Speeding up $KNN - \overline{W}H$ for Origin–Destination Travel Time Estimation

Sofía Alvarez
sofalvarez@alumnos.uai.cl
Universidad Adolfo Ibáñez
Santiago, Región Metropolitana, Chile

Sebastián Moreno
sebastian.moreno@uai.cl
Universidad Adolfo Ibáñez
Santiago, Región Metropolitana, Chile

Alfonso Tobar-Arancibia
alftobar@alumnos.uai.cl
Universidad Adolfo Ibáñez
Santiago, Región Metropolitana, Chile

Wilfredo F. Yushimito
wilfredo.yushimito@uai.cl
Universidad Adolfo Ibáñez
Santiago, Región Metropolitana, Chile

Abstract

Origin-destination (O-D) travel time estimation is among the most important problems studied in transportation. It focuses on determining accurate travel time from a specific origin point to a destination point. Given the development of new technologies such as GPS and mobile applications, this data can be easily gathered, improving the estimation of the O-D travel time and enabling prediction in almost real-time. Currently, one of the simplest and newest algorithms is the $KNN - \overline{W}H$ model, an improvement of the K-Nearest Neighbors method with Haversine distance and a correction factor. Unfortunately, the direct application of this method can take over 50 minutes to predict a new set of 70,000 data points. This paper proposes $k - KNN - \overline{W}H$, a new two-step framework that clusters the data using k-means and then applies $KNN - \overline{W}H$ on the corresponding cluster. The empirical results show a minimal impact on the MAPE performance (1.5%) while reducing the time estimation process from approximately 50 to 20 minutes.

Keywords

origin–destination travel time, machine learning, k-nearest neighbor, speed up estimation process

ACM Reference Format:
Sofía Alvarez, Sebastián Moreno, Alfonso Tobar-Arancibia, and Wilfredo F. Yushimito. 2025. Speeding up $KNN - \overline{W}H$ for Origin–Destination Travel Time Estimation. In *Proceedings of the 1st Workshop on "AI for Supply Chain: Today and Future" @ 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 Introduction

In today's rapidly advancing and interconnected world, cities are becoming increasingly crowded, resulting in traffic congestion and posing significant challenges to urban mobility. Among these problems, one of the most important is predicting travel time between locations, as ensuring the smooth movement of people and goods within the city is crucial for sustainable economic growth and a high quality of life. Accurate prediction of travel times between origin-destination (O-D) pairs enhances traffic management systems and optimizes public transit by synchronizing traffic flows with signal control systems and transit schedules.

Early methods for estimating origin-destination (O-D) travel times primarily relied on home interviews and roadside surveys, which provide a snapshot of the traffic situation and a limited amount of data [7]. These methods were complemented by statistical techniques, such as linear regression models, entropy-maximization approaches, and maximum likelihood estimation methods [17, 24, 28, 34, 39]. However, given the development of new technologies such as GPS and mobile applications, massive data can be gathered on an unmatched scale [25] (spatially and temporally). The GPS applications offer more precise information to describe movements, including the location (latitude and longitude) of the origin and destination of the trip [7, 15]. Then, leveraging this information as an input for machine learning predictive models has improved the prediction of O-D travel times.

Machine learning (ML) is a field of computer science that enables computers to acquire knowledge without explicit programming. In the last years, several models have been developed to predict O-D time travel [12, 21, 23, 31, 42, 47]. While most of these methods consider several variables for their estimation (GPS route, day time, and others), few models focus on the travel time estimation using only the origin and destination data points. Among these few models, one of the simplest and newest algorithms is the $KNN - \overline{W}H$ model [21]. The $KNN - \overline{W}H$ is a model based on the K-Nearest Neighbors method with Haversine distance instead of the Euclidean distance and a correction factor to improve the O-D time travel prediction. Even though the results of $KNN - \overline{W}H$ are statistically better than other baseline models (KNN, extreme gradient boosting, regression tree), the direct application of $KNN - \overline{W}H$ over a new dataset is empirically slow, taking more than 50 minutes to make the predictions over 70,000 data points.

In this paper, we propose a new simple two-step framework by applying k-means and $KNN - \overline{W}H$ to speed up the estimation process of the model, with a small effect on its performance. The framework will group the data into N_k clusters using k-means. Then, when a new data point arrives, we assign it to a cluster and estimate the O-D time travel using $KNN-\overline{W}H$ with the data of the

KDD '25, Toronto, ON, Canada.

ACM ISBN 979-8-4007-1454-2/25/08
https://doi.org/10.1145/XXXXXX.XXXXXX

cluster. By clustering the data into N_k groups with k-means, we reduce the time to predict from 50 to 20 minutes. Unfortunately, this improvement in the prediction process affects the model's performance, as the new data point to predict has access to limited data.

