A Framework for Continuous kNN Ranking of EV Chargers with Estimated Components

Soteris Constantinou*, Constantinos Costa◇*, Andreas Konstantinidis•*,
Mohamed F. Mokbel‡, and Demetrios Zeinalipour-Yazti*

* Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
◇ Rinnoco Ltd., 3047 Limassol, Cyprus
• Department of Computer Science & Engineering, Frederick University, 1036 Nicosia, Cyprus
‡ Department of Computer Science and Engineering, University of Minnesota, MN 55455 Minneapolis, USA
sconst01@ucy.ac.cy; costa.c@rinnoco.com; com.ca@frederick.ac.cy; mokbel@umn.edu; dzeina@ucy.ac.cy

Abstract*—In this paper, we present an innovative framework whose objective is to allow drivers to recharge their *Electric Vehicles (EVs)* from the most environmentally friendly chargers using an intelligent hoarding approach. These chargers maximize renewable (e.g., solar) self-consumption, minimizing this way CO_2 production and also the need for expensive stationary batteries on the electricity grid to store renewable energy that cannot be used otherwise. We model our problem as a *Continuous k-Nearest Neighbor query*, where the distance function is computed using *Estimated Components (ECs)*, i.e., a query we term *CkNN-EC*. An EC defines a function that can have a *fuzzy value* based on some estimates. Specific *ECs* used in this work are: (i) the *(available clean) power at the charger*, which depends on the estimated weather; (ii) the *charger availability*, which depends on the estimated busy timetables that show when the charger is crowded; and (iii) the *derouting cost*, which is the time to reach the charger depending on estimated traffic. We devise the EcoCharge framework that combines these multiple non-conflicting objectives into an optimization task providing user-defined ranking means through an intuitive mobile GIS application. Particularly, our core algorithm uses lower and upper values derived from the ECs to recommend the top ranked EV chargers and present them through an intuitive map user interface to users. Our experimental evaluation with extensive synthetic and real traces from Germany, China, and USA along with EV charger data from Plugshare shows that *EcoCharge* meets the objective functions in an efficient manner, allowing continuous recomputation on the edge devices (e.g., Android Automotive OS, Android Auto or Apple Carplay).

***Keywords*-Mobile Data Management, Green Mobility, Renewable Self-Consumption, Electric Vehicles, Charging.**

I. Introduction

The market penetration of electric vehicles (EVs) has witnessed an exponential growth in recent years, owing to their exceptional advantages over conventional internal combustion vehicles in terms of sustainable transportation and cost-effectiveness. Cities play a pivotal role in achieving climate neutrality by 2050, the goal of the European Green Deal[1], as they are accountable for over 65% of the world's energy consumption and 70% of global CO_2 emissions. In recent years, there has been an increasing interest in the integration of *Renewable Energy Sources (RES)* with *Electric Vehicle (EV)* charging infrastructure [1], [2] (i.e., photovoltaic panels, wind turbines). People frequently tend to charge their EVs during idle times, even when the battery is not substantially depleted, to ensure that the vehicle will be charged sufficiently when required for travel; a habit we term *energy hoarding*. EVs are seen as a way to improve the environment and reduce greenhouse gas emissions. Energy hoarding with non-renewable energy is negating environmental benefits. In the U.S., the EV energy charging demand was attributed to 4.7 terawatt hours in 2020, and is expected to increase to ≈107 terawatt hours by 2035[2].

The research based on various approaches to optimize the charging of EVs utilizing RES, is a prominent research area at present time [3]. Current applications (e.g., Plugshare, Ionity, Tesla's supercharger, EnBW, Shell recharge) focus on allowing users know where to recharge but do not list the environmental impact of the charging process (i.e., energy might come from fossil fuel burning). Different optimization techniques such as smart charging [4], and vehicle-to-grid (V2G) systems [5] are being explored to improve the environmental impact of the charging process and minimize the strain on the power grid. Additionally, regenerative energy methods are applied in loaded electric trucks during their downhill trip through energy recuperation, thus, creating enough energy to get the empty truck uphill again [6]. To the best of our knowledge, there is no other work focusing on *sustainable EV energy hoarding*.

A renewable hoarding technique can be applicable in scenarios with *idle time* (i.e., while an EV user is waiting or parked). For example, consider the following real-life scenarios: (i) electric taxis (e.g., Lyft, Uber, Bolt) during idle periods are waiting to be called or booked online; (ii) parents waiting in their idle EVs while their children attend after-school activities; and (iii) an EV user going for groceries or clothing

‡The work of this author is partially supported by NSF Grant IIS-2203553.

[1]EU Climate-Neutral Smart Cities, https://tinyurl.com/57tzjmyk

[2]Statista-EV charging demand, https://tinyurl.com/mtc6w8nt

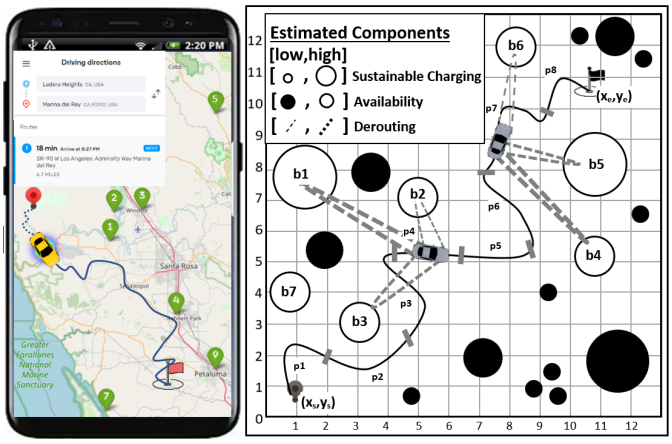

Fig. 1. EcoCharge application: An *Offering Table (O)* example of a moving vehicle based on a scheduled trip P. The ranking selection is derived from each EV charger's rate and solar production curve at a certain time, considering also the estimated time of arrival (ETA), and the return trip time.

shopping. This can also be applied in other similar situations where EV users are engaged in activities that require idle time, such as attending meetings, events, or conferences. Consequently, in all aforementioned scenarios, users could stop or deroute at some nearby charging station to efficiently charge their EVs using power generated from renewable sources, thus, reduce the carbon footprint of their daily routine.

One technical challenge, is that the decision of where to sustainably hoard depends on a variety of Estimated Components (ECs) on where and when to charge. Examples of these estimations are: the *derouting cost* to reach the charger that depends on estimated traffic, the *(available clean) power at the charger* that depends on the estimated weather, and the *charger availability* that depends on the estimated busy timetables showing when the charger is crowded. In such a dynamic context, traditional indexing approaches may not be the most suitable solution, given the continuous changes [7].

To solve this problem, a *Continuous k-Nearest Neighbor (CkNN) query* [8] can be utilized to answer questions like which EV chargers are closer to a path P. CkNN is a query that can retrieve the k nearest neighbors of a given path, however, it does not consider the estimation of various components. Our work falls under the concept of renewable hoarding techniques exploiting ECs. The objective is to optimize EV charging by utilizing only RES and focusing solely on short-term traveling (i.e., urban setting), ignoring multi-stop planning (e.g., VW Multi-Stop EV Route Planner or abetterrouteplanner.com).

We model the problem as a new CkNN-EC query that retrieves the k nearest neighbors of every point on a path segment (e.g., "find all my nearest EV chargers during my route from source to end point."), while considering ECs by employing a distance function that can express a fuzzy value.

Let us assume B as a dataset of EV charging stations on a road network, where a *CkNN-EC* query retrieves the nearest neighbor (e.g., k=1) of every path segment p of path P. Particularly, the result is a set of $<b, p>$, where b is an EV charging point of B. As a simplified example scenario, consider Figure 1, where B=$\{b_1,...,b_{20}\}$. Therefore, considering the path P and emphasizing on moving at the direction of the final destination, the 1*NN* result of the query is $\{< b_7, p_1 >, < b_3, p_2 >, < b_3, p_3 >, < b_1, p_4 >, < b_4, p_5 >, < b_5, p_6 >, < b_5, p_7 >, < b_5, p_8 > \}$, meaning that charger b_7 is the nearest neighbor for the segment of path p_1. The proposed approach utilizes a prediction model with respect to the weather forecast, derouting cost, and charger's availability. The points within the path segment (i.e., $(x_s, y_s), ..., (x_e, y_e)$) at which a transition in neighborhood occurs are referred to as split points SL [8].

