Abstract—A key problem in location-based modeling and forecasting lies in identifying suitable spatial and temporal resolutions. In particular, judicious spatial partitioning can play a significant role in enhancing the performance of location-based forecasting models. In this paper, we investigate two widely used tessellation strategies for partitioning city space, in the context of real-time taxi demand forecasting. Our study compares (i) Geohash tessellation, and (ii) Voronoi tessellation, using two distinct taxi demand data sets, over multiple time scales. For the purpose of comparison, we employ classical time-series tools to model the spatio-temporal demand. Our study finds that the performance of each tessellation strategy is highly dependent on the city geography, spatial distribution of the data, and the time of the day, and that neither strategy is found to perform optimally across the forecast horizon. We propose a hybrid algorithm that picks the best tessellation strategy at each instant, based on their recent performance. Our hybrid algorithm is a non-stationary variant of the well-known HEDGE algorithm for choosing the best advice from multiple experts. We show that the hybrid strategy performs consistently better than either of the two tessellation strategies across the data sets considered, at multiple time scales, and with different performance metrics. We achieved an average accuracy of above 80% per km² for both data sets considered at 60 minute aggregation levels.

Keywords—Taxi Demand, Forecasting, Time-series, Geohash, Voronoi, HEDGE.

I. INTRODUCTION

Mobile application based e-hailing taxi services are gaining popularity across the world due to the advancement in GPS based smart phone technologies. These services can be used to supplement the use of public transit and other traditional modes of transportation. A key challenge faced by these fast growing taxi services is the demand-supply mismatch problem. During peak hours, the demand for taxis surpass the available supply, creating unmet demand. During off-peak hours, the scenario reverses and the vacant taxis cruise for longer periods to find passengers. These issues lead to dynamic surge pricing, along with reduced customer satisfaction and low driver utilization. Therefore, it is crucial to devise accurate location-specific demand forecasting algorithms, so as to gain prior knowledge of under-supplied and over-supplied areas. This knowledge can be used to mitigate the demand-supply imbalance by re-routing vacant taxis, to compensate for the unmet demand. One of the key steps towards devising efficient location-based forecasting algorithms is the selection of appropriate spatial resolution techniques. Each cell should contain sufficient demand density for prediction algorithms to be effective. At the same time, the driver should not travel a large distance before finding a passenger. Hence, a carefully planned tessellation strategy is a key step in any location-based modeling exercise.

A. Related works

Demand-supply levels in taxi services have been studied in the literature [1], [2]. Identification and modeling of passenger hot spots for rerouting taxi drivers is also a widely researched area. Some of the commonly employed modeling techniques are the Auto Regressive Integrated Moving Average model (ARIMA) and its variants [3], [4], Exponential Weighted Average (EWA) models [5], Nearest Neighbour clustering [6], [7], and Neural Networks (NN) [8]. Irrespective of the modeling technique employed, the preliminary step in modeling a location-based entity such as passenger demand is to spatially partition the space. In the transportation literature, two common tessellation strategies are used. Spatial aggregations are either performed using grids, where the space is partitioned into square or rectangular grids of fixed area [3], [8], [9]; or using polygons, where the space is partitioned into regular or irregular polygons of variable area [10], [11], [12]. It is common practice in the transportation literature to consider either one of these tessellation styles for spatial partitioning.

In our previous work [13], we tessellated the city of Bengaluru, India, into fixed-sized partitions known as geohashes, obtained from Geohash tessellation. We observed that for those regions with low demand density, fixed sized partitions resulted in data scarcity, which led to low model accuracy. To improve the accuracy, we explored the spatial correlation between the neighbouring geohashes. The performance limitation due to the chosen tessellation strategy motivates us to conduct a comparison study of various tessellation strategies. It is natural to ask the following questions in this context: (i) How can one decide the tessellation scheme to be used?, (ii) How sensitive is the performance of the models to the tessellation strategy?, (iii) Can we arrive at a tessellation strategy that performs well for a broad range of data sets? We aim to address these questions in this paper. To the best of our knowledge, an extensive study of tessellation techniques and their effects on model performance have not been conducted in the past. In this paper, we explore the relationships between tessellation strategies, demand densities, and city geographies. This is one of the features that distinguishes our work from the existing literature.

For comparing fixed and variable sized tessellation styles, we choose Geohash tessellation, and Centroidal Voronoi tessellation with K-Means [16] respectively. We note that each tessellation strategy has been shown to outperform the other.
In [14], [15], the authors showed that the Geohash technique performs better at partitioning data than the Voronoi technique for their data set, while in [15], the authors preferred Voronoi over Geohash for forecasting supply of drivers in the demand dense localities. In this paper, we favour a partition based clustering technique such as K-Means [16] over other clustering techniques. The reader is referred to [17] and [18] for a comprehensive survey of various clustering algorithms. In the survey paper [18], K-Means is listed as a promising clustering algorithm for large data sets, due to its low time complexity, and high computing efficiency. K-Means has a linear memory and time complexity, which is ideal for our very large data sets. It has also been shown that K-Means performs reasonably well in comparison with other clustering techniques; for example, DBSCAN [20] and Hierarchical clustering [19], among others. In addition, K-Means has been widely used in the literature to generate Centroidal Voronoi polygons [21]. Of the widely used modeling techniques, Exponential Smoothing and ARIMA models are known for their accuracy, ease of implementation, and high computational speed [22]. Their modeling performance have been shown to be comparable to other methods such as Bayesian networks, Support Vector Regression (SVR), and Artificial Neural Networks (ANN) [23], [24]. The authors in [3] proposed an improved ARIMA based prediction method to forecast the spatial-temporal variation of passengers in the hot spots with a prediction error of 5.8%. Using a time varying Poisson process and ARIMA model, the work in [4] obtained an accuracy of 76% for passenger demand on taxi services. Clustering along with Exponential Weighted Moving Average models were used to predict passenger demand hot spots with a 79.6% hit ratio in [5]. The subway ridership demand was analysed in [25] using a combined ARIMA-GARCH model with a maximum error of 7%. This promising performance of regression and smoothing based models in the context of passenger demand modeling, along with high computational speeds, make them appealing choices for our comparison study.