2 Literature Review

The O-D time travel estimation has historically posed a challenge. Initially, the estimation methods were simple and based on home and roadside surveys [5]. Then, during the 1970s decade, alongside the development of computing, several linear models were proposed, such as linear regression [24], Holm's [17], Commonwealth Bureau of Roads [35], and Lamarre's models [22]. Also, nonlinear regression models were introduced, including Robillardls [29], Hogberg's [16], and Wills's [43] models. In the 1980's decade, new methods were explored, such as entropy maximization models [3, 36, 39], and maximum likelihood estimation models, often employing a Poisson distribution [4, 33, 38]. In the following 20 years (1990-2009), the least-squares approach was employed, aiming to minimize the sum of squared differences between each potential element of the observed data points [6, 10, 11, 28, 34].

In the last years (2010-present), advancements in technologies like GPS and mobile applications have enabled the collection of large volumes of data [25], allowing the creation of new models for the estimation of the O-D travel time [26, 27]. This resulted in the use of graphical models [30] and machine learning (ML) models [2, 48] to forecast the time. For example, a 2014 study utilized graphical models to estimate the O-D travel time [9]. However, the computational complexity, due to the numerous variables and relationships involved (each location is treated as a variable), has constrained further research in this area.

Multiple approaches have been considered for estimating the O-D time travel. [46] proposed an ensemble model with Gradient Boosting Regression Tree to improve the prediction accuracy of travel time. This model was later used to empirically reveal the real-world demand for shared mobility-on-demand ride services [8]. In 2016, one of the first frameworks based on the k-nearest-neighbor model (KNN) was published. This study concluded that simple models can be empowered by big data [40]. Later, [20] proposed a deep learning technique that combines convolutional layers and long short-term memory (LSTM) layers, but instead of time prediction, they focus on demand. [19] proposed a spatio-temporal Neural Network composed of two different modules, that jointly predict the travel distance and travel time of the trip. The first module predicts the travel distance, and the output is combined with the time of day to forecast the travel time. Based on the work of [40], [5] estimates the travel time for specific O-D points by using a combination of KNN with shortest-path convex optimization. First, the KNN algorithm for each O-D pair identifies 'k' similar trips, establishing similar routes. Then, the optimization technique focuses on finding the shortest path for each O-D pair over those discovered routes, while minimizing the travel time.

Since 2020, the models to estimate O-D travel time have been mainly based on neural networks and deep learning techniques, changing the focus of the prediction to demand and flow. [44] estimates the demand between O-D pairs by combining a Graph Neural Network and the Kalman filter. [18] applies a graph convolutional neural network to predict demand. The convolutions over the network structure allow the extraction of information from nodes and their neighbors. [37] proposes a three-dimensional convolution-based deep neural network to capture the complex relationships between local traffic patterns and time-of-day patterns, enabling accurate predictions of travel demand. [41] forecasts bus passenger flows (demand) between regions using two steps. First, the O-D data is processed by a Convolutional Neural Network (CNN) to capture spatial relationships and patterns within the data. Then, these results are given to an LSTM neural network to remember information over long periods and generate the predictions. [31] includes urban congestion and taxi flow to predict O-D taxi travel time. After including these data, the paper uses a Gated Recurrent Unit neural network (GRU) for the prediction. [12] predicts travel time using a support vector regressor over a set of features generated by four different neural networks (CNN, LSTM, multiplayer perceptron, and GRU). Lastly, [23] proposed a personalized O-D travel time estimation using Transformers and active adversarial inverse reinforcement learning. Finally, [47] also considers the O-D time travel prediction as a graph problem. In this case, after a feature extraction process of the graph, the paper considered several neural networks (Double Deep Q-Network, SSML, and MetaER-TTE) to make the final prediction.

3 Proposed Method

This section presents a two-step framework to improve the prediction times between an origin and destination (O-D) using k-means and $KNN-\overline{WH}$. The proposed framework combines two models: k-means [13] and a weighted K-Nearest Neighbor with Haversine distance ($KNN-\overline{WH}$, [21]). The first step of the framework groups similar data points, and then these groups are used separately as the input for the predictive model ($KNN-\overline{WH}$). Applying $KNN-\overline{WH}$ to individual clusters reduces the number of computations compared to using the entire training dataset, which decreases prediction time with only a slight impact on model performance.

The proposed $k-KNN-\overline{WH}$ framework is shown in Figure 1, where k-means has been previously trained. As can be observed in the left plot of Figure 1, a data point has four attributes corresponding to a trip's origin and destination coordinates. So, let $\mathbf{x}_i$ be $(Latitude_o, Longitude_o, Latitude_d, Longitude_d)$, where the subindexes o and d mean origin and destination, respectively. Then, in the center plot, $\mathbf{x}_n$ is used as input into the learned k-means model and assigned to G_m one of the N_k possible clusters $(G_1, \ldots, G_{N_k})$. Finally, in the right plot, the time travel for $\mathbf{x}_n$ is predicted using the $KNN-\overline{WH}$ model, based only on the data of the G_m cluster.