In this work, we present an innovative renewable hoarding application for charging EVs, dubbed *EcoCharge*. Our approach utilizes a *CkNN-EC* search and a dynamic caching technique to generate *Offering Tables* with sustainable chargers for urban traveling, in a reasonable response time. The ranking is derived from the cost function, which employs an iterative deepening process to determine the kNN sets with EV chargers from the query point within a predefined time window, while considering ECs, expressed through intervals (i.e., ranges of min-max values). *EcoCharge* can be used over three different Modes: (i) Mode 1, operating in a vehicle's embedded operating system (e.g., Android Automotive OS, Volkswagen OS3); (ii) Mode 2, where calculations are conducted centrally on a server; and (iii) Mode 3, where functionalities are managed by an edge device (e.g., smart phone using Android Auto or Apple CarPlay).

Our solution enables a user to find a CO_2-neutral plan in a road network for charging during an idle period or a short-time trip. The efficiency of the proposed method is measured by: (i) the *Sustainable Charging Level (L)*; (ii) the *Availability (A)*; (iii) the *Derouting Cost (D)*; and (iv) *CPU Execution Time (F_t)*. Our goal is to make a significant contribution to the niche market of environmentally-conscious EV charging, particularly in alignment with the European Union's environmental targets for 2030 [9].

In summary, in this paper we make the following contributions:

- We propose a *Continuous k-Nearest Neighbor search with Estimated Components (CkNN-EC)* to efficiently find the nearest EV chargers of a moving object at each timestamp based on a scheduled trip, while considering ECs.
- We present *EcoCharge*, an innovative renewable hoarding application for EV charging in urban settings, integrated with *CkNN-EC* and a dynamic caching technique to minimize *derouting* traveling and maximize sustainable charging level.
- We evaluate our algorithm through an extensive experimental series on real and synthetic datasets of road network vehicle trajectories (i.e., Oldenburg, California, T-drive, and Geolife) utilizing measurements from EV chargers and weather forecast, showing that *EcoCharge* can be premise for green mobility in the future.

The remainder of the article is organized as follows: Section II presents the system model and formulates the problem, where Section III describes the EcoCharge Framework while Section IV discusses its internal architecture and implementation. Our experimental methodology is presented in Section V, the related work in Section VI, and the article is concluded in Section VII.

II. System Model & Problem Formulation

In this section, we formalize our system model, problem formulation, and the basic terminology used throughout this manuscript. The main symbols and their respective definitions are summarized in Table I.

TABLE I
NOTATION USED THROUGHOUT THIS WORK

Notation	Description
b, B	EV charger b, Set of all b
m, M	Electric vehicle m, Set of all m
p, P	p is a path segment of P, P is a scheduled trip
G	Road network (consists of V nodes and E edges)
O	Offering Table of the top ranked EV chargers
L	Sustainable charging level
A	Charger's availability
D	Derouting cost (distance from m to b)
t	Time (e.g., ETA)
s	Solar photovoltaic power generation
F_t	CPU execution time

A. System Model

We consider a road network as a directed weighted graph $G = (V, E)$, where V is a set of nodes (vertices) and E is a set of edges. Each node v has a spatial coordinate ($v.x$, $v.y$) defining its longitude and latitude. On every single edge $(u, v) \in E (u, v \in V)$, a weight is assigned $w(u, v)$, representing the energy cost to travel from point u to v. One can consider the travel cost as the length of the road segment, the time required to pass the road segment, or other costs like energy consumption or CO_2 emissions. Each vehicle moving on the road network is indicated by m and has certain geographic coordinates, namely loc_m.

We consider a set of public charging points B linked to nearby *Renewable Energy Sources (RES)* in an urban setting. We also assume numerous electric vehicles moving arbitrarily on the road network G. We assume that each vehicle m is equipped with a charging mobility app, such as *EcoCharge*, which can recommend charging points for every path segment p of a scheduled trip P.

We denote the *availability of clean energy* (measured in kWh) at the charger $b \in B$, over a user-specified time window t, as L. The value of t is derived from the *Estimated Time of Arrival (ETA)* the user receives through a cooperating navigation app (e.g., Google Maps or Waze). Without loss of generality, we assume that clean energy might come from either local sources (e.g., locally attached solar panels on carports) or virtually net-metered/net-billed from a remote renewable energy production farm. Given that clean energy might fluctuate based on the weather conditions, L is a prediction, or *Estimated Component (EC)*, as we term it in this work. We denote the *physical availability* of a charger at time point t, as A. Again, given that A is based on how crowded the charger is at different times of a day, A is also an EC. We denote the energy required for vehicle m to reach a prospective charger b as the *derouting energy cost* (measured in kWh), as D. Given that traffic on the road network might fluctuate (e.g., based on time and day), D is also an EC.

The EcoCharge app displays at all times while m is on the move, an *Offering Table O* (e.g., every few minutes) that is computed either in the cloud or on the edge (i.e., on vehicle m). We will outline the architectural approaches to where EcoCharge executes in Section IV. In cases where the range distance from previous to current EV's location is within a configured parameter, then a re-generation of an O is avoided, and an adaptation of a previously generated solution occurs.

B. Problem Formulation

The goal of this work, is to efficiently compute and provide to all EV drivers on a road network an *Offering Table O* with sustainable chargers over a large search space based on their *short-range* trip (i.e., urban driving). The notion of O is based on the minimization of EV-to-charger derouting travel distance, while simultaneously considering availability and green charging by adopting self-consumption of RES based on *ETA*.

The intention of the EcoCharge app is to optimize an objective function to achieve a trade-off between the vehicle's sustainable charging level L, the chargers' availability A, and the derouting travel distance D to a charger.

III. The EcoCharge Framework

In this section, we outline the functionality of EcoCharge, providing a comprehensive explanation of its internal phases, which is subsequently followed by an illustrative example of its operation.

A. Outline of Operation

EcoCharge employs a *Continuous k-Nearest Neighbor search with Estimated Components (CkNN-EC)* query to ascertain the collection of nearest neighbors (i.e., nearest EV chargers) for all future path segments of P. Particularly, we can think about this query generating concurrently multiple kNN results, i.e., one for each segment of a trip P. The results to the kNN computation effectively yield what we consider as an *Offering Table O*, which comprises of multiple sub-results $O_{p1}, ..., O_{pn}$. To avoid numerous database scans, we pass the entire pool of segments once (along with their corresponding coverage). Particularly, our method begins with an initial SL composed of only two split points (x_s, y_s) and (x_e, y_e) with their associated covering points set to 0 (indicating that nearest neighbors of all points within the segment p_i are currently unknown). At every step, SL comprises the current result according to all processed points so far. For each split point $(x_i, y_i) \in SL (0 \leq i < |SL| - 1) : (x_i, y_i) \in p_i$ and all points

in p_i have the same nearest neighbors, expressed as NN_{p_i}. The final output comprises every split point (x_i, y_i) that remains in SL after the termination, along with its nearest neighbors NN_{p_i}.

Our framework processes each vehicle m on the road network at any given time based on the following steps:

- **Step 1:** Gets user's scheduled trip P on a road network G and partitions it in segments of $p \approx$3-5km or respective time window. This setting can be modified according to the user's needs and preferences.
- **Step 2:** Searches for closest b considering the L, A, D objectives and calculating a unified *Sustainable Charging (SC)* score.
- **Step 3:** Displays to the driver an *Offering Table (O)* to decide and select an EV charging location from the collected available options.

B. EcoCharge Objectives

The efficiency of the proposed technique is measured by the following metrics: (i) *Sustainable Charging* (L) level; (ii) *Availability* (A) score; and *Derouting* (D) cost.

Sustainable Charging Level (L): Each EV charging station b has a different charging rate and power generation levels s_t depending on time and location. In this work, we do not consider energy imported from the grid, but only solar excess produced, as we aim to generate zero CO_2 emissions during the EV charging phase (i.e., renewable hoarding). Further, the *sustainable charging level* considers the weather forecast (e.g., sunny, cloudy) at a given time and location retrieved by a cloud service (e.g., OpenWeather, Windy, WindFinder), which utilizes weather models like Global Forecast System (GFS) [10] and European Centre for Medium-Range Weather Forecasts (ECMWF) [11], both with accuracy of 95–96% for up to 12 hours and 85–95% for three days. L consists of a lower and upper estimation values, thus, the final result is an interval L_{min} to L_{max}. We normalize these values by dividing them with the environment's maximum charging level value.

$$L(B) = max\{s_t^b \mid \forall\ b \in B\ \} \tag{1}$$

Availability (A): Each EV charger's *availability* is estimated using some third-party service (e.g., Google Maps POI busy timetables), as shown in Figure 2, enabling the determination of real-time accessibility on a given time t. Therefore, an interval is produced A_{min} to A_{max}. We express these values in percentages, with 0% being not busy and 100% being busy.

