may be prohibitive for large neural networks. By conducting experiments on three different datasets with GPS navigation paths, it is observed that the original configuration of GRETEL, which uses the Adam optimizer, can be further improved. Specifically, utilizing the AdaHessian optimizer, a better performance associated with higher accuracy is obtained in the path inference problem.

The paper's major contribution is the integration of DHT and the AdaHessian optimizer into GRETEL, resulting in DualGRETEL+ applied to navigation data. This integration brings significant enhancements to the model's performance. By leveraging DHT, new features can be extracted that capture complex correlations among GPS data, leading to better results. Tests are conducted on three datasets, expanding the scope of [3] where only one dataset was used. Meanwhile, using the AdaHessian optimizer further improves DualGRETEL+ overall performance, extending the previous work for path inference in Wikipedia links [14]. Together, these two advancements make DualGRETEL+ a highly effective tool for path inference.

The paper is structured as follows: Section 2 provides an in-depth discussion of the methods employed in the GRETEL model, along with the DHT algorithm and the methods used for feature extraction that leads to DualGRETEL+. Section 3 describes the experimental setup and compares the Adam and AdaHessian optimizers [16] using three different datasets. This section also includes a thorough presentation of the experiments related to feature extraction from the dual hypergraph. Finally, conclusions are drawn in Section 4.

2 METHODOLOGY

2.1 Path inference with GRETEL

Assume $G = (\mathcal{V}, \mathcal{E})$ is a graph with n nodes and m edges. We are interested in finding the shortest path between two nodes. An agent moves from one node to another only if there exists a directed edge connecting them. At any given time t, the agent's location is given by the sequence of nodes $p = (v_1, v_2, \ldots, v_t)$, known as the prefix of the path traversed on G. Let h be the prediction horizon. The aim is to estimate the conditional likelihood $\Pr(s \mid h, p, G)$, where $s = (v_{t+1}, \ldots, v_{t+h})$ represents the suffix of the path traversed on G. This estimation uses the CRETEL algorithm proposed in [3].

At time t, the agent is represented as a sparse vector $\boldsymbol{x}_t \in \mathbb{R}^n \geq 0$ that has been normalized to have a sum of one. The ith non-zero element of $\boldsymbol{x}_t$ represents the probability that the agent is located at node v_i at t. A *trajectory* of the agent is defined as $\phi \stackrel{\Delta}{=} (\boldsymbol{x}_\tau : \tau \in \mathcal{I})$, where $\mathcal{I}$ is a sub-sequence of $1, 2, \ldots$. Therefore, estimating the likelihood $\Pr(s \mid h, p, G)$ is equivalent to estimating $\Pr(s \mid h, \phi, G)$.

CRETEL [3] is a generative model for graphs. In other words, the model can generate a suffix path given a prefix path and a horizon. To account for the directionality of edges in the graph, a *latent graph* is defined as $\Phi \stackrel{\Delta}{=} (\mathcal{V}, \mathcal{E}, w_\phi)$, where $w_\phi = f_\theta(G, \phi)$ is a multi-layer perceptron (MLP) network that encodes the edge directionality in the graph G. Specifically, the MLP computes the non-normalized weights of each edge as follows

$$z_{i\to j} = \text{MLP}(\boldsymbol{c}_i, \boldsymbol{c}_j, \boldsymbol{f}_i, \boldsymbol{f}_j, \boldsymbol{e}_{i\to j}). \tag{1}$$

Here, $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ are the pseudo-coordinates of the sender and receiver node, respectively, while $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ represent the features of the sender and receiver nodes.

In (1), the feature vector $\boldsymbol{e}_{i\to j}$ corresponds to the edge connecting the sender and receiver nodes. The computed MLP outputs are normalized using the softmax function, expressed as follows

$$w_\phi(e_{i\to j}) = \frac{z_{i\to j}}{\sum_{v_l \in \mathcal{V}} z_{i\to l}}. \tag{2}$$

Let $\boldsymbol{c}_{i,j}$ denote the jth element of the row vector $\boldsymbol{c}_i \in \mathbb{R}^{1\times|\mathcal{I}|}$. The pseudo-coordinates $\boldsymbol{c}_i = [\boldsymbol{c}_{i,1}, \ldots, \boldsymbol{c}_{i,|\mathcal{I}|}] \in \mathbb{R}^{1\times|\mathcal{I}|}$ are computed using a GNN of K layers as explained next. For a trajectory of length $|\mathcal{I}|$ given by

$$\boldsymbol{X} = [\boldsymbol{x}_{\mathcal{I}_1}, \ldots, \boldsymbol{x}_{\mathcal{I}_{|\mathcal{I}|}}] \in \mathbb{R}^{n\times|\mathcal{I}|}, \tag{3}$$