The k-means model is a centroid-based iterative clustering algorithm [13]. It aims to separate a set of observations $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}\}$ into N_k different groups/clusters $\mathbf{G} = \{G_1, ..., G_{N_k}\}$, where each cluster G_j has a representative center point called centroid $\mathbf{c}_j$. The model seeks to minimize the within-cluster distance (WCD) given by Eq. (1). This metric calculates the total distance between each observation ($\mathbf{x}_i$) and its centroid ($\mathbf{c}_j$), i.e., the WCD is the sum of the clusters' **wcd** which corresponds to the distance between each point $\mathbf{x}_i$ (that belongs to G_j) and its centroid $\mathbf{c}_j$. Note, *dist* from Eq.

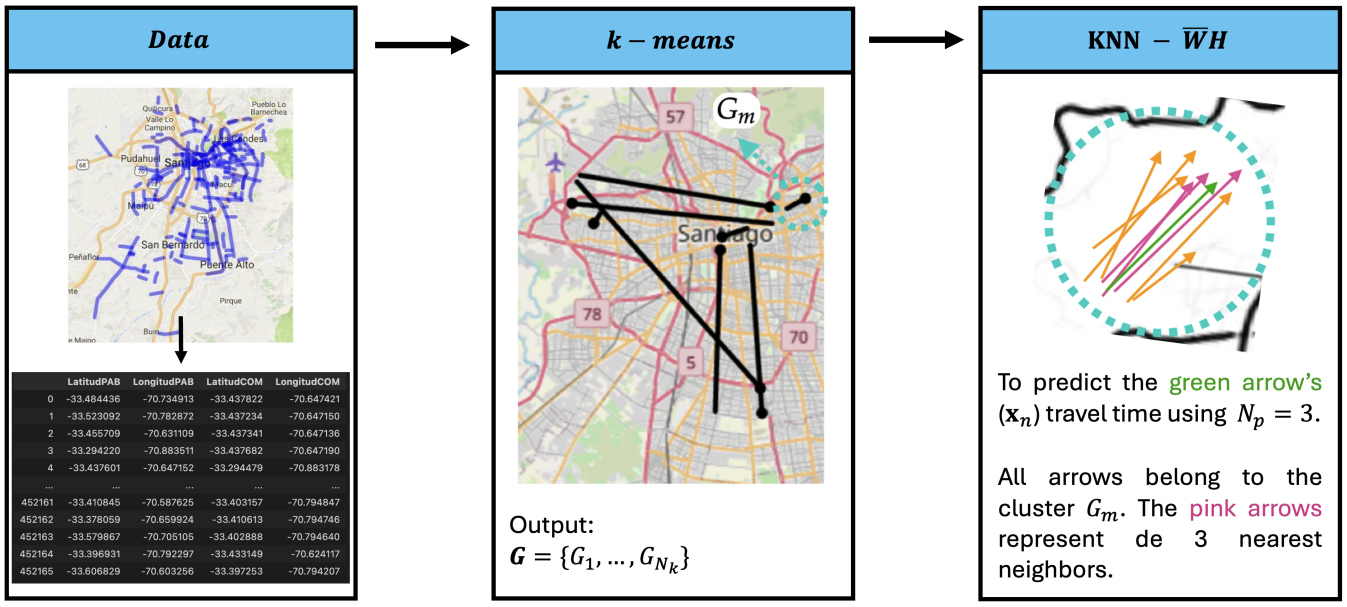

Figure 1: k–KNN–$\overline{WH}$ Framework. Left: training datasets corresponding to origin and destination (O-D) points. Center: k-means is applied to the new datapoint ($\mathbf{x}_n$), which is assigned to the cluster G_m (each black line shows the centroid of a cluster represented by its origin (circle) and destination point (end of line)). Right: the final prediction is estimated using the data points from G_m and the $KNN - \overline{WH}$ model with $N_p = 3$.

(1) is usually the Euclidean distance, but for this paper, we use the Haversine distance to be consistent with the $KNN - \overline{WH}$ model.

$$\text{WCD}(\mathbf{G}) = \sum_{j=1}^{N_k} \mathbf{wcd}(G_n) = \sum_{j=1}^{N_k} \sum_{\mathbf{x}_i \in G_j} \text{dist}(\mathbf{x}_i, \mathbf{c}_j) \tag{1}$$

There are multiple algorithms and strategies to minimize $WCD(\mathbf{G})$ (Eq. (1)). One of the simplest algorithms is to randomly select N_k data points as initial centroids $(\mathbf{c}_1, \ldots, \mathbf{c}_{N_k})$ and assign each point $\mathbf{x}_i$ to one of the centroids. Then, the centroids are recalculated as the average of all the data points belonging to each cluster. Finally, this process is repeated until convergence (the centroids do not change in one iteration). To select the number of clusters (N_k), a typical heuristic is the "elbow" method. This heuristic plots $WCD(\mathbf{G})$ against different numbers of clusters, and the "correct" number of clusters corresponds to the inflection point ("elbow") of the curve [1]. This paper does not aim to determine the optimal number of clusters, as the primary focus is to evaluate the performance of our proposed framework (combining k-means with $KNN - \overline{WH}$) in terms of error and processing time.