$$A(B) = max\{A_b \mid \forall\ b \in B\ \} \tag{2}$$

Derouting Cost (D): A route path from starting point v_0 to target charger v_k is a sequence of nodes $P =< v_0, v_1, ..., v_k >$, where w represents the edge weight in terms of CO_2 emissions (e.g., or kWh consumed). In case $derouting$ is needed to arrive at a selected charger outside the initial scheduled trip, CO_2 emissions will be added to the total calculated travel cost respectively. In case the charger is located on the path of the scheduled trip, no $derouting$ occurs, thus no extra CO_2 emissions will be added. Further, the $derouting$ accurately considers real-time traffic information (e.g., congestion) at a given time and location retrieved from a cloud Geographic Information System (GIS) service (e.g., Google Maps, Waze, HERE Maps), thus, D consists of a lower and upper estimation values. Therefore, the final result is an interval D_{min} to D_{max}. We normalize these values by dividing them with the environment's maximum derouting distance. The minimum derouting cost for a segment p of the path P to all chargers B is:

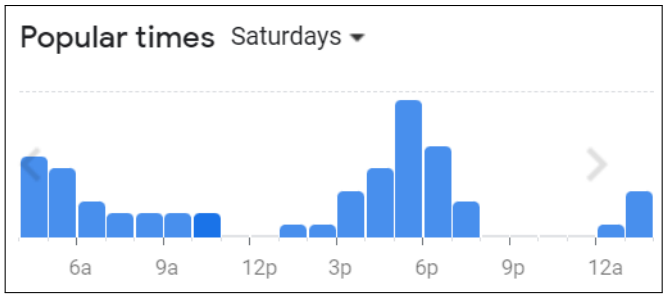

Fig. 2. Popular times of an EV charger's availability estimation example through Google Maps.

$$D(B) = min\{\sum_{i=0}^{|p|} w_{v_i,b} * distance(v_i, b) \mid v_i \in\ p,\ \forall\ b \in B\} \tag{3}$$

The equation ensures the minimization of D and consequently the reduction of CO_2 emissions since they are correlated.

Sustainability Score (SC): In this paper, we evaluate *SC* as a weighted sum function, where w_1 the weight of *Sustainable Charging Level* (L) objective, w_2 the weight of *Availability* (A) objective, and w_3 the weight of the *Derouting Cost* (D) objective, respectively. These weights are user-configurable to tailor the decision-making process according to users' individual preferences and priorities. Using $CkNN$ with SC as the distance function, *EcoCharge* produces two result-sets, one based on SC_{min} and another on SC_{max}, and the final output is their intersection, which contains k chargers.

$$SC_{min} = (L_{min} * w_1) + (A_{min} * w_2) + ((1 - D_{min}) * w_3) \tag{4}$$

$$SC_{max} = (L_{max} * w_1) + (A_{max} * w_2) + ((1 - D_{max}) * w_3) \tag{5}$$

$$SC(B) = sort(SC_{max}(b) \cap SC_{min}(b)), \forall b \in B \tag{6}$$

Additionally, the proposed approach is also discussed with respect to *CPU Execution Time (F_t),* which is the processing time required by the algorithm for running the weighted sum optimization function and generating the output.

C. The EcoCharge Algorithm

In this section, we present the algorithmic solution behind the EcoCharge Framework, coined EcoCharge, presented in Algorithm 1.

Algorithm 1 *EcoCharge*: An Energy Hoarding Algorithm for Sustainably Charging Electric Vehicles (EVs)

Input: B: set of all EV chargers; P: scheduled trip; m: EV's information; s: solar energy production data;
Output: A generated Offering Table O, consisted of sustainable EV chargers (i.e., CkNN-EC of B)

1: **EcoCharge**(B, P, m, s) ▷ **Renewable Hoarding Algorithm for Sustainable Charging**
2: $p \leftarrow tripSegmentation(P)$ ▷ divides scheduled trip into path segments
3: **for each** $(b \in B)$ ▷ iterates through all EV chargers
4: $ETA \leftarrow estimatedTimeArrival(loc_b, loc_m)$
5: $L_{min(b)} \leftarrow sustainableMinChargingLevel(b, s, ETA)$ ▷ Sustainable Charging Level
6: $L_{max(b)} \leftarrow sustainableMaxChargingLevel(b, s, ETA)$
7: $A_{min(b)} \leftarrow availabilityMin(loc_b, ETA)$ ▷ Availability
8: $A_{max(b)} \leftarrow availabilityMax(loc_b, ETA)$
9: $D_{min(b)} \leftarrow deroutingMinCost(loc_b, loc_m, p)$ ▷ Derouting
10: $D_{max(b)} \leftarrow deroutingMaxCost(loc_b, loc_m, p)$
11: $SC_{min(b)} \leftarrow CkNN\_EC\_Min(L_{min(b)}, A_{min(b)}, D_{min(b)})$ ▷ lower estimation set
12: $SC_{max(b)} \leftarrow CkNN\_EC\_Max(L_{max(b)}, A_{max(b)}, D_{max(b)})$ ▷ upper estimation set
13: $SC^*_{min(b)} \leftarrow SC^*_{min(b)} + [SC_{min(b)}]$ ▷ stores lower estimation Sustainability Score
14: $SC^*_{max(b)} \leftarrow SC^*_{max(b)} + [SC_{max(b)}]$ ▷ stores upper estimation Sustainability Score
15: **end for each**
16: $SC \leftarrow SC^*_{max(b)} \cap SC^*_{min(b)}$ ▷ finds the common chargers between SC_{min} and SC_{max}
17: $O \leftarrow sort(SC)$ ▷ sorts the SCs (highest to lowest rank b optimizing the SC score)
18: **return** (O) ▷ returns a generated Offering Table

In order to continuously monitor the result of kNN, the *CkNN-EC* method necessitates partitioning the entire route distance into separate segments (see line 2), which are sequentially considered for the kNNs determination of the query object. The partitioning procedure is responsible to divide the scheduled trip P into segments (e.g., ≈5km each segment; can be modified in settings as per preference). It is essential to note that the road network distances between all chargers and the query object (i.e., EV vehicle) have to be updated every time the query object reaches a segment intersection SL of the scheduled trip. For each segment p_i, the process of finding the kNNs is composed of two phases. The first one is called *Filtering phase*, which is used to discard non-qualifying chargers. The second phase is called *Refinement*, where an evaluation is conducted to determine the eligibility of candidate chargers as *CkNN-EC* utilizing the equation 6.

Filtering Phase: According to the driver's location (e.g., segment p_i), the *Filtering* process ensures that only the k most suitable chargers are considered, while pruning all the rest. The particular phase loops through the entire pool of EV chargers and examines each one based on several estimation components. The first component is the sustainable charging level (L), which considers charger's rate and the power generation at a given time (lines 5-6). Since the weather conditions might change at any time (e.g., sunny clear sky becomes cloudy in 10 minutes), upper and lower estimations are also considered. The second component focuses on the charger's availability (A) based on ETA. This means that our approach analyzes when EV users are expected to arrive at a station, allowing for the estimation of potential peak usage times and ensuring that chargers are accessible. The second component is similarly to the previous case, an interval that is generated with upper and lower availability values (see lines 7-8). The last component is the derouting cost (D), which calculates the distance and time required to reach a station and to return back to the scheduled trip. The returning back phase could either mean going back to the same segment p_i or going to the next one (i.e., p_{i+1}), thus, whichever has the less derouting is selected (or can be a user preference). Lower and upper estimation values are taken into account due to the varying traffic conditions and unexpected situations (lines 9-10).

Refinement Phase: In this phase, each charging station selected in the pool of filtered candidates undergoes another assessment to evaluate suitability. The assessment is conducted by applying equation 6, which utilizes an intersection in the minimum and maximum interval values of Sustainability Score SC (see line 16). Thereafter, a sustainability function sorts results (see line 17) to ensure that only the most relevant and efficient charging stations are identified, contributing to an optimized renewable hoarding strategy for the user.

Example: Consider the scenario illustrated at the right side of Figure 3, where a vehicle m is on a scheduled trip P (i.e., consisted of several path segments $p_i \in P$) and there are 20 EV charging stations $b_1, .., b_{20}$. Let us assume that the weather forecast for the 1-hour trip (i.e., 10:00am-11:00am) is sunny, which means the EV chargers produce sustainable energy. The user (with an 11kW AC charger car) wishes at timestamp 10:15am to visit a station to charge his/her EV. The decision on where to charge is made based on the user's current location, sustainable charging level L at b based on weather conditions, physical availability A of b, and derouting cost D to b. For simplicity we do not consider any weights in this example. Considering the nearest chargers with respect to vehicle's location at 10:15am and path segment p_4, we

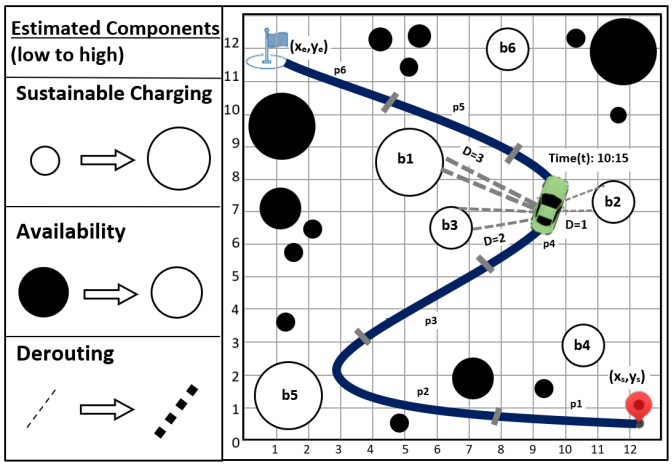

Fig. 3. Example of a Continuous k-Nearest Neighbor query with Estimated Components (CkNN-EC). An illustration of a selected EV charger in a specific path segment (p) of the scheduled trip (P).