We aim to perform an extensive city wide spatio-temporal analysis and compare the two tessellation techniques for two independent data sets – an e-hailing mobile application based demand data set in Bengaluru, India and a street hailing based taxi demand data set in New York city, USA. Further, in addition to comparing the two strategies, we also propose a hybrid strategy by using a combination of the two tessellation strategies based on their past performances, suitable for any data set. To the best of our knowledge, no similar study has been made in the transportation literature to use a combination of tessellation strategies.

**B. Our contributions**

The demand data points are segregated into clusters using the K-Means clustering algorithm. The cluster centroids that we obtain are used to partition the city into geohashes and Voronoi cells. In previous studies, road intersections [10], [12] and bus stops [26] were employed to act as tessellation centers. In contrast, we use K-Means cluster centroids to act as the tessellation centers. Different time-series modeling techniques are applied on the Geohash and the Voronoi aggregated data from each centroid to compare the two techniques. The data is aggregated and analyzed at two different time-scales; at 15 minute to enable real-time decisions, and at 60 minute to observe patterns on longer time scales. While comparing the two tessellation strategies, we observe that the performance of the strategies vary with different data sets. They also showed time dependent variations within each data set. In order to deal with this apparent non-stationarity, we need an algorithm that can pick the best tessellation strategy at every time instant.

The method of using advice from multiple experts was first introduced in [27], and later generalized in [28] to arrive at the well-known HEDGE algorithm. Specifically, by adopting a multiplicative update of the weight parameter, the authors of [28] were able to produce algorithms performing almost as well as the best expert in the pool. The authors in [29] extended the HEDGE algorithm to include non-stationary experts, by introducing a discounting factor. We modify the discounted variant of HEDGE to arrive at a hybrid tessellation strategy for enhanced taxi demand forecasts.

The key contributions and findings of this paper can be summarized as follows:

1. We conduct a comparative study of Voronoi and Geohash tessellation strategies for spatial demand partitioning.
2. We highlight the dependence of the performance of the tessellation strategy on the time of the day, and on the properties of the data set.
3. We find that models based on Voronoi tessellation have superior performance compared to models based on Geohash tessellation for demand scarce cells. On the other hand, Geohash tessellation performs better than Voronoi tessellation in demand dense cells.
4. We develop a hybrid tessellation strategy using a HEDGE based combining algorithm. We find that our hybrid strategy always performs at least as well as the best strategy out of the list of experts.
5. Our algorithm picks the best tessellation strategy for each instant in the forecasting horizon, across various temporal and spatial resolutions.

At a broader level, our work demonstrates the potential of improving location-based forecasts by efficiently partitioning large-scale temporal and spatial data for taxi hailing services. The rest of the paper is organized as follows. In Section II, we outline the problem setting and provide a brief description of the data. In Section III, the tessellation strategies are outlined. The time-series modeling procedure is explained, along with some preliminary results in Section IV. The proposed model combining algorithm and the results are provided in Section V, and we conclude in Section VI.

**II. Problem setting**

Let \( y_t \) be the number of taxi bookings (i.e., the demand) observed at time \( t \) in a particular region \( R \). In order to model \( y_t \), we can represent the bookings as a time-sequence, where the sequence contains points aggregated over the space \( R \) at

---

1In our case, the “experts” refer to the tessellation strategies.
time \( t_1, \ldots, t_n \). The space \( R \) can be fixed-sized or variable-sized. In our case, we denote them as geohash and Voronoi cells respectively. Geohash partitioning of a city is straightforward, as it does not incorporate neighbour information or the volume of the data. On the other hand, to perform Voronoi partitioning, we need to run a clustering algorithm to cluster nearby points together. In order to reduce the passenger search time, we aim to route drivers to regions of area no more than 1 km\(^2\). The parameters in the clustering algorithm are set accordingly. We define the density of a K-Means cluster centroid as the number of data points closer to that centroid than to any other centroid. We assume that a unique smallest distance K-Means centroid exists for each data point.

For an idle taxi, we aim to provide the driver with the location of the nearest suitable K-Means centroid. The demand aggregated from a Voronoi cell may vary in magnitude from the demand aggregated from its corresponding geohash. Hence, for a standard comparison of the two techniques, we normalize the demand by the area of the partition to obtain demand per km\(^2\). The area normalized demand \( D_{\text{norm}} \) for the \( P \)th geohash/Voronoi partition is computed as follows:

\[
D_{\text{norm}} = \frac{\text{aggregate}(d) \text{ over } sp \text{ period}}{\text{area}(P)},
\]

where \( d \) refers to the demand in \( P \), \( sp \) is the sampling period of 15 minute or 60 minute intervals. This \( D_{\text{norm}} \) is used to generate the time-sequences for further analysis. The Symmetric Mean Absolute Percent Error (SMAPE) and Mean Absolute Scaled Error (MASE) [30] are the error performance metrics that are considered in this paper, and are defined in Section IV. See Fig. 1 for a schematic representation of the work that will be conducted in this paper. First, we tessellate the city using Voronoi and Geohash strategies. Then, we apply time-series modeling techniques on the data, and finally, we develop a hybrid tessellation strategy based on the dHEDGE algorithm. The data sets used for our study are mentioned below.