the graph signals of the GNN of size $n \times |\mathcal{I}|$ are formally given by

$$\boldsymbol{H}^{(k)} = \sigma\left(\boldsymbol{A}\, \boldsymbol{H}^{(k-1)}\, \boldsymbol{W}^{(k-1)}\right), \quad k = 1, 2, \ldots K, \tag{4}$$

where $\sigma(\cdot)$ denotes the logistic function, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the graph, and $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ are the GNN weights. The recursion in (4) is initialized with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. This way, we have

$$\boldsymbol{c}_i = [\boldsymbol{H}^{(K)}]_{i:} = [\boldsymbol{c}_{i,1}, \ldots, \boldsymbol{c}_{i,|\mathcal{I}|}]. \tag{5}$$

Given a target distribution $\boldsymbol{x}_{t+h}$, the model tries to estimate the destination distribution $\hat{\boldsymbol{x}}_{t+h}$ over a horizon h. This is done by the non-backtracking walk [7]

$$\hat{\boldsymbol{x}}_{t+h} = \boldsymbol{B}_\phi^+ \boldsymbol{P}_\phi^h \boldsymbol{B}_\phi \boldsymbol{x}_t, \tag{6}$$

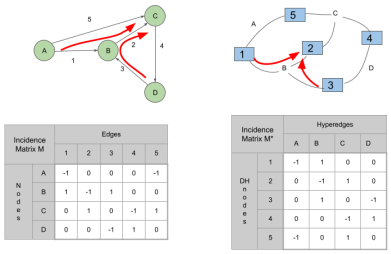
where $\boldsymbol{P}_\phi \in \mathbb{R}^{m\times m}$ has elements

$$[\boldsymbol{P}_\phi]_{e_{i\to j}, e_{k\to l}} = \begin{cases} 0 & \text{if } j \neq k \text{ or } i = l \\ \frac{w_\phi(e_{k\to l})}{1 - w_\phi(e_{k\to i})} & \text{otherwise} \end{cases} \tag{7}$$

and $\boldsymbol{B}_\phi$ a is a $m \times n$ matrix with $[\boldsymbol{B}_\phi]_{e_{i\to j},k} = 0$ if $k \neq i$ and $w_\phi(e_{k\to j})$, otherwise. In this case, the dot loss

$$g(\hat{\boldsymbol{x}}_{t+h}, \boldsymbol{x}_t) = -\hat{\boldsymbol{x}}_{t+h}^T \boldsymbol{x}_{t+h} \tag{8}$$

can be applied to train the model.



Incidence Matrix M		Edges				
		1	2	3	4	5
Nodes	A	-1	0	0	0	-1
	B	1	-1	1	0	0
	C	0	1	0	-1	1
	D	0	0	-1	1	0

Incidence Matrix M*		Hyperedges			
		A	B	C	D
DH nodes	1	-1	1	0	0
	2	0	-1	1	0
	3	0	1	0	-1
	4	0	0	-1	1
	5	-1	0	1	0

Figure 2: Transformation of a simple graph to its corresponding dual hypergraph. The incidence matrix of the original graph and the dual hypergraph is presented. A visual representation of how the directed edges are computed in the dual hypergraph is given.

2.2 Dual Hypergraph Transformation (DHT)

A graph G can be fully described by its node and edge features as well as the connections among them. Node features are represented with matrix $\boldsymbol{F} \in \mathbb{R}^{n \times d}$ where n is the number of nodes, and d is the dimension of a node feature vector. Respectively, edge features are represented with matrix $\boldsymbol{E} \in \mathbb{R}^{m \times d'}$, where m is the number of edges, and d' is the dimension of an edge feature vector. The adjacency matrix $\boldsymbol{A}$ captures node connections. The incidence matrix of an undirected graph given by $\boldsymbol{M} \in \{0,1\}^{n \times m}$ or the incidence matrix of a directed graph given by $\boldsymbol{M} \in \{-1,0,1\}^{n \times m}$, provides additional information that captures the node-edge relationships as well as the orientation of the edges. Thus, a graph can be represented as $G = (\boldsymbol{F}, \boldsymbol{M}, \boldsymbol{E})$.

A hypergraph is a mathematical structure that generalizes the concept of a graph. In a hypergraph, edges can connect any number of vertices, not just two, as in a traditional graph. In this way, higher-order interactions can be represented. A hypergraph is typically represented by a set of vertices and a collection of hyperedges that connect subsets of these vertices. This information can be extracted by the incidence matrix $\boldsymbol{M}$. A hypergraph can be defined as $G^* = (\boldsymbol{F}^*, \boldsymbol{M}^*, \boldsymbol{E}^*)$, where $\boldsymbol{F}^*$ and $\boldsymbol{E}^*$ are the node and hyperedge features respectively, and $\boldsymbol{M}^*$ is the incidence matrix of the hypergraph.