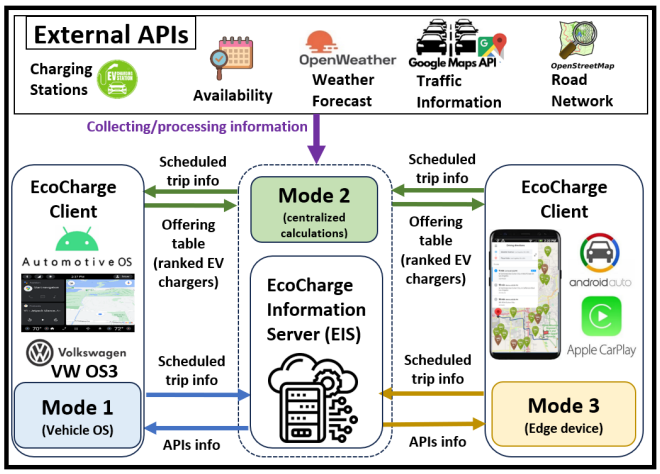

Fig. 4. **EcoCharge Architecture:** the server takes as an input all available EV chargers, weather forecast, availability, traffic data, and road network information. The collected data are provided to the client through three discrete modes of operations.

have several options: (i) Charger b_1 is available and derouting costs D_{b_1} = -2kWh, while sustainable charging gains L_{b_1} = +9kWh, thus SC_{b_1} = +7kWh; (ii) Charger b_2 is available and derouting costs D_{b_2} = -1kWh, while sustainable charging gains L_{b_2} = +4kWh, thus SC_{b_2} = +3kWh; and (iii) Charger b_3 is available and derouting costs D_{b_3} = -2kWh, while sustainable charging gains L_{b_3} = +4kWh, thus SC_{b_3} = +2kWh. Therefore, the ranking for the path segment is as follows $< b_1, b_2, b_3 >$ (highest to lowest).

IV. The EcoCharge Architecture

In this section, we describe the comprehensive architecture of our framework. The core of our system resides in an *EcoCharge Client* supported by a centralized server, which interacts with external APIs to retrieve essential data (see Figure 4). Leveraging external APIs, our *EcoCharge Information Server (EIS)* acquires real-time weather forecast data, detailed road network information, and a comprehensive list of all available EV charging stations based on user's location. This centralized approach allows the server to efficiently consolidate the required data and distribute to individual clients as per request. Our framework mitigates the need for redundant API call requests by intelligently employing a smart caching mechanism, called *Dynamic Caching* presented in this section.

The service can be provided to the users with three modes of operation: (i) *Mode 1*, where EcoCharge operates in a vehicle's embedded operating system (e.g., Automotive OS, Volkswagen OS3); (ii) *Mode 2*, where *EIS* takes over EcoCharge calculations centrally; and (iii) *Mode 3*, where EcoCharge functionalities are managed by an edge device (e.g., smart phone using Android Auto or Apple CarPlay).

A. EcoCharge Client

Upon receiving, through an API call, the weather forecast, road network details, and EV charging station information from the *EIS*, the client application takes on the pivotal role of processing this data. Tasked with the responsibility of route optimization, the client application employs a novel algorithmic approach (see Algorithm 1) to calculate the most efficient route considering sustainable charging and derouting cost based on the user's scheduled trip. This process involves dynamically identifying EV chargers along the route, considering factors such as real-time sunlight conditions, road network intricacies, and availability. Hence, users make informed decisions about their EV charging strategy while minimizing environmental impact (see Figure 5a).

EcoCharge Client continuously recomputes the path using a ≈3-5 minutes window, which could change the initial route to accommodate a visit to an offering charging station. This deviation is done with the objective of finding a more efficient overall route (i.e., current location to charger, and charger to destination), which includes the additional distance to and from an EV charging station.

B. EcoCharge Implementation

In this subsection, we describe the technologies utilized for the implementation of our proposed framework. *EcoCharge Client*, prototyped in Python 3, leverages the capabilities of the Folium[3] library - a robust tool designed for creating diverse Leaflet maps. The utilization of Folium is integral to our system's functionality, providing a dynamic and interactive mapping component. It empowers our system to generate visually compelling maps with various layers and features, enhancing the user experience and facilitating a comprehensive understanding of geographical data. Through the integration of Leaflet, HTML, and JavaScript, we ensure that our system not only delivers powerful functionality, but also presents information in a visually engaging and accessible manner. This strategic use of technology aligns with our commitment to providing users with an intuitive and effective platform for exploring geographical data.

[3]Folium, `https://python-visualization.github.io/folium/`

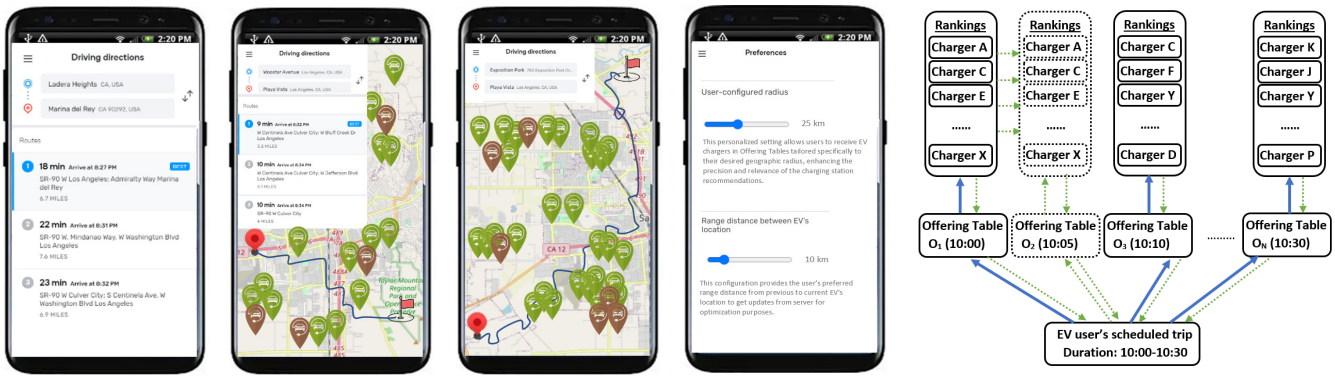

Fig. 5. **(a) EcoCharge Client GUI:** First four images show GUI preferences and route functionalities used by EcoCharge to generate an Offering Table with the most sustainable chargers identified; **(b) Dynamic Caching (bottom up strategy):** The graph on the right side is an example demonstrating the adaptation of an already generated Offering Table (i.e., O_1 to O_2) consisted of the higher SC chargers. O_N is the last Offering Table at user's final destination.

Our mobile-based application enhances user experience by integrating with the device's location services. Through an intuitive GUI, users can easily set their desired destination for a trip and receive comprehensive route information, leveraging the application's functionality for efficient navigation. EcoCharge acquires multiple input parameters critical for its processing. It gathers comprehensive data, including the locations of all available EV charging stations retrieved by PlugShare[4], real-time weather forecasts, and user preferences. To gauge environmental conditions such as sunlight availability, we rely on data from OpenWeatherMap[5], an API service offering up-to-the-minute weather information for diverse global locations.

Moreover, EIS is designed using the Laravel[6] PHP Model-View-Controller Framework, ensuring a robust and organized structure, and it is deployed with the high-performance Nginx[7] web server for efficient handling of HTTP requests. It efficiently retrieves road network information by integrating with OpenStreetMap[8], a robust and versatile mapping and location data platform. Leveraging OpenStreetMap allows us to tap into a wealth of detailed road network data, including information on streets, highways, and various transportation routes. This integration empowers our application to provide users with accurate and up-to-date maps, ensuring a comprehensive understanding of the road infrastructure.