A. Data set description

The Bengaluru taxi demand data is acquired from a leading Indian e-hailing taxi service provider. The data contains GPS traces of taxi passengers booking a taxi by logging into the mobile application. The data is available for a period of two months, 1\(^{st} \) of January 2016 to 29\(^{th} \) of February 2016. The data set contains latitude-longitude coordinates of the logged in customer, along with his/her identification number, session duration and time stamp. The latitude and longitude coordinates of the city are 12.9716 N, 77.5946 E, with an area of approximately 740 km\(^2\).

The New York yellow taxi cab data set is publicly available at [31]. The data set contains GPS traces of a street hailing yellow taxi service. This data set differs from the mobile application based data set, both in terms of the volume of the data and the city structure. The geographical structure of Bengaluru is radial, while that of New York city is linear. We considered the period of January-February 2016, for analysis. We extract the pick up locations and time stamps from the data to form the demand data. The latitude and longitude coordinates of New York city are 40.7128 N, 74.0059 W, with an area of approximately 780 km\(^2\).

III. TESSELLATION STRATEGIES

As motivated in Section II, we perform spatial partitioning of both cities using two tessellation strategies for the purpose of modeling and comparison. We generate clusters of demand points for each city, and then, the centroids of these clusters aid in the generation of tessellation cells.

A. K-Means algorithm

K-Means [16] is a widely used unsupervised learning algorithm to classify a given data set into a certain number of clusters (assume \( k \) clusters) fixed apriori. This algorithm aims to minimize the squared error function given as:

\[
J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||y_{(i)}^{j} - c_{j}||^2,
\]

where \( ||y_{(i)}^{j} - c_{j}||^2 \) is the Euclidean distance between a data point \( y_{(i)}^{j} \) and its center \( c_{j} \), and \( n \) is the total number of data points. For ensuring that the average cluster area remains close to 1 km\(^2\), the number of centers \( K \) is set to 740 and 780 for Bengaluru and New York city respectively. For each data point, the algorithm calculates the distance from the data point to each cluster. If the data point is closest to its own cluster, we leave it where it is, else, we move it into the closest cluster. The algorithm stops when no data point is reassigned. By sorting these clusters in descending order of density, we can find locations that generate relatively higher demand (hot spots) compared to other locations.

B. Voronoi tessellation

Voronoi tessellation is a spatial partitioning method that divides space according to the nearest neighbour-rule. Specifically, each point, called a seed or site, is associated with the region (i.e., the Voronoi cell) that is closer to it than all other points in the space. A Voronoi tessellation is called a Centroidal Voronoi tessellation when the generating site of each Voronoi cell is also its mean (center of mass). In our work, these sites are obtained from the K-Means algorithm. Based on the closeness of sites, this tessellation strategy produces polygon partitions of varying areas. For example, the 740 demand centers result in 740 variable-sized quadrilateral partitions for Bengaluru. Note that the overall time complexity of the Voronoi and K-Means algorithm is \( O(n \log n) \). See Fig. 2a and Fig. 3a for heat maps.
generated from Voronoi tessellations of Bengaluru and New York city. The partitioned cells are color-coded according to their sample size, for ease of representation on the color scale.

C. Geohash tessellation

Geohash is a technique that hashes the latitude and longitude coordinates into a character string. It is an extension of the grid based method, with a simple naming convention. Note that with Geohash (first G capitalized), we refer to the technique of encoding a coordinate pair into a single string, while geohash (all small letters) refers to the string itself. Geohash can be visualized as a division of Earth into 32 planes, each of which can be divided again into 32 planes, and so on. We refer to these divisions as Geohash levels by defining level x as the division that results in geohashes of string length x. A 6-level geohash spans a grid of area 1.2 km × 0.6 km, covering approximately 1 km², while a 5-level geohash spans an area of 4.9 km × 4.9 km, covering approximately 25 km². We choose 6-level grids over 5-level for better re-routing of idle taxis. In terms of time complexity, this algorithm is O(1). See Fig. 2b and Fig. 3b for heat maps obtained from Geohash tessellations of Bengaluru and New York city.