The Dual Hypergraph Transform (DHT) interchanges the roles of nodes and edges of the original graph [6]. The features accompanying the nodes and edges are preserved, but they also change roles. More specifically, an edge in the original graph is transformed into a node in the dual hypergraph. A node in the original graph is transformed into a hyperedge in the dual hypergraph, i.e., $\boldsymbol{F}^* = \boldsymbol{E} \in \mathbb{R}^{m \times d'}$ and $\boldsymbol{E}^* = \boldsymbol{F} \in \mathbb{R}^{n \times d}$. The incidence matrix of the dual hypergraph is the transposed incidence matrix of the original graph, i.e., $\boldsymbol{M}^* = \boldsymbol{M}^\top$. The formal representation of this transformation is given by

$$G = (\boldsymbol{F}, \boldsymbol{M}, \boldsymbol{E}) \rightarrow G^* = (\boldsymbol{E}, \boldsymbol{M}^\top, \boldsymbol{F}). \tag{9}$$

The DHT is a bijective algorithm, implying that by applying it to the dual hypergraph G^*, the original graph G can be reconstructed. Fig. 2 shows an example of the DHT applied to a simple graph.

2.3 Feature Extraction in Dual Hypergraph

The DHT algorithm [6] transforms a given graph into its dual hypergraph and extracts features using the incidence matrix in two ways.

If the original graph G is undirected, the incidence matrix $\boldsymbol{M} \in \{0,1\}^{n \times m}$ is a binary matrix of size $n \times m$, where n and m are the numbers of nodes and edges, respectively. Each node l is associated with an incidence row vector $\boldsymbol{q}_l \in \{0,1\}^m$ with elements indexed by $\kappa = 1, 2, \ldots, m$, where κ corresponds to the edge's id. Applying the DHT algorithm, the row vector $\boldsymbol{q}_l \in \{0,1\}^m$ of matrix $\boldsymbol{M}$ is transformed into a column vector $\boldsymbol{q}_l^* \in \{0,1\}^m \equiv \boldsymbol{q}_l^\top$ of matrix $\boldsymbol{M}^*$. As a result, the role of κ changes, and it indexes the ids of the nodes in the dual hypergraph. Each 1 in column vector $\boldsymbol{q}_l^*$ corresponds to a value of κ indicating which nodes of the dual hypergraph are connected with the hyperedge l^*.

For example, if $\boldsymbol{q}_l^*$ has 1 in positions $\kappa = 1, 2, 5$, it means that hyperedge l^* is associated with the nodes of the dual hypergraph v_1^*, v_2^*, and v_5^*. The corresponding description for the original graph indicates that node l participates in edges e_1, e_2, and e_5. This is essentially a one-hot encoding scheme for multi-categorical data, where the categories correspond to the edges in the original graph. The extracted feature is the cosine similarity between incidence row vectors $\boldsymbol{q}_{v_i}$ and $\boldsymbol{q}_{v_j}$, where v_i is the source node, and v_j is the target node of an arbitrary edge e. The key name `similarity-hyperedge` refers to the experiments that use this feature.

For a directed original graph $\boldsymbol{M} \in \{-1,0,1\}^{n \times m}$ has size $n \times m$, where n and m are the numbers of nodes and edges, respectively. Each node l is associated with an incidence row vector $\boldsymbol{q}_l \in \{-1,0,1\}^m$. Examining the corresponding row vector $\boldsymbol{q}_l$ of the node l, a value of -1 in position κ indicates that node l is a source node in edge κ. In contrast, a value of 1 in position κ indicates that node l is a target node in edge κ since κ represents the id of edges. If $\boldsymbol{q}_l$ consists only of $\{-1,0\}$ values, then there are no outgoing edges from node l, and if it consists only of $\{1,0\}$ values, there are no incoming edges to node l. To extract new features related to the input and output edge degrees of the dual hypergraph nodes, the edges' direction must be determined. In this case, the key name for experiments is `DHnode-in-out-degree`. This is accomplished by examining the column vectors $\boldsymbol{q}_l^*$ of matrix $\boldsymbol{M}^*$ and considering the combinations between the associated nodes of the dual hypergraph for each hyperedge l^*. Each node v_i^* with $i = 1, 2, \ldots, m$ in the dual hypergraph corresponds to an edge e_i with $i = 1, 2, \ldots, m$ in the original graph. For every combination (v_i^*, v_j^*), the existence of the path $e_i \rightarrow e_j$ in the original graph that passes through the examined node l is verified.