C. Dynamic Caching

EcoCharge adapts a *dynamic caching* method by breaking the problem space down to smaller, simpler sub-problems and solving each one only once. It stores solutions (i.e., Offering Tables) and API responses in a table and uses them to solve similar problems until the overall problem is completely solved. The advantage of our applied technique, is the ability to avoid redundant computations and reduce the overall complexity of the problem by re-using already generated solutions on corresponding cases, hence, leading to significant speedup. A caching hypothesis can be made as the decision space exposes temporary behavior, namely a solution will naturally be invalidated after a certain time point (t) as L, A, D objectives will naturally be invalid after t. Specifically, we have utilized a *bottom-up* approach where sub-problems are solved separately and their results are stored in a table, which can then be used to solve future queries.

To clearly express the utilized *dynamic caching bottom-up* approach, consider an EV user's current location p_i in a road network with various charging stations B. *EcoCharge* will normally start calculating each charger's Sustainability Score SC_b (i.e., solving sub-problems) to form an Offering Table O_1 based on user's location (see Figure 5b). The O_1 consists of the best charging stations sorted, having the highest SC charger first. Then, following the driver's next location p_{i+1} according to a scheduled trip, a new problem will need to be tackled. Considering the distance between the last and current locations, an already calculated solution can be re-applied in the context of solving a larger problem. To address this, we have utilized radius R, which allows users to receive EV chargers within their desired geographic radius, and parameter Q, which indicates users' preferred distance from previous to current location for getting server updates and calculating new solutions. In this manner, our algorithm does not need to loop through the entire search space of chargers for every new path segment in case previous EV location's requirements match R and Q parameters. Hence, O_1 can be adjusted to O_2, in case parameter conditions are satisfied, and this carries on to the next EV path segments until the user's final destination.

In particular, before a new Offering Table is generated and provided to the user, *EcoCharge* examines the previous and current location in order to decide whether it needs to re-generate a new solution or the previously generated one can be applied. Consequently, our applied hoarding method minimizes CPU execution time by eliminating unnecessary computations over the large search space of chargers and enables CO_2-neutral EV charging.

[4]PlugShare-EV Charging Stations, `https://www.plugshare.com/`
[5]OpenWeatherMap, `https://openweathermap.org/`
[6]Laravel MVC, `https://laravel.com/`
[7]Nginx, `https://www.nginx.com/`
[8]OpenStreetMap: `https://www.openstreetmap.org/`

V. Experimental Methodology & Evaluation

This section presents an experimental evaluation of our proposed approach. We start-out with the experimental methodology and setup, followed by various experiments conducted that expose the core benefits of our *EcoCharge* framework.

A. Methodology

This section provides details regarding the algorithms, metrics and datasets used for evaluating the performance of our approach.

Testbed: Our evaluation is carried out on our laboratory VMware private datacenter. Our computing node comprises of a Ubuntu 22.04 server image, featuring 6GB of RAM with 4 virtual CPUs (@ 2.60GHz). The image uses fast local 10K RPM RAID-5 LSILogic SCSI disks, formatted with VMFS 6 (1MB block size).

Datasets: We have adopted a trace-driven experimental methodology in which real and synthetic datasets are fed into our simulator. Specifically, we utilize real road network trajectories in California, collected by Boston University's Computer Science department [12]. The second utilized trajectory dataset is a synthetic one based on Oldenburg's road network, and was generated with Brinkhoff spatio-temporal generator [13]. Additionally, we used two large GPS trajectory datasets, named Geolife and T-drive, collected by Microsoft Research Asia [14], [15]. The datasets of EV charging stations and their production information based on weather forecast were retrieved by PlugShare and "California Distributed Generation Statistics" (CDGS) [16].

- **Oldenburg [13]:** The synthetic dataset generated by Brinkhoff spatio-temporal tool, includes 4,000 vehicle trajectories in a 45km x 35km area of Oldenburg, Germany.
- **California [12]:** The real road network dataset consists of 7,000 vehicle trajectories (e.g., edges and nodes) in a 1,220 km x 400 km area of California, U.S.
- **T-drive [14]:** It contains the GPS trajectories of 10,357 taxis during the period of February 2 to February 8, 2008 within Beijing, China. The total number of points is about 15 million and the distance of the trajectories reaches 9 million km.
- **Geolife [15]:** This dataset contains 17,621 trajectories with a total distance of 1,292,951 km and a total duration of 50,176 hours. These trajectories were recorded by different GPS loggers and GPS phones, and have a variety of sampling rates. 91.5% of the trajectories are logged in a dense representation of every 1∼5 seconds or every 5∼10 meters per point.
- **EV Chargers & Solar Production [16]:** The utilized dataset consists of more than 1,000 chargers along with various information about their charging rates, timestamps, and solar generation in a 15-minute time-interval from 2016 to 2018.

Metrics: The efficiency of the proposed technique to achieve the research goal introduced earlier, is measured by the *Sustainable Charging Level (L)*, *Availability (A)*, *Derouting Cost (D)*, and *CPU Execution Time (F_t)*, as detailed described in Section II. For consistency and simplicity the Sustainability Score (SC) is computed in the first three experimental series using equal weights, particularly $w_1 = w_2 = w_3$ and $w_1 + w_2 + w_3 = 1$. The presented SC score is presented as a percentage of the Brute Force solution (with it scoring the optimal solution 100%). The mean and standard deviation of the results are shown with error bars in the experiments, based on approximately ten repetitions.

A concise overview of the compared baseline approaches optimizing the objectives follows below:

- **Brute-Force Method:** performs an exhaustive search over the entire pool of chargers to find the ones maximizing the SC.
- **Index-Quadtree Method:** uses a specialized tree data structure used for partitioning a two-dimensional space and improve the runtime performance of the Brute-Force method from $O(n)$ to $O(log_n)$, where n is the number of chargers.
- **Random Method:** generates an Offering Table with random EV chargers within the configured input radius R, while completely ignoring the objectives of the weighted sum function.

B. Performance Evaluation

In this experimental series, we evaluate the performance of *EcoCharge* against all methods over all datasets introduced, with respect to the average Sustainability Score (SC) and CPU execution time (F_t) for all points in each dataset. SC is based upon 33.3% of the Sustainable Charging Level weight, 33.3% of the Availability weight, and 33.3% of the Derouting Cost weight.

As demonstrated in Figure 6, the Brute Force method achieved the best SC = 100% for all datasets, however, it has the worst execution time of all; F_t ≈345 ms for Oldenburg, ≈348 ms for California, ≈351 ms for T-drive, and F_t ≈355 ms for Geolife. This is because Brute Force exhaustively searches space for an optimal solution. Considering the execution of the Index-Quadtree method, it shows that can efficiently manage sparse data across large areas, allowing for rapid searching and data retrieval. Having a much faster execution time than Brute Force against all datasets F_t≈78-85 ms, the Index-Quadtree approach also provides great performance SC≈80-85%. The Random approach selects randomly chargers by completely ignoring SC, thus, it generates *Offering Tables* much faster than Brute Force, but with the worst SC score achieved out of all methods; SC≈40% at F_t≈10 ms for the smallest dataset, and SC≈35% at F_t≈15 ms for largest one.

According to $EcoCharge$, after different combinations tested regarding the user-configured radius R and range distance Q parameters (i.e., presented in the following experiments), we decided that the best configurations to be adjusted are R = 50 km and Q = 5 km. $EcoCharge$ seems to be the fastest in all datasets against all methods (i.e., F_t ≈56-67 ms). This is due to the fact that our solution does not need to search

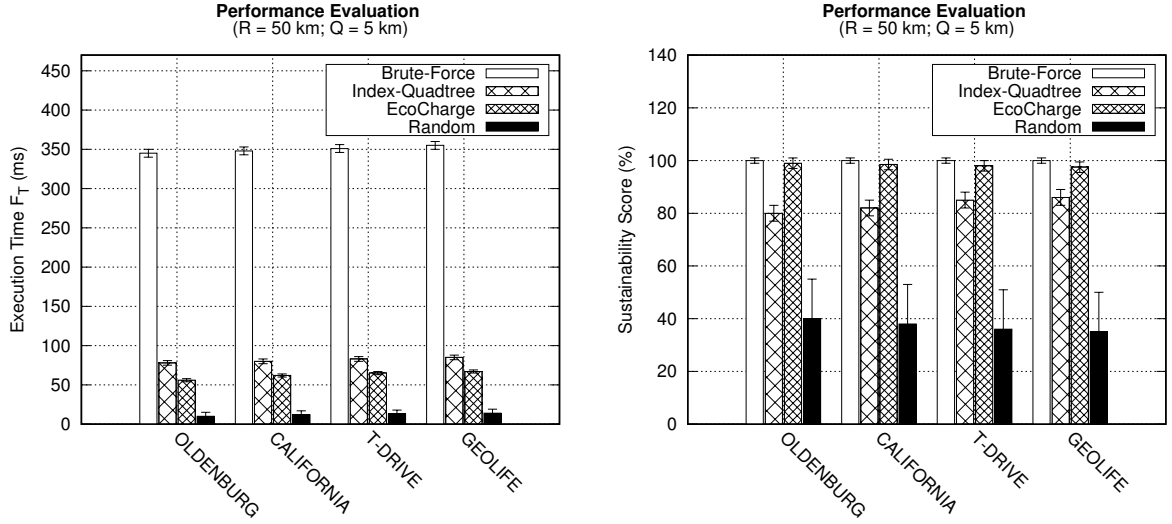

Fig. 6. **Performance Evaluation:** Evaluation in terms of CPU Execution Time and Sustainability Score, based on Brute Force performance.