The $KNN - \overline{WH}$ model [21] is a supervised learning method based on KNN regressor [32]. For this paper, given a set of observations $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}\}$, KNN predicts the time t_n of a trip $\mathbf{x}_n$, based on the information of its N_p nearest points (also called neighbors). This proximity between points is determined using a distance metric, such as the Euclidean distance. KNN starts by calculating the distance between $\mathbf{x}_n$ and each point $\mathbf{x}_i \in \mathbf{X}$, then it generates $\mathcal{N}(\mathbf{x}_n)$ corresponding to the N_p closest neighbors of $\mathbf{x}_n$. Finally, the estimated $\hat{t}_n$ is the average over the outputs of each $\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_n)$. For example, if $N_p = 3$ and $\mathcal{N}(\mathbf{x}_n) = \{\mathbf{x}_3, \mathbf{x}_6, \mathbf{x}_{19}\}$, then $\hat{t}_n = \frac{t_3 + t_6 + t_{19}}{3}$.

The $KNN - \overline{WH}$ model considers three modifications with respect to KNN regressor. First, it uses the Haversine distance instead of the Euclidean distance. Second, instead of assigning equal importance to each neighbor (averaging their values), the $KNN - \overline{WH}$ assigns a weight w_i to each neighbor of $\mathbf{x}_n$ ($\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_n)$). The weight is defined by Eq. (2), and it is inversely proportional to the distance of the data point to its neighbors. Note, these weights are also a common approach in the original KNN regressor model [32]. Finally, the estimated time of $\hat{t}_n$ is given by Eq. (3), where $KNN - \overline{WH}$ corrects the potential bias generated by the N_p chosen observations by considering the traveled distance of each neighbor trip ($\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_n)$). So, the value for $\hat{t}_n$ is adjusted based on two distances, the distance of the trip being predicted (d_n) and the distance trip of each neighbor (d_i). Continuing with the previous example, where $\mathcal{N}(\mathbf{x}_n) = \{\mathbf{x}_3, \mathbf{x}_6, \mathbf{x}_{19}\}$, while the basic KNN models estimates $\hat{t}_n = \frac{t_3 + t_6 + t_{19}}{3}$, $KNN - \overline{WH}$ predicts $\hat{t}_n = d_n[\frac{w_3 t_3}{d_3} + \frac{w_6 t_6}{d_6} + \frac{w_{19} t_{19}}{d_{19}}]$.

A pseudocode of the proposed framework k–KNN–$\overline{WH}$ can be observed in Algorithm 1. The algorithm receives as input the training data ($\mathbf{X}$) with their outputs ($\mathbf{t}$), the clusterization of the original data generated by k-means ($\mathbf{G}$), the number of data points using for the estimation (N_p), and the data point to estimate $\mathbf{x}_n$. Lines 3 to 6 calculate the distance for each centroid to every point using the information from $\mathbf{G}$. Then, lines 7 and 8 calculate the index of the closest centroid to $\mathbf{x}_n$. Line 9 obtains the output corresponding to the data points from cluster G_m (obtaining all the training data

points for $KNN - \overline{WH}$). Finally, line 10 estimates the trip distances using the $KNN - \overline{WH}$ model.

Algorithm 1 : $k - KNN - \overline{WH}$ (k-means & $KNN - \overline{WH}$)

1: **Input:** $\mathbf{X}$, $\mathbf{t}$, $\mathbf{G}$, N_p, $\mathbf{x}_n$
2: **Output:** $\hat{t}_n$ (time travel prediction)
3: #Assigning the data point to a cluster
4: **for** i from 1 to $|\mathbf{G}| \equiv N_k$ **do**
5: $\quad d_{i,n}$ = distance between centroid $\mathbf{c}_i$ and $\mathbf{x}_n$
6: **end for**
7: # Obtaining the index of the closest centroid to $\mathbf{x}_n$
8: $m = \arg\min_i d_{i,n}$
9: $\mathbf{X}_{G_m}, \mathbf{t}_{G_m}$ = training data corresponding to cluster G_m
10: $\hat{t}_n = KNN - \overline{WH}(\mathbf{X}_{G_m}, \mathbf{t}_{G_m}, N_p, \mathbf{x}_n)$.

$$w_i = \left(\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_n)} \frac{1}{d(\mathbf{x}_n, \mathbf{x}_j)} \right)^{-1} \frac{1}{d(\mathbf{x}_n, \mathbf{x}_i)}, \quad \forall \mathbf{x}_i \in \mathcal{N}(\mathbf{x}_n) \tag{2}$$

$$\hat{t}_n = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_n)} \frac{d_n}{d_i} w_i t_i \tag{3}$$

4 Experimental methodology

In this section, we describe the dataset, evaluation, and experiment to demonstrate the speedup in the estimation processing time of our proposal.