For example, consider hyperedge $\boldsymbol{B}$ in Fig. 2, which connects nodes v_1, v_2, and v_3. The corresponding values in the incidence matrix $\boldsymbol{M}^*$ is the column vector $\boldsymbol{q}_2$ with values $[1, -1, 1, 0, 0]^\top$. We check each combination of participating nodes, which are $(1-2)$, $(1-3)$, $(2-1)$, $(2-3)$, and $(3-1)$, $(3-2)$. The original graph has a path for edges $e_1 \rightarrow e_2$ through node v_B and a path for edges $e_3 \rightarrow e_2$ through node v_B.

2.4 Optimizer Selection

Neural network optimization typically involves updating the model's weights using gradient-based algorithms such as Gradient Descent

Table 1: Performance of GRETEL model when the Lausanne dataset is used.

Dataset	Optimizer	Feature	Target-probability	Choice-accuracy	Choice-accuracy_deg3	Precision-top1	Precision-top5	Path-nll	Path-nll-deg3
Lausanne	Adam	Primal	**15.5 ± 0.1**	90.22 ± 0.1	52.28 ± 1.1	20.32 ± 0.5	49.5 ± 0.8	3.80 ± 1.27	3.32 ± 1.16
		Similarity-hyperedge	**15.5 ± 0.08**	90.5 ± 0.08	53.7 ± 0.2	**20.72 ± 0.2**	49.38 ± 0.6	3.48 ± 0.83	3.03 ± 0.78
		DHnode-in-out-degree	15.39 ± 0.01	**90.6 ± 0.03**	**54.18 ± 0.1**	20.32 ± 0.3	**50.28 ± 1.32**	**2.23 ± 0.06**	**1.88 ± 0.05**
		Similarity-hyperedge-DHnode-in-out-degree	15.44 ± 0.1	90.46 ± 0.04	53.56 ± 0.1	**20.72 ± 0.2**	49.9 ± 1.3	3.04 ± 0.80	2.65 ± 0.74
	AdaHessian	Primal	23.8 ± 0.1	**94.6 ± 0.2**	72.04 ± 1.8	**32.58 ± 0.4**	65.72 ± 0.1	2.06 ± 0.07	1.62 ± 0.06
		Similarity-hyperedge	23.2 ± 0.1	94.0 ± 0.1	72.56 ± 0.4	30.4 ± 0.5	66.32 ± 0.7	**1.90 ± 0.01**	**1.51 ± 0.01**
		DHnode-in-out-degree	**24.0 ± 0.2**	94.0 ± 0.1	**72.9 ± 1.0**	30.14 ± 1.0	**67.46 ± 0.3**	2.36 ± 0.11	1.86 ± 0.1
		Similarity-hyperedge-DHnode-in-out-degree	23.8 ± 0.2	93.78 ± 0.08	71.18 ± 0.6	32.2 ± 0.7	65.46 ± 0.5	2.41 ± 0.08	1.92 ± 0.07

Table 2: Performance of GRETEL model when the Geolife dataset is used

Dataset	Optimizer	Feature	Target-probability	Choice-accuracy	Choice-accuracy_deg3	Precision-top1	Precision-top5	Path-nll	Path-nll-deg3
Geolife	Adam	Primal	**6.04 ± 0.3**	**82.84 ± 0.7**	**64.98 ± 1.7**	**6.1 ± 0.4**	29.76 ± 0.2	3.58 ± 0.69	1.33 ± 0.26
		Similarity-hyperedge	5.94 ± 0.3	81.88 ± 0.1	63.64 ± 0.7	6.04 ± 0.4	**30.06 ± 0.5**	3.85 ± 0.930	1.406 ± 0.32
		DHnode-in-out-degree	5.52 ± 0.04	81.34 ± 0.2	63.22 ± 0.3	5.56 ± 0.1	28.94 ± 0.2	3.39 ± 0.31	1.17 ± 0.07
		Similarity-hyperedge-DHnode-in-out-degree	5.6 ± 0	82.0 ± 0.1	62.8 ± 0.1	5.56 ± 0.05	29.42 ± 0.1	**2.81 ± 0**	**1.041 ± 0**
	AdaHessian	Primal	11.7 ± 0.1	84.32 ± 0.1	74.88 ± 0.2	14.72 ± 0.2	33.98 ± 0.2	7.69 ± 0.373	1.97 ± 0.04
		Similarity-hyperedge	11.92 ± 0.2	85.36 ± 0.2	76.88 ± 0.5	15.44 ± 0.4	**35.0 ± 0.2**	**7.52 ± 0.35**	**1.93 ± 0.06**
		DHnode-in-out-degree	11.6 ± 0.02	84.0 ± 0.08	73.78 ± 0.2	14.3 ± 0.1	33.1 ± 0.3	7.72 ± 0.12	2.14 ± 0.03
		Similarity-hyperedge-DHnode-in-out-degree	**12.22 ± 0.03**	**85.52 ± 0.01**	**77.02 ± 0.04**	**15.68 ± 0.06**	34.62 ± 0.08	8.83 ± 0.10	2.47 ± 0.02