the entire pool of chargers for every new node location in case R and Q parameters are satisfied. In contrast to other methods, it intelligently inherits existing *Offering Tables* that match the requirements. In addition, it managed to achieve reasonable SC score for Oldenburg at ≈99%, ≈98.5% for California, ≈98% for T-drive, and ≈97.5% for Geolife, respectively. Consequently, EV users can utilize our proposed technique for considerably faster execution obtaining accurate results with very high SC, almost as good as an exhaustive search method. $EcoCharge$ can proficiently balance the trade-off between Derouting Cost, Availability, and Sustainable Charging Level. It has the ability to support EV owners save time and money while reducing their carbon footprint, making it a valuable tool for both individuals and organizations looking to adopt sustainable transportation practices.

C. R-opt Evaluation

In the second experiment, we evaluate the performance of the proposed $EcoCharge$ algorithm against different user configured radius values (R), with respect to the average Sustainability Score (SC) and CPU execution time (F_t) for all points in each dataset. Figure 7 illustrates that when users configure lower radius the execution is faster, however, SC decreases gradually in comparison to higher radius values.

According to the smaller dataset of Oldenburg, the trade-off can be clearly identified regarding the fastest execution having R = 25 km, where F_t = 55 ms with the worst SC = 98.5%, and the slowest execution having R = 75 km, where F_t= 61 ms with the best SC = 99.5%. Similarly for California, the fastest execution occurs when R = 25 km, where F_t= 56 ms with the worst SC = 98%, and the slowest execution occurs when R = 75 km, where F_t= 67 ms with the best SC = 99%. The same stands for the two bigger datasets of T-drive and Geolife, where the fastest execution is achieved when R is configured to 25 km at F_t= 56 ms and 57 ms with the worst score SC 97.5% and 97%, respectively. In a similar manner to the previous cases, the slowest execution occurs when R = 75 km, where F_t = 75 ms and 88 ms with the best score SC = 98.5% and 98%, correspondingly. As expected, the results retrieved for all datasets when R is configured to 50 km are somewhere in between the two aforementioned cases. Consequently, increasing the radius setting means more EV chargers are identified and processed based on user's location (i.e., larger search space), hence, more time is needed for the generation of *Offering Tables*.

Consider for example a generated table of an EV owner using R = 75 km and another EV user using R = 25 km. In the first case (i.e., R = 75 km), since EV user does not mind traveling farther for charging by setting higher radius, they will get considerably more charger options in comparison to the second driver (i.e., R = 25 km). Hence, it is at the discretion of the user to dedicate more execution time for retrieving an O with numerous charger choices, or to spend less execution time utilizing only the available chargers within a close neighborhood area.

D. Q-opt Evaluation

In the third experimental series, we evaluate the performance of the proposed algorithm against various distance range values (Q), with respect to the average Sustainability Score (SC) and CPU execution time (F_t) for all points in each dataset. Figure 8 shows that when the configured range distance value is longer, then the execution time is slightly faster, however, SC gradually drops in comparison to shorter range distance values.

Regarding the smaller dataset of Oldenburg, the trade-off can be identified based on the fastest execution when Q is configured to 15 km, where F_t= 38 ms with the worst SC = 97%, in contrast when Q is set to 5 km, where the slowest execution F_t= 56 ms with the best SC = 99%. Similarly for California, the fastest execution having Q = 15 km is F_t= 39 ms with the worst SC = 96.5%, and the slowest

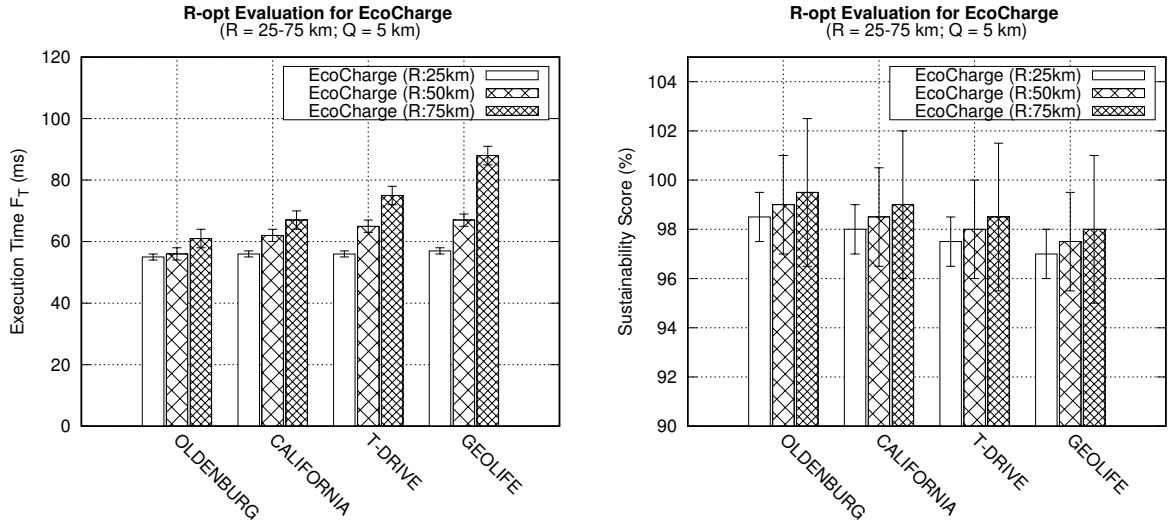

Fig. 7. **R-opt Evaluation of EcoCharge:** Evaluation in terms of CPU Execution Time and Sustainability Score based on different configured radius values, based on Brute Force performance.

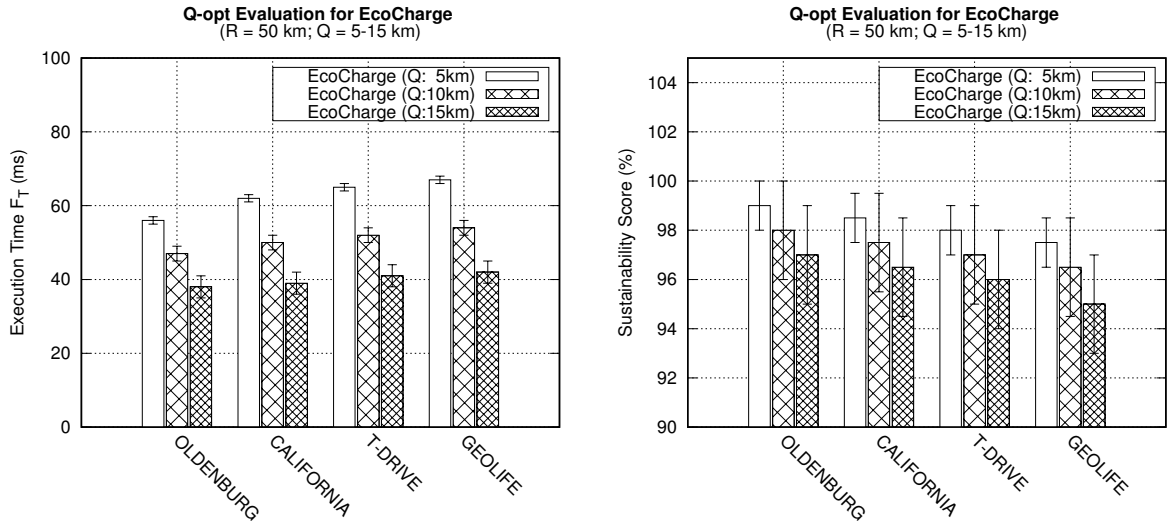

Fig. 8. **Q-opt Evaluation of EcoCharge:** Evaluation in terms of CPU Execution Time and Sustainability Score based on different range distance values, based on Brute Force performance.

execution having Q = 5 km is F_t= 62 ms with the best SC = 98.5%. The fastest execution of the two bigger datasets of T-drive and Geolife, is achieved when Q = 15 km, where F_t= 41 ms and 42 ms with the worst score SC 96% and 95%, respectively. Correspondingly, the slowest execution occurs when Q = 5 km, where F_t = 65 ms and 67 ms with the best score SC = 98% and 97.5%. The results obtained for all datasets when Q is configured to 10 km are somewhere in between the two aforementioned cases, as expected. As we increase $EcoCharge$'s range distance parameter Q, we observe minor decrease in both, the execution time and the Sustainability Score. This is due to the fact that the farther previous EV location is from current, the less accurate the previous generated results are, since chargers' production data might change and a recalculation of SC is required.