This paper uses the Fantaxico data set available in the GitHub repository 'k-KNN-WH', and previously used in [14, 15, 21, 45]. The Fantaxico data set corresponds to a real dataset with 356,930 taxi trips from Santiago, Chile [21]. The dataset has four variables corresponding to the latitude and longitude of the origin and destination (O-D pairs), one variable corresponds to the trajectory distance, and the final variable is the travel time of the trip (our objective variable). The dataset corresponds to taxi trips within the latitude -33.87 to -33.15 and longitude -70.98 to -70.40, corresponding to Santiago, Chile. The trips were collected over weekdays from three different months in different years (March 2014, 2015, 2016, July 2014, 2015, 2016, and November 2014, 2015). The trips have a distance of at least 30 meters, an average speed of less than 110 kilometers per hour, and a trip time of less than 3 hours.

We evaluated our framework $k - KNN - \overline{WH}$, using two different metrics, time in minutes and Mean Absolute Percentage Error (MAPE), for different numbers of clusters. The time corresponds to the original prediction time of the $KNN - \overline{WH}$ model against our framework's training (k-means) and prediction time. We also estimate the MAPE performance over the dataset to corroborate that $k - KNN - \overline{WH}$ still performs well for time travel prediction. The MAPE is given by equation (4), and it corresponds to the average percentage error of the prediction ($\hat{t}_i - t_i$) with respect to the real value (t_i). One of the main problems of the MAPE is the undefined value of the equation when $t_i = 0$. However, this is not a problem for this dataset, given that the shortest time equals 1 minute. We discard other metrics, such as mean square error, to keep a fair comparison against the original $KNN - \overline{WH}$, which focused on MAPE [21].

$$MAPE = \frac{1}{N_t} \sum_{i=i}^{N_t} \left| \frac{t_i - \hat{t}_i}{t_i} \right| \times 100 \tag{4}$$

To estimate the metrics (time in minutes and MAPE), we use a k-fold cross-validation with $k = 5$. k-fold is a methodological process where the dataset is separated into k-folds, $k-1$ folds are used as training data, and the remaining fold is used as test data. The process is repeated k times, so each fold is used as test data. After applying k-fold cross-validation, we obtain k different values for training and test, so the final results correspond to the mean and standard deviation over these k folds.

For the experiment, we limited the hyperparameter tuning to the number of clusters. Recall that the main focus of this paper is to speed up the estimation process of the model, keeping a comparable performance. So, to keep a fair comparison, we fixed the number of neighbors to $N_p = 20$, replicating the experiment from [21], and varied the number of clusters N_k from 1 to 30, where $N_k = 1$ corresponds to the $KNN - \overline{WH}$ model.

5 Results

The results of this paper can be observed in Figures 2 to 4. While the first two figures show the time in minutes of the k-means and estimation process, Figure 4 shows the MAPE of our proposal against the baseline ($N_k = 1$). The experiments were run on a 2023 Mac Studio equipped with an Apple M2 Ultra chip, 64 GB of RAM, and macOS Sonoma 14.3.1. The authors implemented all the code, which is available in the following GitHub repository https://github.com/Sofialcist/k-KNN-WH.

As can be observed in Figure 2, the average k-means time varies between 0.0 and 0.3 seconds. Note that this time is offline. This implies that the training process of the k-means can be applied before we get any set of data points to predict (test set). For this reason, we omit k-means in the final prediction time. Also, the small standard deviation shows that all times are stable.

Figure 3 shows the mean and standard deviation of the total times in minutes to analyze the test fold using the $KNN - \overline{WH}$ model. As can be observed, when $N_k = 1$, the original $KNN - \overline{WH}$ can take up to 54 minutes to predict one fold of test data (approximately 71,386 data points). In contrast, increasing the number of clusters (N_k) considerably reduced the time to make the prediction, reducing from 54 to 34 minutes by just separating the data into two clusters. As can be expected, the reduction of time can be even larger when N_k increases, decreasing the time to approximately 20 minutes with $N_k = 4$. However, the impact on the reduction time is smaller when N_k increases, and with $N_k > 20$, it starts to plateau around 10 minutes. Similarly to the training time of the k-means, the small standard deviations show that the estimation times are stable.

Figure 3 shows the mean and standard deviation of the MAPE performance using the $k - KNN - \overline{WH}$ framework on the test data, where $N_k = 1$ corresponds to the original $KNN - \overline{WH}$ model. Note, the training data has MAPE=0 because the closest neighbor of a training data point is itself, and an infinite importance is assigned in the prediction. As expected, the performance decreases when the number of clusters is increased. This behavior can be explained

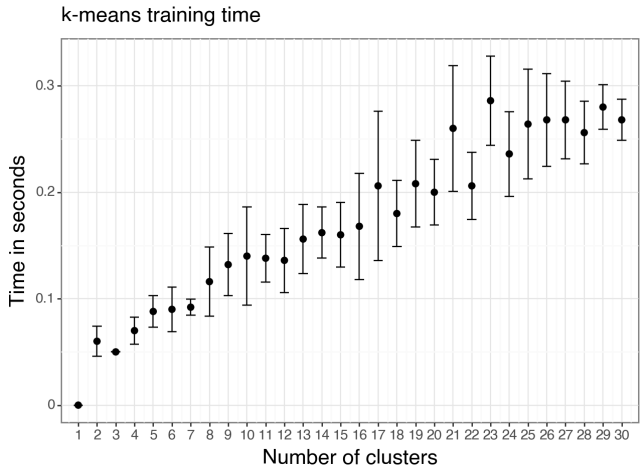

Figure 2: first stage of the $k-KNN-\overline{WH}$ framework: k-means training time in minutes using 1 to 30 clusters, in the training set.