or Stochastic Gradient Descent [13]. However, these methods can slowly converge and tend to overfit the training data, resulting in poor generalization performance on unseen data [8]. To address these issues, researchers have proposed second-order optimization methods that utilize information from the Hessian matrix of the loss function. In addition to the Hessian information, gradient information can improve convergence and generalization performance. AdaHessian [16] is one such method that modifies the update rule of the popular Adam optimizer [8] to include second-order information, resulting in improved convergence and generalization performance on various tasks. Specifically, AdaHessian uses a Hessian diagonal matrix approximator [1] to estimate the second-order information and updates the model parameters as follows

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \, \boldsymbol{p}_t \oslash \boldsymbol{m}_t, \tag{10}$$

where $\oslash$ defines the element-wise division operator between two vectors. Here, $\boldsymbol{\theta}_t$ and η_t are the model parameters and the learning rate at time step t, respectively. $\boldsymbol{m}_t$ and $\boldsymbol{p}_t$ are the first and second moments of the AdaHessian, respectively, computed using exponential moving averages, i.e., for $0 \leq k < 1$:

$$\boldsymbol{m}_t = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \boldsymbol{g}_i}{1-\beta_1^t} \tag{11}$$

and

$$\boldsymbol{p}_t = \left(\sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \boldsymbol{D}_i \boldsymbol{D}_i}{1-\beta_2^t}} \right)^k \tag{12}$$

Here, β_1 and β_2 stand for the exponential decay rates for the first and second-moment estimations, respectively. Typical values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Moreover, in (11), $\boldsymbol{g}_i$ is the gradient of the loss function while in (12) $\boldsymbol{D}_i$ is the spatially averaged Hessian diagonal approximation of the loss function at time step i [16]. It should be noted that Adam applies similar formulas for modifying the model parameters. However, it differs in the aspect that instead of using the averaged Hessian diagonal $\boldsymbol{D}_i$ as in (12), it utilizes the gradient $\boldsymbol{g}_i$ in the second-order moment calculation. DualGRETEL+ uses the Adahessian optimizer in contrast to the original GRETEL model, which resorts to the Adam optimizer [3]. Experiments are reported in Section 3, which demonstrate the effectiveness of AdaHessian in path inference.

3 EXPERIMENTS

Experiments have been conducted on three different datasets with two main objectives. Firstly, to demonstrate the effectiveness of incorporating novel hypergraph features into the GRETEL model. Secondly, to evaluate the impact of using the AdaHessian optimizer on model performance and generalization ability. The experimental findings indicate that the AdaHessian optimizer has resulted in better performance and higher accuracy in path inference problems than the Adam optimizer.

All three datasets use navigation paths derived from GPS data, which can be prone to errors due to GPS noise, signal loss, or other factors. Therefore, a pre-processing step is required to align the GPS data to a known road network, enabling the determination of the vehicle's route. This process is known as map matching and aims to improve the accuracy and usefulness of the location data.

The first dataset is the same as in [3], which includes food deliveries occurring over the OpenStreetMap road network of Lausanne. The map graph includes 18, 156 nodes and 32, 468 edges.The second dataset, Geolife [17], contains GPS trajectories recorded by Microsoft Research Asia from April 2007 to August 2012 in Beijing, China. The dataset includes additional data, such as timestamps, altitude, user speed, and GPS coordinates. Geolife consists of 32, 442 nodes and 53, 050 edges. The third dataset, iWet, includes tourist itineraries for buses in the Central Macedonia region of Greece. iWet comprises 18, 317 nodes and 43, 787 edges. Both Geolife and iWet use the Fast Map Matching algorithm [15], a graph-based approach that leverages a probabilistic model and dynamic programming to match GPS points to road segments. In contrast, Lausanne dataset uses a Hidden Markov Model (HMM) as a map matching algorithm [11].

Table 3: Performance of GRETEL model when the `iWet` dataset is used.