Consider for example a case where Q is set to 15 km and an *Offering Table* is already generated for the EV driver's current location. Therefore, if next node's location range distance is within 15 km, no new *Offering Table* will be generated. That means in case next location is at $\approx$ 13 km a recalculation of O is skipped, thus, it is possible that some chargers' production data might change and previously generated results won't be applicable according to the new location. Consequently, it is recommended to use shorter range distances Q, even if that means devoting some extra F_t for higher SC.

E. Ablation study

The fourth experimental series is an ablation study of weight parameters on Sustainability Score (SC). Particularly, we evaluate the performance of EcoCharge against different distance functions:

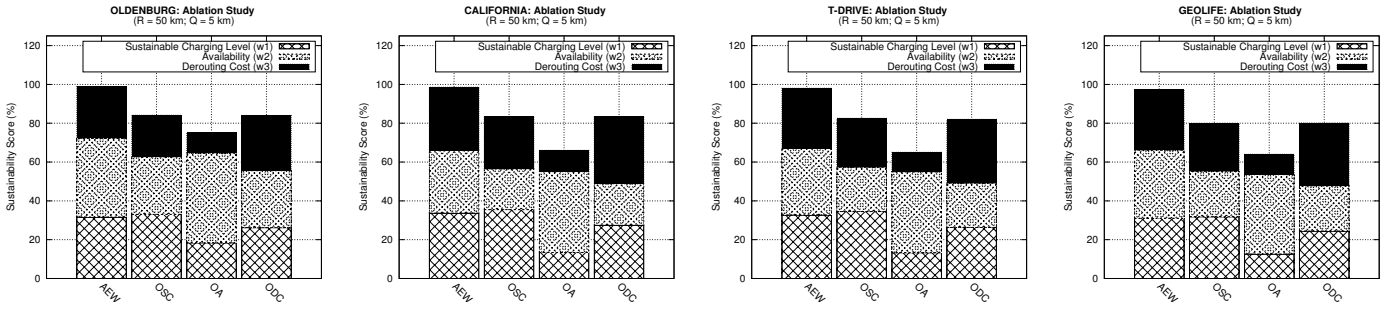

Fig. 9. **Ablation Study of Weight Parameters and their impact on Sustainability Score:** Evaluation in terms of EcoCharge performance according to different weight values, based on Brute Force performance.

- **AWE**: considers *all* weights equally (i.e., w_1=w_2=w_3). This is the default configuration of EcoCharge.
- **OSC**: considers *only* Sustainable Charging Level (w_1)
- **OA**: considers *only* Availability (w_2)
- **ODC**: considers *only* Derouting Cost (w_3)

Figure 9 demonstrates different distance functions in terms of SC for all datasets. We can clearly see that AWE outperforms all functions, while it achieves , $w_1 \approx$ 31.5-35%, $w_2 \approx$ 32-40%, $w_3 \approx$ 26.5-31% , and SC score ranging at $\approx$ 97.5-99%. The following distance functions are compared with AWE since it has the best output. According to OSC, where the emphasis is given only on Sustainable Charging Level, we observe an increase on $w_1 \approx$ 2-3%, with a reduction on $w_2 \approx$ 9-12% and $w_3 \approx$ 5-6%. As OA targets only Availability, there is a notable increase on $w_2 \approx$ 6-9%, and a drastic decrease on $w_1 \approx$ 13-15% and $w_3 \approx$ 16-22%, resulting in a much lower $SC \approx$ 64-75%. ODC focuses on EV rankings only based on Derouting, where we notice an increase on $w_3 \approx$ 2-4%, and a decrease on the other two weights; $w_1 \approx$ 5-7%, and $w_2 \approx$ 10-13%, respectively. This causes the total SC score to also decrease by about $\approx$ 15-18%.

The findings reveal a distinct interplay between these parameters, showing for example that focusing only on Derouting could affect Charging Level or disregarding Availability could also have an impact on Derouting. The system's performance appears to be highly sensitive to the weight configurations, suggesting that fine-tuning these weights could be crucial for optimizing EcoCharge's operation. Our experiments revealed that the balance between w_1, w_2, and w_3 is key to the system's overall efficiency.

VI. Related Work

In this section, we provide pointers to the interested reader by presenting various green mobility approaches in road networks covering topics related to EV charging, *kNN* query processing and sustainable systems.

A. EV Charging

Research on EV charging has grown significantly in recent years, given the rise of electric vehicles' popularity. Several studies have been conducted on EV charging infrastructure deployment, charging station location planning, and optimization algorithms for EV charging [4], [17]. Additionally, some studies have focused on user behavior and preferences regarding EV charging, and the development of smart charging systems to enable demand-side management and energy balancing [2]. The research approach in [2], presents a framework for strategically deploying charging stations in a city by utilizing a polynomial time approximation algorithm. It consists of two optimization components, the optimal charging station placement that minimizes the average time to travel to the nearest charging station, and the optimal charging point assignment that minimizes the average waiting time for an available charging point. In [4], an EV charging station placement method is proposed, called SocialAware Optimal Electric Vehicle Charger Deployment (SOCD), which considers multiple complex social influences of EV chargers arrangement. The authors utilized two algorithms to tackle the matter, while considering both urban and rural areas, but without any sustainability provisions. Furthermore, in [18], the authors applied a phase abstraction approach with Markov chains to accurately estimate energy consumption in EV trips using kernel density, and developed a trip planner tool that can efficiently model the EV energy consumed in future trips along a route. An EV path-planning system is introduced in [19], where the waiting time at charging stations is considered, but no sustainability aspects are considered.

B. kNN for Spatio-Temporal Data

In applications involving spatio-temporal data, datasets comprise objects and queries that traverse through time within an Euclidean space. Current works in this domain only address the challenge of responding to a k-nearest neighbor query for a single user throughout the temporal dimension (referred to as a CkNN query). Formally, the kNN of an object o from some dataset O, denoted as $kNN(o, O)$, are the k objects that have the most similar attributes to o. Specifically, given objects $o_a \neq o_b \neq o_c$, $\forall o_b \in kNN(o_a, O)$ and $\forall o_c \in O - kNN(o_a, O)$ it always holds that $dist(o_a, o_b) \leq dist(o_a, o_c)$[9].

In the context of large-scale disk-resident datasets, Frentzos et al. [20], Benetis et al. [21], Tao et al. [8], Iwerks et al. [22], and Raptopoulou et al. [23], posit the assumption that

[9]In our discussion, *dist* can be any L_p-norm distance metric, such as Manhattan (L_1), Euclidean (L_2) or Chebyshev (L_∞).

the velocity of the moving objects remains constant, allowing for the estimation of an object's future position. Huan et al. [24] assume the existence of an uncertainty degree in the velocity and direction attributes of moving objects. In response to this premise, they introduce algorithms designed to optimize scenarios where uncertainty extends to the estimation of future positions. This research employs time-parameterized R-trees to conduct efficient searches for the nearest neighbors. Kollios et al. [25] introduce an approach for addressing nearest-neighbor (NN) queries concerning moving objects within a one-dimensional spatial context. The foundation of their method lies in a dual transformation, where a native space line segment corresponds to a point in the transformed space, and vice-versa. Xiong et al. [26] focus on addressing multiple k-NN queries and propose an incremental search approach that relies on hashing objects into a regular grid, with a specific emphasis on optimizing CPU time. Regarding the disk-resident data, the primary aim is to reduce disk input/output (I/O) operations and prioritize CPU time as a secondary objective.

Main-memory processing is typically imperative for spatio-temporal applications, particularly when dealing with highly mobile objects. The frequency of location updates poses significant constraints for disk-based storage and indexing, necessitating CPU-time optimization. The optimization of kNN queries, similar to the approach employed by Xiong et al. in the context of disk-resident data, is also addressed by Mouratidis et al. [27], Hu et al. [28], and Yu et al. [29]. Data objects are indexed by a grid in main-memory given a grid size system-defined parameter. All approaches employ an iterative process of expanding a range search to identify the kNN for each query. Assuming a grid size of $\sqrt{n}$ cells, their stateless solution has a time complexity of O($n^{1.5}$) for uniform distributions and O($n^{2.5}$) for the worst-case distribution, where the search for most of the users needs to be deepened iteratively until it covers most of the space.