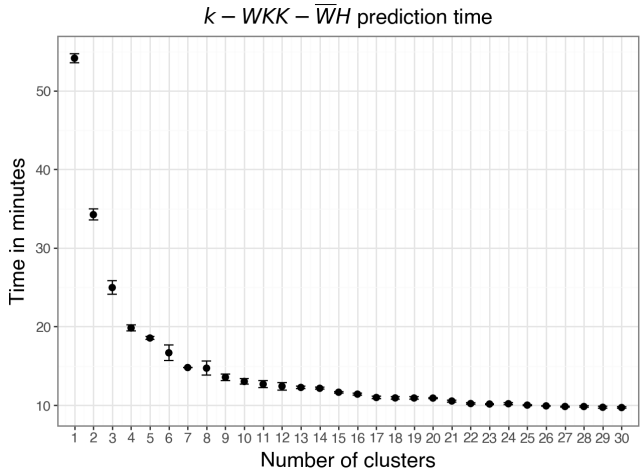

Figure 3: Second stage of the $k - KNN - \overline{WH}$ framework: $k - KNN - \overline{WH}$ prediction time in minutes, in the test set, using 1 to 30 clusters.

by the lower number of data points used to predict each data point. However, as can be observed in the plot, even though the performance decrease is statistically significant, the MAPE increase is minimal compared to the time that can be saved using the proposed framework. Just as an example, if we use $N_k = 4$, increasing the MAPE from 35.98%±0.16% to 37.54%±0.18% (a 1.56% difference), the time prediction is reduced from approximately 50 to 20 minutes, a 60% decrease with respect to the original time.

6 Conclusions

In this paper, we propose $k - KNN - \overline{WH}$, a two-step framework (k-means and $KNN - \overline{WH}$), to speed up the prediction time of the

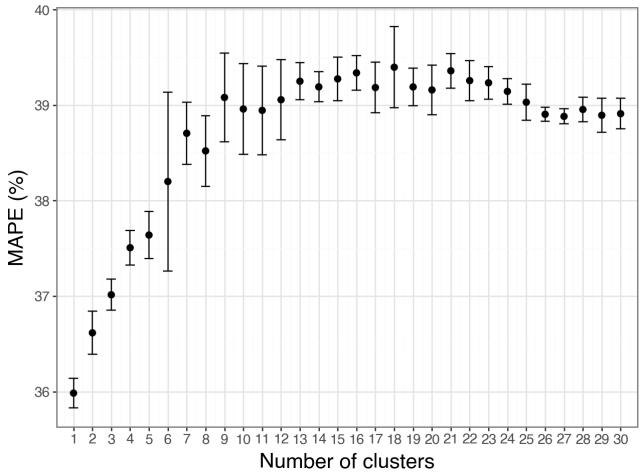

Figure 4: MAPE performance of the $k-KNN-\overline{WH}$ framework, in the test set, using 1 to 30 clusters.

$KNN - \overline{WH}$ model, with minimal impact on the original performance. The framework starts by clustering the original training data with k-means using N_k clusters. Then, when a new set of data points is used as input for prediction, each data point is assigned to one of the N_k clusters, and the $KNN - \overline{WH}$ model is used to predict the travel time with $N_p = 20$ neighbors.

The empirical study shows an increment of MAPE from 35.98% to 38.91% for $N_k = 30$ while decreasing the prediction time from 54 to 10 minutes. Although minimal hyperparameter tuning was performed, setting $N_k = 4$ produces a negligible impact on the performance (1.56%), but a significant time reduction (30 minutes).

The proposed $k - KNN - \overline{WH}$ framework shows promising results, but the framework can still be improved. First, instead of using $N_p = 20$ for all the clusters, a hyperparameter tuning over N_p for each of the N_k clusters should reduce the overall MAPE. This reduction could lead to even better results than the original model. We also propose to change the k-means model for OD-means. OD-means is a two-step hierarchical clustering process able to obtain general and local patterns on O-D data points. Better clusters over the data should also help to improve the model's MAPE performance.

Acknowledgments

This work was funded by the Agencia Nacional de Investigación y Desarrollo (ANID), Fondo Nacional de Desarrollo Científico y Tecnológico (FONDECYT) of the Chilean government under grant number 11230076.

References

[1] Charu C. Aggarwal. 2015. *Data Mining - The Textbook.* Springer.

[2] Ethem Alpaydin. 2021. *Machine learning.* MIT press.