D. Observations

1) Bengaluru data set

On observing the heat maps generated by the two tessellation techniques from Fig. 2, we make the following inferences. Geohash is a region-oriented city map partition approach, and therefore results in some cells that are highly dense, and in some cells that are highly sparse. On the other hand, Voronoi cells are more uniformly distributed in terms of density (see Fig. 4a and Fig. 4b). The Voronoi tessellation technique tries to uniformly distribute samples among the partitions. This, in fact, increases the chance of finding a passenger in a cell as there are fewer “demand scarce” cells. On the other hand, this process of uniformly distributing samples increases the partition size to above 1 km² for some Voronoi cells, as seen in Fig. 4a. As a result, the driver might have to traverse a larger distance to find a passenger. The size of the smallest Voronoi partition observed is 0.10 km² and 85% of the partitions are below 2 km². The area of a geohash cell remains constant at 0.72 km². So when the driver is re-routed to a cell, a vacant taxi has to spend less time searching in a geohash when compared to its corresponding Voronoi cell, provided that sufficient demand density exists in the cell. However, at times, this advantage comes at the expense of masking the actual demand hot spots. In some locations, typically near the city center, there can be more than one hot spot in 1 km² area. Due to the inflexible structure of a geohash, these multiple hot spots are considered as a single hot spot. Note that if the service providers are more inclined towards re-allocating drivers to fixed sized locations, multiple hot spots in a geohash might not be an issue. With Voronoi technique, we have the added flexibility of slicing one geohash into further cells. So from the perspective of identifying a hot spot, the Voronoi tessellation seems to be a better strategy than the Geohash tessellation for the Bengaluru data set.
Before modeling the data, we remove duplicate user IDs, if they appear multiple times in a 30 minute interval per partition. We assume that it is unlikely for a passenger to book multiple taxis within that time, from the same cell. A Box-Cox transformation is applied to stabilize the variance of the raw data, thus making the data amenable for linear processing [32]. Each K-Means demand centroid has two time-series associated with it; one using the demand aggregated at the Voronoi cell level, and the other with the demand aggregated at the geohash level. When these sequences are plotted, we observe that the series shows significant trend and seasonal patterns. On performing a spectral analysis, we find strong daily and weekly seasonality. A computationally efficient approach to real-time forecasting is to use linear time series models. Hence, the data is modeled using single and double-seasonal linear parametric time-series models. The city comprises of various activity zones such as residential, entertainment, office, school zones, etcetera. The demand patterns for each of these zones differ, and hence a single time-series model may not be a satisfactory fit for all the time-sequences. We perform the time-series modeling exercise for each tessellation strategy separately, at two time scales: 15 minute and 60 minute.

A. Shortlisted models

Some of the widely used single-seasonal exponential smoothing models are Holt-Winters, Auto-Regressive and Seasonal Auto-regressive Integrated Moving Average (ARIMA and SARIMA) models, Seasonal and Trend Decomposition using LOESS (STL) combined with non-seasonal exponential smoothing [32, Chapters 6-8]. Some of the common double-seasonal models are Double Seasonal Holt-Winters (DSHW) [32] and Trigonometric BATS (TBATS) [33]. In order to make sure that these models work better than naïve alternatives, we compare the aforementioned models against a simple averaging model, the Nāive and Seasonal Nāive models, and the Random drift model. It was observed that in a few cells, the simple alternatives indeed performed better than the parametric time-series models. We consider a simple seasonal averaging model as the baseline model for our analysis. If a model performs better than the baseline model, it is kept, else it is discarded. We now briefly explain the models with which almost all the activity zones are well-modeled for our data.

1) Baseline model: In order to forecast for a particular time step, the mean of all the previous season samples corresponding to that time step is computed. Our baseline model is as follows:

\[ y_{t+1} = \frac{1}{S} \sum_{i=1}^{S} y_{t+1-im}, \]  

(3)

where \( y_t \) is the demand at time \( t \), \( S \) is the number of seasons and \( m \) is the seasonality period. For example, if weekly forecast is considered, forecast for a Sunday at 12 a.m. is the average of all Sunday 12 a.m. demands.

2) New York data set

Fig. 3 is a zoomed-in heat map of the Manhattan borough. We note that 92.5% of the total generated demand originated from Manhattan over the period of study. Thus, the spatial data distribution is highly heterogenous, resulting in very closely packed Voronoi cells in the Manhattan borough. In fact, the smallest partition is of the size 0.08 km\(^2\) and 75% of the partitions are below 0.10 km\(^2\), as observed in Fig. 4c. Routing drivers to such a small area is infeasible and economically not viable for any service provider. The Geohash strategy, on the other hand, is not a density-dependent technique and hence, assigns geohashes uniformly. From Fig. 3a, we observe that there are inter-island tessellations, even though all the centroids are in mainland. From the perspective of re-routing drivers, this is not very appealing. This problem does not arise in Fig. 3b and the 6-level geohashes are confined to either sides of the river. Because of the inter-island tessellations, Voronoi does not appear to be a suitable spatial tessellation strategy for New York city, which is spread over multiple islands. If the Voronoi tessellation is to perform better, extensive parameter tuning of the K-Means algorithm will have to be conducted. The Voronoi tessellation has to be performed on each borough to avoid inter-island tessellations, which adds to computational complexity. For this data set, the sheer simplicity of the Geohash tessellation makes it an attractive strategy.

The inferences and comparisons made above are based on the spatial aggregation of the data. In the next section, we perform comparisons based on the temporal aggregated data. The spatially aggregated data for each Voronoi and Geohash partition is temporally aggregated and modeled using time-series techniques.

IV. Time-series modeling

Before modeling the data, we remove duplicate user IDs, if they appear multiple times in a 30 minute interval per partition. We assume that it is unlikely for a passenger to book multiple taxis within that time, from the same cell. A Box-Cox transformation is applied to stabilize the variance of the raw data, thus making the data amenable for linear processing [32]. Each K-Means demand centroid has two time-series associated with it; one using the demand aggregated at the Voronoi cell level, and the other with the demand aggregated at the geohash level. When these sequences are plotted, we observe that the series shows significant trend and seasonal patterns. On performing a spectral analysis, we find strong daily and weekly seasonality. A computationally efficient approach to real-time forecasting is to use linear time series models. Hence, the data is modeled using single and double-seasonal linear parametric time-series models. The city comprises of various activity zones such as residential, entertainment, office, school zones, etcetera. The demand patterns for each of these zones differ, and hence a single time-series model may not be a satisfactory fit for all the time-sequences. We perform the time-series modeling exercise for each tessellation strategy separately, at two time scales: 15 minute and 60 minute.