Dataset	Optimizer	Feature	Target-probability	Choice-accuracy	Choice-accuracy-deg3	Precision-top1	Precision-top5	Path-nll	Path-nll-deg3
iWet	Adam	Primal	1.32 ± 0	63.52 ± 0	50.93 ± 0.02	3.31 ± 0	5.2 ± 0	5.09 ± 0	2.63 ± 0
		Similarity-hyperedge	1.32 ± 0	63.52 ± 0	50.93 ± 0.02	3.31 ± 0	5.2 ± 0	5.09 ± 0	2.63 ± 0
		DHnode-in-out-degree	1.32 ± 0	63.52 ± 0	50.93 ± 0.02	3.31 ± 0	5.2 ± 0	5.09 ± 0	2.63 ± 0
		Similarity-hyperedge-DHnode-in-out-degree	1.32 ± 0	63.52 ± 0	50.93 ± 0.02	3.31 ± 0	5.2 ± 0	5.09 ± 0	2.63 ± 0
	AdaHessian	Primal	5.07 ± 0.1	**86.16 ± 0.3**	**88.01 ± 0.9**	11.17 ± 0.03	18.38 ± 0.4	**4.40 ± 0.23**	**1.51 ± 0.06**
		Similarity-hyperedge	-	-	-	-	-	-	-
		DHnode-in-out-degree	**6.29 ± 0.06**	80.94 ± 0.3	75.68 ± 0.5	**17.02 ± 0.6**	**22.83 ± 0.6**	14.87± 0.5	7.36 ± 0.39
		Similarity-hyperedge-DHnode-in-out-degree	5.81 ± 0.1	81.19 ± 0.6	76.8 ± 0.8	15.91 ± 0.6	21.45 ± 0.8	13.0 ± 0.98	6.12 ± 0.53

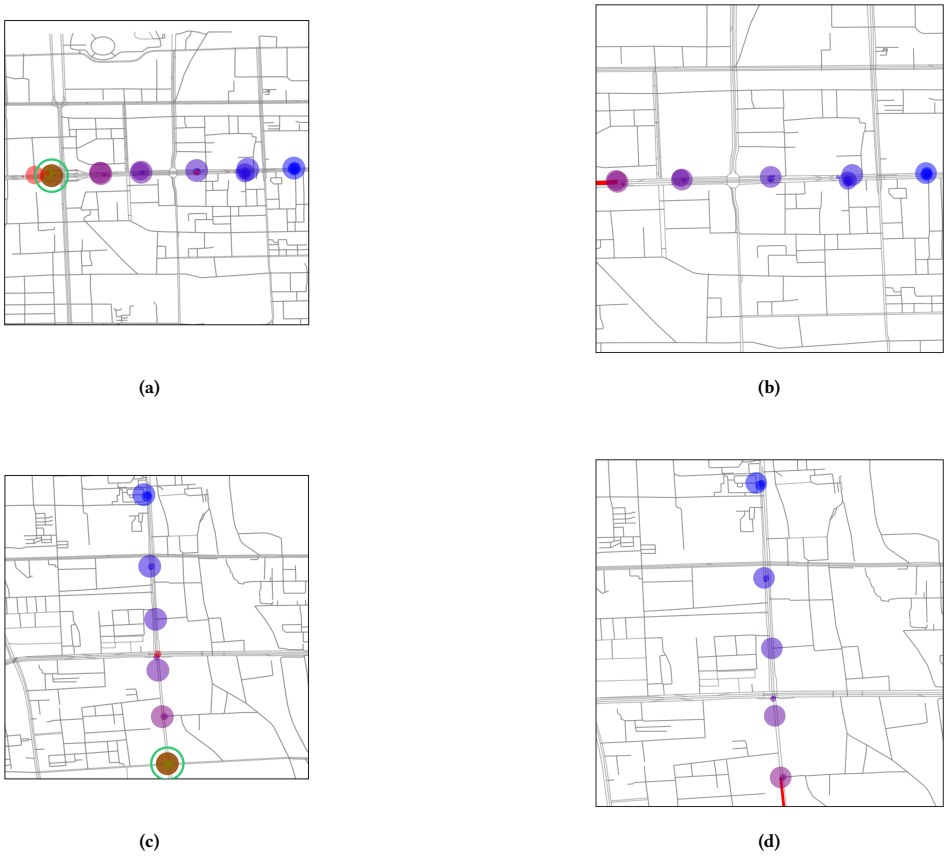
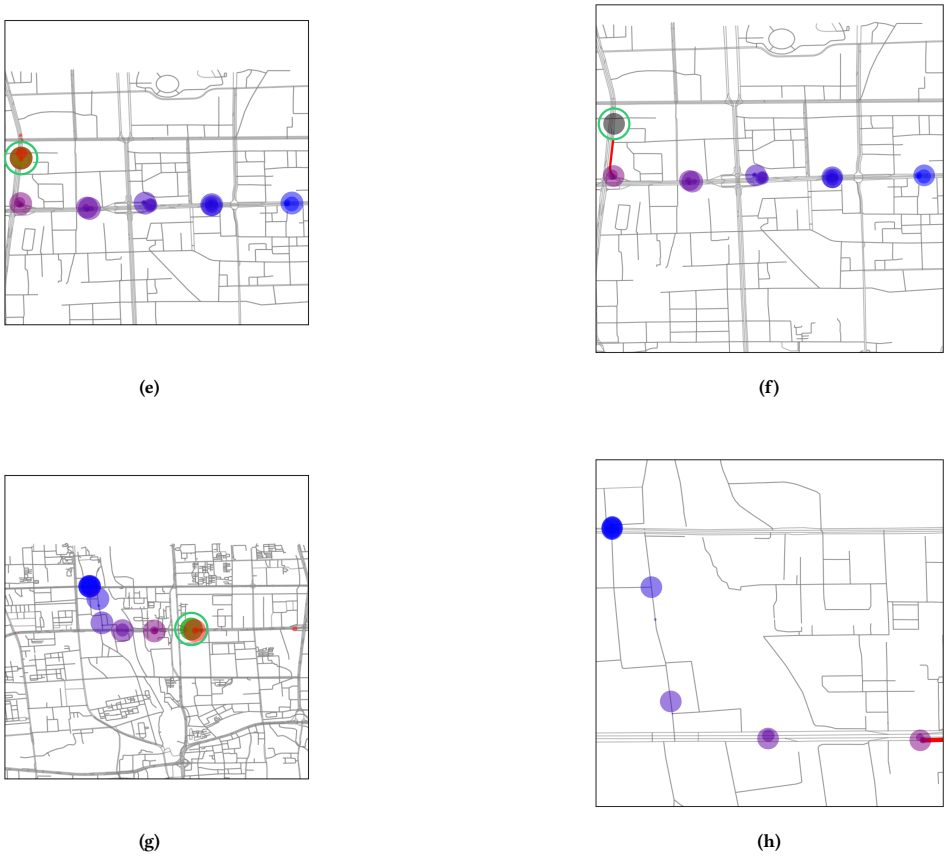