[3] Michael GH Bell. 1984. Log-linear models for the estimation of origin-destination matrices from traffic counts: an approximation. In *the Ninth International Symposium on Transportation and Traffic Theory.*

[4] ME Ben-Akiva. 1987. Methods to combine different data sources and estimate origin-destination matrices. *Transportation and traffic theory* (1987).

[5] Dimitris Bertsimas, Arthur Delarue, Patrick Jaillet, and Sébastien Martin. 2019. Travel time estimation in the age of big data. *Operations Research* 67, 2 (2019), 498–515. doi:10.1287/opre.2018.1784

[6] Lucio Bianco, Giuseppe Confessore, and Pierfrancesco Reverberi. 2001. A network based model for traffic sensor location with implications on O/D matrix estimates. *Transportation Science* 35, 1 (2001), 50–60.

[7] N Caceres, JP Wideberg, and FG Benitez. 2007. Deriving origin–destination data from a mobile phone network. *IET Intelligent Transport Systems* 1, 1 (2007), 15–26. doi:10.1049/iet-its:20060020

[8] Xiqun Michael Chen, Majid Zahiri, and Shuaichao Zhang. 2017. Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach. *Transportation Research Part C: Emerging Technologies* 76 (2017), 51–70. doi:10.1016/j.trc.2016.12.018

[9] Lin Cheng, Senlai Zhu, Zhaoming Chu, and Jingxu Cheng. 2014. A Bayesian Network Model for Origin-Destination Matrices Estimation Using Prior and Some Observed Link Flows. *Discrete Dynamics in Nature and Society* 2014, 1 (2014), 192470. doi:10.1155/2014/192470

[10] Esteve Codina and Jaume Barceló. 2004. Adjustment of O–D trip matrices from observed volumes: an algorithmic approach based on conjugate directions. *European Journal of Operational Research* 155, 3 (2004), 535–557.

[11] Michael Florian and Yang Chen. 1995. A coordinate descent method for the bi-level OD matrix adjustment problem. *International Transactions in Operational Research* 2, 2 (1995), 165–179.

[12] Irfan Ul Haq, Omair Shafiq, Muhammad Muneeb, et al. 2022. Travel time prediction using hybridized deep feature Space and machine learning based heterogeneous ensemble. *IEEE Access* 10 (2022), 98127–98139.

[13] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society, series c (applied statistics)* 28, 1 (1979), 100–108.

[14] Cristobal Heredia, Sebastian Moreno, and Wilfredo Yushimito. 2024. ODMeans: An R package for global and local cluster detection for Origin–Destination GPS data. *SoftwareX* 26 (2024), 101732. doi:10.1016/j.softx.2024.101732

[15] Cristóbal Heredia, Sebastián Moreno, and Wilfredo F Yushimito. 2021. Characterization of mobility patterns with a hierarchical clustering of origin-destination gps taxi data. *IEEE Transactions on Intelligent Transportation Systems* 23, 8 (2021), 12700–12710. doi:10.1109/TITS.2021.3116963

[16] Per Högberg. 1976. Estimation of parameters in models for traffic prediction: a non-linear regression approach. *Transportation Research* 10, 4 (1976), 263–265.

[17] J Holm, T Jensen, SK Nielsen, A Christensen, B Johnsen, and G Ronby. 1976. Calibrating traffic models on traffic census results only. *Traffic Engineering & Control* 17, 4 (1976).

[18] Jilin Hu, Bin Yang, Chenjuan Guo, Christian S Jensen, and Hui Xiong. 2020. Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks. In *IEEE 36th International conference on data engineering.* 1417–1428. doi:10.1109/ICDE48307.2020.00126

[19] Ishan Jindal, Xuewen Chen, Matthew Nokleby, Jieping Ye, et al. 2017. A unified neural network approach for estimating travel time and distance for a taxi trip. *arXiv preprint arXiv:1710.04350* (2017). doi:10.48550/arXiv.1710.04350

[20] Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Michael Chen. 2017. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation research part C: Emerging technologies* 85 (2017), 591–608. doi:10.1016/j.trc.2017.10.016

[21] Felipe Lagos, Sebastián Moreno, Wilfredo F. Yushimito, and Tomás Brstilo. 2024. Urban Origin–Destination Travel Time Estimation Using K-Nearest-Neighbor-Based Methods. *Mathematics* 12, 8 (2024). doi:10.3390/math12081255

[22] Louis Lamarre. 1977. *Une méthode linéaire simple d'estimation de la matrice des orignes et des destinations à partir de comptages sur les liens d'un réseau: une application au réseau routier du Québec.* Université de Montréal, Centre de recherche sur les transports.

[23] Shan Liu, Ya Zhang, Zhengli Wang, Xiang Liu, and Hai Yang. 2025. Personalized origin–destination travel time estimation with active adversarial inverse reinforcement learning and Transformer. *Transportation Research Part E: Logistics and Transportation Review* 193 (2025), 103839.