Some of the widely used single-seasonal exponential smoothing models are Holt-Winters, Auto-Regressive and Seasonal Auto-regressive Integrated Moving Average (ARIMA and SARIMA) models, Seasonal and Trend Decomposition using LOESS (STL) combined with non-seasonal exponential smoothing [32, Chapters 6-8]. Some of the common double-seasonal models are Double Seasonal Holt-Winters (DSHW) [32] and Trigonometric BATS (TBATS) [33]. In order to make sure that these models work better than naïve alternatives, we compare the aforementioned models against a simple averaging model, the Nāive and Seasonal Nāive models, and the Random drift model. It was observed that in a few cells, the simple alternatives indeed performed better than the parametric time-series models. We consider a simple seasonal averaging model as the baseline model for our analysis. If a model performs better than the baseline model, it is kept, else it is discarded. We now briefly explain the models with which almost all the activity zones are well-modeled for our data.

A. Shortlisted models

Some of the widely used single-seasonal exponential smoothing models are Holt-Winters, Auto-Regressive and Seasonal Auto-regressive Integrated Moving Average (ARIMA and SARIMA) models, Seasonal and Trend Decomposition using LOESS (STL) combined with non-seasonal exponential smoothing [32, Chapters 6-8]. Some of the common double-seasonal models are Double Seasonal Holt-Winters (DSHW) [32] and Trigonometric BATS (TBATS) [33]. In order to make sure that these models work better than naïve alternatives, we compare the aforementioned models against a simple averaging model, the Nāive and Seasonal Nāive models, and the Random drift model. It was observed that in a few cells, the simple alternatives indeed performed better than the parametric time-series models. We consider a simple seasonal averaging model as the baseline model for our analysis. If a model performs better than the baseline model, it is kept, else it is discarded. We now briefly explain the models with which almost all the activity zones are well-modeled for our data.

1) Baseline model: In order to forecast for a particular time step, the mean of all the previous season samples corresponding to that time step is computed. Our baseline model is as follows:

\[ y_{t+1} = \frac{1}{S} \sum_{i=1}^{S} y_{t+1-im}, \]  

(3)

where \( y_t \) is the demand at time \( t \), \( S \) is the number of seasons and \( m \) is the seasonality period. For example, if weekly forecast is considered, forecast for a Sunday at 12 a.m. is the average of all Sunday 12 a.m. demands.
in the past. Then, for data aggregated over 60 minute, $S$ is the number of lookback Sundays, and $m$ is 168 (i.e., 24 hours $\times$ 7 days).

2) TBATS: TBATS model is a state space model introduced in [33] for forecasting time-series with multiple seasonal periods, high frequency seasonal periods, non integer seasonality, and calendar effects. TBATS is an acronym for Trigonometric (T), Box-Cox transform (B), ARMA errors $d_t$ (A), Trend $b_t$ (T) and Seasonal components $s_t^{(i)}$ (S). The seasonal components are represented using trigonometric fourier series. The equations for a $h$-step ahead additive trend, multiplicative seasonality TBATS model prediction are as follows:

**Forecast**  
$\hat{y}_{t+h} = (l_t + h b_t) \prod_{i} s_{t-m_i+h}^{i}$

**Level**  
$l_t = \alpha \left( \frac{y_t}{\prod s_{t-m_i}^{i}} \right) + (1-\alpha)(l_{t-1} + b_{t-1})$,

**Trend**  
$b_t = \beta(l_{t} - l_{t-1}) + (1-\beta)(b_{t-1})$,

**Seasonal component**  
$s_{t}^{i} = \sum_{j=1}^{k_i} s_{j,t}^{i}$,

where, $s_{j,t}^{i} = s_{j,t-1}^{i} \cos \lambda_{j}^{i} + s_{j,t-1}^{i} \sin \lambda_{j}^{i} + \gamma^{i}$,

$\lambda_{j}^{i} = \frac{2\pi j}{m_i}$,  

\[ \text{where, } \alpha, \beta, \text{ and } \gamma \text{ are the smoothing parameters, } m_i \forall i \in \{1,\ldots,T\} \text{ are the seasonal periods, } s^{*i} \text{ is the complex conjugate of } s^i, \text{ and } k_i \text{ is the number of harmonics needed for the } i^{th} \text{ seasonal component.} \]

3) Seasonal and Trend decomposition using Loess (STL): STL is a time-series decomposition technique where the data is decomposed into seasonal ($S_t$) and non-seasonal ($NS_t$) components:

$Y_t = S_t + NS_t$,

$NS_t = T_t + R_t$, \hspace{1cm} \(5\)

$S_{T+h} = S_{T+h-km}$; $k = [(h-1)/m] + 1$.

Here, $T_t$ denotes the trend, $R_t$ is the irregular component, $\lfloor u \rfloor$ is the integer part of variable $u$, $h$ is the prediction horizon, and $m$ is the seasonal period. The non seasonal component is modeled using techniques such as ARIMA, ExponenTial Smoothing models (ETS), Random drift, Ñaive, etcetera. The seasonal component, after a Seasonal Ñaive forecast for the seasons, is added to the non-seasonal component to form the final model.

**B. Model validation**

The suitability of the models for the data can be analyzed as follows:

- Plotting the histogram of the residuals: The histogram should follow a Gaussian distribution with residual mean as the mean of the distribution.
- Plotting the Auto Correlation Function (ACF) of the residuals: The residuals should lie within the significant band of the ACF plot, i.e., within $\pm2/\sqrt{N}$ band, where $N$ is the sample size.
- Resemblance with white noise: The residuals should appear to be uncorrelated and should pass a statistical test for correlation; for example, the Box-Ljung test. The Box-Ljung statistic is a function of the accumulated sample auto correlations up to any specified time lag.