Figure 3: Prediction examples in Geolife dataset, using `similarity-hyperedge` and `DHnode-in-out-degree` features. Each row showcases different examples. The left-column images (a) and (c) display 5 historical trajectory prefixes, visually represented with markers that transition from blue to purple, with the green circle signifying the actual target location. Red markers indicate the predicted trajectory suffix reflecting the target distribution. In the right-column images (b) and (d), the direction of the predicted trajectory is presented along with the 5 trajectory prefixes.

The GRETEL model utilizes specific features for nodes and edges, referred to as `primal`, including the distance between nodes and the average speed limit. These features are combined with all features extracted from the dual hypergraph to serve as edge features. The metrics provided by [3] are used. The average probability of the model selecting a node with non-zero likelihood is measured by the *target-probability*. In contrast, *choice accuracy* evaluates the accuracy of an algorithm's decisions at each intersection of the ground-truth path between nodes v_t and v_{t+h}, where h represents the prediction horizon. This metric is calculated for nodes with at least 3 and 1 degrees. The metrics *precision-top1* and *precision-top5* calculate the accuracy of the model's prediction against the actual target by considering the best predictions, where the number of best predictions ranges from 1 to 5. The *path-nll* measures the negative log-likelihood. All results presented here are based on five independent executions, including the mean and standard deviation.



(e) **(f)**

(g) **(h)**

Figure 3: (cont.) The left-column images (e) and (g) display 5 historical trajectory prefixes, visually represented with markers that transition from blue to purple, with the green circle signifying the actual target location. Red markers indicate the predicted trajectory suffix reflecting the target distribution. In the right-column images (f) and (h), the direction of the predicted trajectory is presented along with the 5 trajectory prefixes.

The experimental analysis is performed for each dataset, where the model's performance is compared when using the hypergraph features and AdaHessian optimizer. The baseline case involves using the `primal` features and Adam optimizer.

Table 1 presents the results of applying GRETEL to the `Lausanne` dataset. Incorporating dual hypergraph features improves performance across all metrics when the Adam optimizer is used, with the `DHnode-in-out-degree` feature achieving the highest performance. Similar results are observed when using the AdaHessian optimizer, except for *choice accuracy* and *precision-top1*. Both `similarity-hyperedge` and `DHnode-in-out-degree` features outperform the `primal` features. Of particular interest is the substantial increase in the overall performance of GRETEL with the AdaHessian optimizer. The most significant increase is observed in *target-probability* and *precision-top1*, with a percentage increase of 54.8% and 57.2%, respectively. The same percentage increase around 34% is observed for *choice-accuracy-3* and *precision-top5*. For *path-nll* and *path-nll-3*, the percentage increase is 14.8% and 19.6% respectively, while the smallest increase is observed for *choice-accuracy*, but the value is already high.