Formally, an *All kNN (AkNN)* query generates a kNN graph. It computes the *kNN*(o, O) result for every $o \in O$ and has a quadratic worst-case bound. An AkNN query can alternatively be viewed as a *kNN Self-Join*: *Given a dataset O and an integer k, the kNN Self-Join of O combines each object $o_a \in O$ with its k nearest neighbors from O, i.e.,* $O \bowtie_{kNN} O = \{(o_a, o_b) | o_a, o_b \in O \text{ and } o_b \in kNN(o_a, O)\}$. In [7], we devise a high-performance distributed main-memory algorithm named Spitfire, which carries out the AkNN computation in the cloud. Such an operator could be useful shall we decide to implement EcoCharge in Mode 2 (i.e., cloud mode).

C. Sustainable Home Energy Management Systems

In our previous publications, we have presented *Energy Planner (EP)* and *Green Planner (GP)*, integrated in a system called $IMCF^+$ [30], [31]. Both, EP and GP, adapted off-the-shelf AI algorithms (hill climbing and simulated annealing), and focus on "long-term" planning, meaning that they would compute a whole year plan by doing less complex daily computations. For example, $IMCF^+$ generates a residential plan while considering the family's configured annual energy budget (e.g., 11500 kWh) and Rule Automation Workflow (RAW) pipelines. The high-level system's objective is to identify the rules that must be dropped so that users stay within the desired annual energy budget. Furthermore, we developed a system called *GreenCap* [32], which refers to "daily" planning as it attempts to find the best combination for allocating and shifting appliances during a day by minimizing the imported energy from the grid, while considering peak demand and high energy production times.

VII. Conclusion

In this paper, we present $EcoCharge$, an innovative framework for sustainable EV charging by leveraging renewable energy sources, optimizing charging strategies, and reducing operational costs. The *CkNN-EC* query approach and the dynamic caching method applied, allow for efficient computation and optimization of the renewable hoarding process, resulting in significant speedups and minimizing CPU execution time, proving also accurate results very close to an exhaustive search method.

The ability to accumulate renewable energy during production periods and take advantage of self-consumption is a key feature, making $EcoCharge$ an environmentally-friendly alternative to traditional charging methods. Providing accurate *Offering Tables* in a reasonable response time by taking into consideration *Estimated Components (ECs)* (e.g., sustainable charging level, availability, derouting costs) the proposed algorithm allows drivers to make informed decisions and choose the most sustainable charging stations along their scheduled route, while reducing the environmental impact of transportation.

In the future, we plan to extend our solution by integrating $EcoCharge$ with smart grid technologies and taking advantage of off-peak electricity rates and grid stabilization services. Additionally, we plan to investigate the balance of the produced traffic to chargers by the suggested *Offering Tables*, and monitor the congestion to redirect drivers to alternative EV charging stations.

References

[1] S. Schmoll, S. Friedl, and M. Schubert, "Scaling the dynamic resource routing problem," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, ser. SSTD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 80–89. [Online]. Available: https://doi.org/10.1145/3340964.3340983

[2] Y. Li, J. Luo, C. Chow, K. Chan, Y. Ding, and F. Zhang, "Growing the charging station network for electric vehicles with trajectory data analytics," in *2015 IEEE 31st International Conference on Data Engineering (ICDE)*, 2015, pp. 1376–1387.

[3] S. N. Basharzad, F. M. Choudhury, E. Tanin, L. H. Andrew, H. Samet, and M. Sarvi, "Electric vehicle charging: It is not as simple as charging a smartphone (vision paper)," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3557915.3560967

[4] Q. Liu, Y. Zeng, L. Chen, and X. Zheng, "Social-aware optimal electric vehicle charger deployment on road network," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 398–407. [Online]. Available: https://doi.org/10.1145/3347146.3359382

[5] S. Constantinou, A. Konstantinidis, and D. Zeinalipour-Yazti, "Green planning systems for self-consumption of renewable energy," *IEEE Internet Computing*, pp. 1–1, 2022.

[6] S. Jung and J. Ko, "Study on regenerative energy recovery of electric vehicle through voltage control using switched capacitor," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4324–4339, 2021.

[7] G. Chatzimilioudis, C. Costa, D. Zeinalipour-Yazti, W. C. Lee, and E. Pitoura, "Distributed in-memory processing of all k nearest neighbor queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 925–938, 2016.

[8] Y. Tao, D. Papadias, and Q. Shen, "Chapter 26 - continuous nearest neighbor search," in *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*, P. A. Bernstein, Y. E. Ioannidis, R. Ramakrishnan, and D. Papadias, Eds. San Francisco: Morgan Kaufmann, 2002, pp. 287–298. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781558608696500330

[9] "United nations framework convention on climate change: The paris agreement," *https://unfccc.int/process-and-meetings/the-paris-agreement*, 2023.

[10] "National centers for environmental information - global forecast system (gfs)," *https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast*, 2023.

[11] "European centre for medium-range weather forecasts - advancing global nwp through international collaboration," *https://www.ecmwf.int/*, 2023.

[12] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Advances in Spatial and Temporal Databases*, C. Bauzer Medeiros, M. J. Egenhofer, and E. Bertino, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 273–290.

[13] T. Brinkhof, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, 153-180, 2002.

[14] Y. Jing, Z. Yu, Z. Chengyang, X. Wenlei, X. Xing, S. Guangzhong, and H. Yan, "T-drive: driving directions based on taxi trajectories," in *18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'10)*. New York, NY, USA: Association for Computing Machinery, 2010, p. 99–108.

[15] Y. Zheng, X. Xie, and W. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–39, 2010. [Online]. Available: http://sites.computer.org/debull/A10june/geolife.pdf

[16] "California distributed generation statistics-california solar initiative (csi)," *https://www.californiadgstats.ca.gov/*, 2023.

[17] C. Claramunt, C. Bassem, D. Zeinalipour-Yazti, B. Zheng, G. Trajcevski, and K. Torp, "Spatial data management for green mobility," in *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3589132.3625626

[18] P. Rajan and C. V. Ravishankar, "The phase abstraction for estimating energy consumption and travel times for electric vehicle route planning," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 556–559. [Online]. Available: https://doi.org/10.1145/3347146.3359383

[19] J. C. Gareau, E. Beaudry, and V. Makarenkov, "An efficient electric vehicle path-planner that considers the waiting time," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 389–397. [Online]. Available: https://doi.org/10.1145/3347146.3359064

[20] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis, "Algorithms for nearest neighbor search on moving object trajectories." Geoinformatica 11, 2007, p. 159–193.

[21] R. Benetis, C. Jensen, G. Karciauskas, and S. Saltenis, "Nearest neighbor and reverse nearest neighbor queries for moving objects," in *Proceedings International Database Engineering and Applications Symposium*, 2002, pp. 44–53.

[22] G. S. Iwerks, H. Samet, and K. Smith, "Continuous k-nearest neighbor queries for continuously moving points with updates," ser. VLDB '03. VLDB Endowment, 2003, p. 512–523.

[23] K. Raptopoulou, A. Papadopoulos, and Y. Manolopoulos, "Fast nearest-neighbor query processing in moving-object databases." Geoinformatica 7, 2003, p. 113–137.

[24] Y. K. Huang, S. J. Liao, and C. Lee, "Efficient continuous k-nearest neighbor query processing over moving objects with uncertain speed and direction," in *Scientific and Statistical Database Management*, B. Ludäscher and N. Mamoulis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 549–557.

[25] G. Kollios, D. Gunopulos, and V. J. Tsotras, "Nearest neighbor queries in a mobile environment," in *Spatio-Temporal Database Management*, M. H. Böhlen, C. S. Jensen, and M. O. Scholl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 119–134.

[26] X. Xiong, M. Mokbel, and W. Aref, "Sea-cnn: scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases," in *21st International Conference on Data Engineering (ICDE'05)*, 2005, pp. 643–654.

[27] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on management of data*, 2005, pp. 634–645.

[28] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 479–490. [Online]. Available: https://doi.org/10.1145/1066157.1066212

[29] X. Yu, K. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving objects," in *21st International Conference on Data Engineering (ICDE'05)*, 2005, pp. 631–642.

[30] S. Constantinou, A. Konstantinidis, P. K. Chrysanthis, and D. Zeinalipour-Yazti, "Green planning of iot home automation workflows in smart buildings," *ACM Trans. Internet Things*, jun 2022.

[31] S. Constantinou, A. Konstantinidis, D. Zeinalipour-Yazti, and P. K. Chrysanthis, "The iot meta-control firewall," in *37th IEEE ICDE*, April 19 - April 22, 2021, Chania, Crete, 2021, conference, p. 12 pages.

[32] S. Constantinou, N. Polycarpou, C. Costa, A. Konstantinidis, P. K. Chrysanthis, and D. Zeinalipour-Yazti, "An iot data system for solar self-consumption," in *24th IEEE International Conference on Mobile Data Management (MDM)*, July 3- July 6, 2023, Singapore, 2023, conference, p. 10 pages.