We refer the reader to [32, Chapter 2, Section 2.6] for an elaborate explanation of the afore-mentioned residual diagnostic tools. Results of residual diagnostics performed on the Voronoi time-series generated from an entertainment zone in Bengaluru are plotted in Fig. 5. Here, STL decomposition with ARIMA(2,1,2) performed satisfactorily and the residuals from the fit appear to be uncorrelated.

The performance of the models can further be evaluated by employing performance metrics such as Symmetric Mean Absolute Percent Error (SMAPE) and Mean Absolute Scaled Error (MASE). The SMAPE is a symmetrized version of the Mean Absolute Percent Error (MAPE), which is defined if, at all future time points, the point forecasts and actuals are both not zero. For a forecast horizon $N$ considered, SMAPE is defined as follows:

$$\text{SMAPE} = \frac{100}{N} \sum_{t=1}^{N} \frac{|\hat{y}_t - y_t|}{\hat{y}_t + y_t},$$ \hspace{1cm} \(6\)
where $y_t$ is the actual demand and $\hat{y}_t$ is the forecast at time $t$. Even though this metric is widely used, SMAPE is a scale-dependent error, and is not well-suited for intermittent demand. Hence, we also evaluate our models against MASE. For a time-series of forecast horizon $N$ and seasonal period $m$, MASE is defined as:

$$\text{MASE} = \frac{1}{N} \sum_{t=1}^{N} \left( \frac{1}{N-m} \sum_{t=m+1}^{N} \left| y_t - \hat{y}_t \right| \right).$$ (7)

For a non-seasonal time-series, $m = 1$. The denominator of the equation is the mean absolute error of the one-step Naïve forecast method on the training set. This error metric compares the models with the standard Naïve model. A MASE value < 1 ensures that the models perform better than the Naïve technique. MASE can be used to compare forecasts across data sets with different scales, as the metric is independent of the scale of the data. Smaller the value of the metric, the better is the model. The selected models passed the residual diagnostics and performed satisfactorily with SMAPE and MASE. In Table I, we list the shortlisted models with which Bengaluru and New York city data sets are well-modeled at 60 minute aggregation levels. The errors encountered by the individual models are not listed because the modeling technique that works for one scenario might not provide good performance with another scenario.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Bengaluru</th>
<th>New York</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geohash</td>
<td>STL (ETS, a.e., d.t.), STL (ARIMA, BoxCox)</td>
<td>STL (ETS, a.e., d.t.), STL (Random drift)</td>
</tr>
<tr>
<td>Voronoi</td>
<td>STL (ETS, a.e., d.t.), TBATS (ARMA error, BoxCox,trend)</td>
<td>STL (ETS, a.e., d.t.), STL (ARIMA, BoxCox), STL (Naïve)</td>
</tr>
</tbody>
</table>

**Table I:** Best performing models for the data sets, where a.e. means additive errors, and d.t. means damped trend.

The Empirical Cumulative Distribution Function (ECDF) of the average errors obtained from the models based on all the Voronoi and Geohash partitions are plotted in Fig. 7. We see that the Voronoi technique performs better than the Geohash technique at 15 and 60 minute aggregation levels in Fig. 7a and Fig. 7b for Bengaluru. The errors from the Voronoi based models reach the upper bound faster than errors from the Geohash based models. On the other hand, for New York city, Fig. 7c and Fig. 7d show that the Geohash technique has better performance over the Voronoi technique. Additionally, we conducted the Kolmogorov-Smirnov test (K-S test) [34] to check if the ECDFs differ significantly. The results obtained provide sufficient evidence that Voronoi technique is statistically better than the Geohash technique for Bengaluru and vice versa for New York city, consistent with the inferences from Fig. 7. We find that the Geohash tessellation performs better than Voronoi for New York city, while the Voronoi tessellation has better performance over Geohash for Bengaluru. Further, we proceed to plot the instantaneous errors obtained from the models based on the two tessellation techniques in Fig. 8a for Bengaluru at 15 minute aggregation level. Each point on the graph corresponds to the mean of errors obtained from all Geohash/Voronoi cells at that time instant. We can see that during early mornings and late nights, Geohash based models work better compared to Voronoi based models. For the rest of the day, i.e., the daylight hours, Voronoi based

**Table II:** Preliminary results for the Bengaluru and New York city data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Error Metric</th>
<th>60 minute</th>
<th>15 minute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vor</td>
<td>Geo</td>
<td>Vor</td>
</tr>
<tr>
<td>Bengaluru</td>
<td>SMAPE (%)</td>
<td>17.6</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>MASE</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>New York</td>
<td>SMAPE (%)</td>
<td>15.2</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>MASE</td>
<td>0.60</td>
<td>0.59</td>
</tr>
</tbody>
</table>

**Fig. 6:** The variation of SMAPE with the sample size in each partition at 60 minute aggregation level.

**C. Preliminary results**

After tessellating the city space into geohashes and Voronoi cells, we ran the time-series models mentioned in the previous section on the spatially and temporally aggregated area-normalized demand. In general, we observe strong heterogeneity in the city demand across both spatial and temporal dimensions. It was observed that for cells with low density, the Voronoi tessellation performs better than the Geohash tessellation (Fig. 6). The optimal density based partition in Voronoi cells appears to be the reason for this behavior. For high density cells, Geohash based models perform at least as well as Voronoi based models. The comparable performance along with the low computational complexity makes Geohash tessellation the preferred choice for data dense locations. The numerical results are summarized below:

1) Using the models mentioned in Section IV-A, we achieve an average accuracy of about 80% per km² for both data sets, at 60 minute aggregation levels.
2) From Table II, we notice the absence of an universal winner on evaluating the performances of Geohash and Voronoi tessellation based models on 740 Bengaluru and 780 New York city demand centers.
3) On average, Voronoi tessellation appears to outperform Geohash tessellation for Bengaluru. On the other hand, Geohash tessellation emerges as the preferred technique for most of the tested cases for New York city.
models tend to produce better forecasts compared to Geohash based models. Note that even within a single city, a unique tessellation technique does not have better performance over the other at all time horizons. This time varying behavior of the models prompts us to combine the models for better location-based forecasts, which we do in the next section.