The corresponding results using `Geolife` dataset are presented in Table 2. When using the Adam optimizer, the dual hypergraph features outperform the baseline case only in metrics such as *precision-top5*, *path-nll*, and *path-nll-3*. In other metrics, the `primal` features show slightly better results. However, when using the AdaHessian optimizer, the dual hypergraph features improve the model's performance in all metrics. The `similarity-hyperedge-DHnode-in-out-degree` and `Similarity-hyperedge` features exhibit the best performance. In this case, the impact of the two optimizers on the results is mixed, with both positive and negative effects. Specifically, there is a significant increase in *precision-top1* and *target-probability* of 157.04% and 102.3%, respectively. The difference in *precision-top5* and *choice-accuracy-3* is similar, around 17% while in *path-nll* and *path-nll-3*, there is a decrease of 167.6% and 85.57% respectively.

The experimental results for the iWet dataset can be found in Table 3. When the Adam optimizer was utilized, the model exhibited behavior indicative of being stuck in a local minimum and showed no progress during training. This behavior can sometimes occur with the Adam optimizer when training deep learning models, mainly when dealing with non-convex optimization problems. The optimizer may fail to converge to the global minimum. In contrast, while the model's performance with AdaHessian was notably lower than in the other datasets, the use of dual hypergraph features has still improved the model's performance. The use of `DHnode-in-out-degree` feature increases the model's performance.

Figure 3 presents 4 instances of GPS trajectory prediction in the Geolife dataset. The model used for the prediction uses `similarity-hyperedge` and `DHnode-in-out-degree` features.

4 CONLCUSION

GPS navigation data can be represented as graphs, where each node corresponds to a location, and each edge represents a route or a path between locations. Such representation enables the modeling of complex relationships between GPS points. In this work, we presented DualGRETEL+, an enhanced version of the GRETEL model, for accurate predictions of object movement in space. DualGRETEL+ utilizes a dual hypergraph to extract additional features, allowing for a more flexible and expressive representation of relationships within GPS data. The resulting hypergraph can be represented using an incidence matrix, which captures all the information. By incorporating these new features into GRETEL, we significantly improved its performance on three datasets. We also demonstrated the efficacy of the AdaHessian optimizer for further enhancing the model's performance. This study highlights the potential of hypergraphs and second-order optimization methods for analyzing and predicting object movement in GPS data.

ACKNOWLEDGMENTS

This research was carried out as part of the project "Optimal Path Recommendation with Multi Criteria" (Project code: KMP6-0078997) under the framework of the Action "Investment Plans of Innovation" of the Operational Program "Central Macedonia 2014-2020", that is co-funded by the European Regional Development Fund and Greece.

REFERENCES

[1] C. Bekas, E. Kokiopoulou, and Y. Saad. 2007. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics* 57, 11-12 (2007), 1214–1229.
[2] D. Choi, C. J Shallue, Z. Nado, J. Lee, C. J Maddison, and G. E Dahl. 2019. On empirical comparisons of optimizers for deep learning. arXiv:1910.05446
[3] J.-B. Cordonnier and A. Loukas. 2019. Extrapolating paths with graph neural networks. arXiv:1903.07518
[4] M. Defferrard, X. Bresson, and P. Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates, Inc., Barcelona, Spain, 3844—-3852.
[5] A. El-Rabbany. 2002. *Introduction to GPS: The global positioning system.* Artech House, Norwood, MA.
[6] J. Jaehyeong, B. Jinheon, L. Seul, K. Dongki, K. Minki, and J. H. Sung. 2021. Edge Representation Learning with Hypergraphs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., Virtual Conference, 7534–7546.
[7] M. Kempton. 2016. Non-backtracking random walks and a weighted Ihara's theorem. arXiv:1603.05553
[8] D. P Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980
[9] T. N. Kipf and M. Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
[10] C. Kotropoulos. 2016. Multimedia social search based on hypergraph learning. In *Graph-Based Social Media Analysis*, I. Pitas (Ed.). Vol. 39. CRC Press, Boca Raton, FL, 215–273.
[11] P. Newson and J. Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, Seattle, WA, 336–343.
[12] P. Veličković and G. Cucurull and A. Casanova and A. Romero and P. Lio and Y. Bengio. 2017. Graph attention networks. arXiv:1710.10903
[13] S. Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv:1609.04747
[14] A.-S. Toufa, C. Kotropoulos, and I. Tsingalis. 2023. Dual hypergraph features for path inference in Wikipedia links. In *Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Gold Coast, Queensland, Australia, 1–7.
[15] C. Yang and G. Gidófalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science* 32, 3 (2018), 547–570.
[16] Z. Yao, A. Gholami, S. Shen, k. Keutzer, and M. W. Mahoney. 2021. AdaHessian: An adaptive second order optimizer for machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35 (12). AAAI Press, Virtual Conference, 10665–10673.
[17] Z. Yu, X. Xing, and M. Wei-Ying. 2010. GeoLife: A collaborative social networking service among user, location, and trajectory. *IEEE Data Engineering Bulletin* 33, 2 (2010), 32–39.
[18] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the International Conference on Ubiquitous Computing*. ACM, Seoul, South Korea, 312–321.