[24] Dana E Low. 1972. New approach to transportation systems modeling. *Traffic quarterly* 26, 3 (1972).

[25] Yongmei Lu and Yu Liu. 2012. Pervasive location acquisition technologies: Opportunities and challenges for geospatial studies. *Computers, Environment and Urban Systems* 36, 2 (2012), 105–108. doi:10.1016/j.compenvurbsys.2012.02.002

[26] Marcela Munizaga, Carolina Palma, and Pamela Mora. 2010. Public transport OD matrix estimation from smart card payment system data. In *The 12th World Conference on Transport Research.*

[27] Marcela A Munizaga and Carolina Palma. 2012. Estimation of a disaggregate multimodal public transport Origin–Destination matrix from passive smartcard data from Santiago, Chile. *Transportation Research Part C: Emerging Technologies* 24 (2012), 9–18.

[28] Yu Nie, Hui-Min Zhang, and WW Recker. 2005. Inferring origin–destination trip matrices with a decoupled GLS path flow estimator. *Transportation Research Part B: Methodological* 39, 6 (2005), 497–518.

[29] Pierre Robillard. 1975. Estimating the OD matrix from observed link volumes. *Transportation research* 9, 2-3 (1975), 123–128.

[30] Havard Rue and Leonhard Held. 2005. *Gaussian Markov random fields: theory and applications.* Chapman and Hall/CRC. doi:10.1201/9780203492024

[31] Zhaoyu Sheng, Zhiqiang Lv, Jianbo Li, Zhihao Xu, Haokai Sun, Xue Liu, and Rongkun Ye. 2023. Taxi travel time prediction based on fusion of traffic condition features. *Computers and Electrical Engineering* 105 (2023), 108530.

[32] Yunsheng Song, Jiye Liang, Jing Lu, and Xingwang Zhao. 2017. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing* 251 (2017), 26–34.

[33] Heinz Spiess. 1987. A maximum likelihood model for estimating origin-destination matrices. *Transportation Research Part B: Methodological* 21, 5 (1987), 395–412.

[34] Heinz Spiess. 1990. A gradient approach for the OD matrix adjustment problem. *Centre de Recherche sur les Transports de Montréal* 1 (1990), 2.

[35] J Symons, R Wilson, and J Paterson. 1976. A model of inter city motor travel estimated by link volumes. In *Australian Road Research Board (ARRB) Conference, 8th, 1976, Perth*, Vol. 8.

[36] OZ Tamin and LG Willumsen. 1989. Transport demand model estimation from traffic counts. *Transportation* 16 (1989), 3–26.

[37] Keshuang Tang, Yumin Cao, Can Chen, Jiarong Yao, Chaopeng Tan, and Jian Sun. 2021. Dynamic origin-destination flow estimation using automatic vehicle identification data: A 3D convolutional neural network approach. *Computer-Aided Civil and Infrastructure Engineering* 36, 1 (2021), 30–46.

[38] Henk J Van Zuylen and David M Branston. 1982. Consistent link flow estimation from counts. *Transportation Research Part B: Methodological* 16, 6 (1982), 473–476.

[39] Henk J Van Zuylen and Luis G Willumsen. 1980. The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological* 14, 3 (1980), 281–293.

[40] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–22. doi:10.1145/3293317

[41] Yun Wang, Faiz Currim, and Sudha Ram. 2022. Deep learning of spatiotemporal patterns for urban mobility prediction using big data. *Information Systems Research* 33, 2 (2022), 579–598. doi:10.1287/isre.2021.1072

[42] Yue Wang, Enjian Yao, Yongsheng Zhang, Long Pan, and He Hao. 2024. Deep Learning Model for Short-Term Origin–Destination Distribution Prediction in Urban Rail Transit Network Considering Destination Choice Behavior. *Transportation Research Record* (2024). doi:10.1177/03611981241243

[43] M Wills. 1977. Nonlinear estimation of origin-destination flows and gravity model parameters from traffic counts on links. *Department of Geography, University of British Columbia, Mimeographed Notes* (1977).

[44] Xi Xiong, Kaan Ozbay, Li Jin, and Chen Feng. 2020. Dynamic origin–destination matrix prediction with line graph neural networks and kalman filter. *Transportation Research Record* 2674, 8 (2020), 491–503. doi:10.1177/03611981209193

[45] Wilfredo F. Yushimito, Sebastian Moreno, and Daniela Miranda. 2023. The Potential of Battery Electric Taxis in Santiago de Chile. *Sustainability* 15, 11 (2023). doi:10.3390/su15118689

[46] Yanru Zhang and Ali Haghani. 2015. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies* 58 (2015), 308–324. doi:10.1016/j.trc.2015.02.019

[47] Zhihan Zheng, Haitao Yuan, Minxiao Chen, and Shangguang Wang. 2025. RLER-TTE: An Efficient and Effective Framework for En Route Travel Time Estimation with Reinforcement Learning. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–26.

[48] Zhi-Hua Zhou. 2021. *Machine learning.* Springer nature.