V. MODIFIED HEDGE FOR COMBINING MODELS

Consider a learning framework, where models provide recommendations. Each model is referred to as an expert. Assuming that the predictor has complete knowledge about all the past decisions and performances of the experts, the goal is to perform as well as the best expert in the pool. This exercise belongs to the class of ensemble learning where we make use of multiple experts to obtain better predictive performance than could be obtained from any of the constituent experts alone. In our context, whenever the Geohash tessellation performs better than the Voronoi tessellation at a time step \( t \), the ideal decision would be to choose the Geohash technique as the preferred tessellation strategy for that time step. For all demand centers, the forecasts at that instant should then be made from their geohash cells.

On observing the performances of both strategies at 15 and 60 minute aggregation levels, we see that a single tessellation technique is unable to yield optimal results for the entire forecast horizon. The best strategy varies with time. A plausible approach towards addressing this issue is to combine the two experts. As an initial step towards that goal, we apply the classic HEDGE algorithm as proposed in [28] to combine the models (Fig. 8b). We see that the decision maker is able to follow the best expert, the model based on Voronoi in this case, till a cross over occurs. The algorithm is unable to adapt to the shifting nature of the experts. In order to adapt to such a non-stationary environment, the dHEDGE [29] was proposed, where the authors modify the conventional HEDGE to include an exponential discounting factor. By exponentially filtering the past performances of the experts, the dHEDGE takes into account the non-stationarity of the process. By normalizing the weights of each expert at every instant, the dHEDGE provides a set of coefficients for experts. This can then be used to generate a convex combination of expert opinions. Our problem does not require normalized weights. Since there are only two experts, at every instant we can pick the expert which has the maximum weight, i.e., the expert with the minimum loss. We modify the dHEDGE algorithm to pick an expert if it has a higher weight than the other expert. The original algorithm has a generic loss function \( L(x, y) \in [0, 1] \). For illustrative purposes, the authors in [29] have used a discrete loss function \( L(x, y) = [x \neq y] \), where \([·]\) is the indicator function, \( x \) is the observed symbol and \( y \) is the predicted symbol. For their cellular LTE network application, that loss function outputs 0 for all the experts who predicted the symbol correctly at that instant, and output 1 for all other experts. On the other hand, we build our loss functions based on the errors.
observed for each strategy. Our modified dHEDGE algorithm is given in Algorithm 1.

**Algorithm 1:** Modified dHEDGE algorithm for selecting the best expert

1. **Parameters:** Learning parameter $\beta \in [0, 1]$.
   Discounting factor $\gamma \in [0, 1]$.
2. **Initialization:** Set $w_k[1] = W > 0$ for $k \in 1, 2$ s.t. $\sum_{k=1}^{2} w_k[1] = 1$.
3. for $t = 1, 2, \ldots$ do
   4. Choose expert with index $i = \arg\max_i (w_i[t])$.
   5. Error $e_v = \text{MEAN}$ (forecast errors from models based on Voronoi at $t$).
   6. Error $e_g = \text{MEAN}$ (forecast errors from models based on Geohash at $t$).
   7. Loss $L_t = \frac{e_v}{e_v + e_g}$, $L_{\gamma}[t] = \frac{\gamma e_g}{e_v + e_g}$.
   8. Update weights as $w_i[t+1] = w_i[t] \gamma \cdot L_{\gamma}[t]$.

The weights can be initialized either uniformly or based on some prior knowledge about the experts. The factor $\gamma$ gives more leverage to the observations made in the recent past compared to the observations in the distant past. Thus, the expert which have been performing well in the recent past is boosted. By tuning the parameter $\gamma$, we can get the dHEDGE algorithm to respond to sudden changes in the average error performance. By setting $\gamma = 1$, this algorithm becomes the HEDGE algorithm with no forgetting factor. When the modified dHEDGE is used, we observe that the algorithm tends to choose the best expert quite rapidly compared to the original HEDGE (Fig. 8c).

The modified dHEDGE algorithm was run on the two data sets at 15 and 60 minute sampling periods. The parameters $\beta$ and $\gamma$ for each scenario were chosen based on an independent validation set spanning over 24 hours. The performance of the algorithm can be seen in the Fig. 9. We can see that the algorithm picks up the best shifting expert, by giving more weightage to the behavior of that expert in the recent past (decided by $\gamma$). Note that in Fig. 9a, the plots of Voronoi and dHEDGE overlap as Voronoi at that temporal and spatial resolution. Note that our policy does not switch continuously between strategies every time step. This is because the dHEDGE chooses the winner strategy for a time step based on neither the current nor the immediate past performances of the experts. The dHEDGE makes a switch only if one strategy performs better than the other for a period of time in the past, which is decided by the discounting factor $\gamma$. On tracking the performance of our strategy for all the test cases mentioned in Table III, we observe an average of only 1.5 switches per day at 60 minute aggregation levels. In other words, for 24 time steps, the hybrid strategy made less than 2 switches on average for all tested scenarios. For 30 minute (i.e., 48 steps) and 15 minute (i.e., 96 steps) aggregation levels, the average switches made were only 3.2 and 8.4 per day. With finer temporal resolutions, the number of switches increases, which is inevitable, considering the increase in the number of time steps in the forecast horizon. We feel that the observed number of switches is acceptable, in return for a significant improvement in accuracy.

Based on our observations from two independent data sets, we find that there is no universal winning tessellation technique that works for all data sets. However, an algorithm that combines different tessellation strategies has the potential to enhance prediction performance across a broad range of data sets.

### VI. Conclusions

Efficient spatial partitioning is a key step towards better location-based demand modeling and forecasting. We performed a comparison of two widely used spatial partitioning techniques, i.e., Voronoi tessellation and Geohashing. Our study was conducted using two distinct data sets: (i) a mobile application based taxi demand data set from Bengaluru, India and (ii) a street hailing yellow taxi service data set from New York city, USA. A K-Means clustering algorithm was employed to generate demand clusters, that acted as generating sites for the tessellations. In order to compare the Voronoi and Geohash based models at multiple time scales, time-series techniques were applied to the spatially partitioned data at 15 and 60 minute temporal aggregation levels. The shortlisted models, that were found to be suitable, were ARIMA, ETS, Naïve, Random drift which were decomposed using STL, and TBATS. We noted that neither Geohash nor the Voronoi tessellation technique proved to be optimal for the entire forecast horizon. Further, models based on Voronoi had superior performance over models based on Geohash when the demand density was low, and vice versa. While Voronoi tessellation appeared to be the recommended strategy for tessellating Bengaluru, Geohash
Fig. 9: The prediction performance of the dHEDGE ($\beta, \gamma$) algorithm based model against single strategy based models at 15 and 60 minute aggregation levels for the two cities with SMAPE and MASE as the metrics are plotted in (a)-(d). In (e), the first two subplots correspond to the dHEDGE performance on Manhattan-Bronx boroughs, and the last two subplots correspond to the dHEDGE performance on Brooklyn-Queens boroughs respectively, at 15 minute aggregation levels.
TABLE III: The table contains the performance comparison of Voronoi, GeoHash and dHEDGE using SMAPE, for the entire set of simulations performed. We observe that the dHEDGE has a superior performance for all the tested combinations on all 4 data sets. The unique set of parameters ($\beta, \gamma$) obtained for each test case are mentioned below the errors encountered using dHEDGE algorithm.

tessellation was the suggested strategy for New York city. We concluded that the tessellation strategy is very dependent on the demand density in each partition, and the geography of the city.

The lack of a clear winning strategy prompted us to devise a hybrid tessellation strategy by combining models based on the well known HEDGE algorithm. We modified dHEDGE, which is a discounted version of the HEDGE algorithm, to suit our context. Our modified algorithm always picked up the best possible strategy at each time step in the forecast horizon. It is appealing that the tessellation performance is not impacted by the time of the day, or the features of the underlying data set. The hybrid strategy was the winning strategy for all the data sets considered, performing consistently better at multiple time scales with different performance metrics.

This paper was directed towards a comparison of tessellation strategies, where we focused on combining strategies to enhance performance. There are numerous avenues that merit further investigation. For improving the prediction accuracy, we can explore techniques such as Support Vector Regression and Artificial Neural Networks. In fact, recent developments in Recurrent Neural Networks have also shown promise in improving predictions [8]. It will certainly be interesting to consider the supply side of the data as well, to build a more comprehensive demand-supply modelling framework.

ACKNOWLEDGEMENTS

The work is undertaken as part of the Cognizant-Ernet IIT Madras project. The authors are also thankful to Vishnu Raj, Gopal Krishna Kamath, Saarthak Jain, and Venkata Siva Sai Prasad for many helpful discussions.

REFERENCES


[31] Neema Davis is currently pursuing her Dual M.S.-Ph.D. degree in Electrical Engineering from the Indian Institute of Technology Madras. She obtained her B. Tech. in Electronics and Communication Engineering from the Mahatma Gandhi University Kottayam in 2013. Her research interests lie in transportation planning, predictive data analysis, and intelligent transportation systems.

[32] Gaurav Raina holds a faculty position in Electrical Engineering at the Indian Institute of Technology Madras, and is also a visiting research fellow in Mathematics at the University of Cambridge, UK. His research interests are in the design and performance evaluation of communication networks, intelligent transportation systems, control theory, and non-linear systems.

[33] Krishna Jagannathan obtained his B. Tech. in Electrical Engineering at IIT Madras in 2004, and the S.M. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT) in 2006 and 2010 respectively. During 2010-2011, he was a visiting post-doctoral scholar in Computing and Mathematical Sciences at Caltech, and an off-campus post-doctoral fellow at MIT. Since November 2011, he has been an assistant professor in the Department of Electrical Engineering, IIT Madras. He worked as a consultant at the Mathematical Sciences Research Center, Bell Laboratories, Murray Hill, NJ in 2005, and as an engineering intern at Qualcomm, Campbell, CA in 2007. His research interests lie in the stochastic modeling and analysis of communication networks, transportation networks, network control, and queuing theory. Dr. Jagannathan serves on the editorial boards of the journals IEEE/ACM Transactions on Networking and Performance Evaluation. He is the recipient of a best paper award at WiOpt 2013, and the Young Faculty Recognition Award for excellence in Teaching and Research at IIT Madras (2014).