Ατομική Διπλωματική Εργασία

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΙΜΗΣΗ ΓΕΩΓΡΑΦΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΗΛΙΑΚΗ ΦΟΡΤΙΣΗ ΗΛΕΚΤΡΙΚΩΝ ΟΧΗΜΑΤΩΝ

Δημήτρης Παπαζαχαρίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση και Αποτίμηση Γεωγραφικού Συστήματος για Ηλιακή Φόρτιση Ηλεκτρικών Οχημάτων

Δημήτρης Παπαζαχαρίου

Επιβλέπων Καθηγητής
Δημήτρης Ζεϊναλιπούρ

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2024

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Δρ. Δημήτρη Ζεϊναλιπούρ για την ανάθεση της διπλωματικής αυτής εργασίας, την καθοδήγηση, τις συμβουλές, την υποστήριξη και την μεταλαμπάδευση των γνώσεων του, όχι μόνο κατά τη διάρκεια εκπόνησης της εργασίας αυτής, αλλά καθ' όλη τη διάρκεια των προπτυχιακών μου σπουδών στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου.

Επίσης, θα ήθελα να ευχαριστώ τον Σωτήρη Κωνσταντίνου, διδακτορικό φοιτητή του Εργαστηρίου Συστημάτων Διαχείρισης Δεδομένων στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου για την καθοδήγηση και τις συμβουλές του καθ' όλη τη διάρκεια εκπόνησης της εργασίας αυτής.

Αναγνώριση

Η υλοποίηση του συστήματος EcoCharge που περιγράφεται σε αυτήν τη διπλωματική εργασία θα παρουσιαστεί ως demo στο 25ο Διεθνές Συνέδριο του IEEE για τη Διαχείριση Δεδομένων Κινητού Υπολογισμού.

S. Constantinou, D. Papazachariou, C. Costa, A. Konstantinidis, M. Mokbel, and D. Zeinalipour-Yazti, “EcoCharge: A Framework for Sustainable Electric Vehicles Charging”, in 25th IEEE International Conference on Mobile Data Management, June 24 - June 27, 2024, Brussels, Belgium.

Περίληψη

Η διπλωματική αυτή εργασία ασχολείται με την υλοποίηση και την αποτίμηση ενός γεωγραφικού συστήματος για την ηλιακή φόρτιση των ηλεκτρικών οχημάτων.

Τα τελευταία χρόνια η αγορά ηλεκτρικών οχημάτων έχει τεράστια αύξηση, ενώ υπάρχει και αυξανόμενο ενδιαφέρον για την ενσωμάτωση των ανανεώσιμων πηγών ενέργειας στις υποδομές φόρτισης των ηλεκτρικών οχημάτων (φωτοβολταϊκά πάνελ, ανεμογεννήτριες). Αν η ενέργεια όμως με την οποία φορτίζονται τα οχήματα αυτά προέρχεται από μη ανανεώσιμη ενέργεια μπορεί να αναιρεθούν τα περιβαλλοντικά οφέλη της ηλεκτροκίνησης. Το EcoCharge είναι ένα καινοτόμο σύστημα που υπολογίζει και παρουσιάζει σε ένα χάρτη τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης ηλεκτρικών οχημάτων σε κάθε σημείο του χάρτη. Το πρόβλημα μοντελοποιήθηκε ως ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που ανακτά τους k πλησιέστερους γείτονες κάθε σημείου λαμβάνοντας υπόψη τα εκτιμώμενα στοιχεία (κόστος παρέκκλισης από προγραμματισμένη διαδρομή, διαθεσιμότητα φορτιστή, βιώσιμο επίπεδο φόρτισης).

Για την υλοποίηση του συστήματός μας χρησιμοποιήθηκαν για το backend: Python 3, Flask, CORS, GeoPy, ενώ για το frontend: HTML, CSS, JavaScript, Leaflet, Leaflet Routing Machine, Leaflet.Locate, Leaflet.markercluster, Bootstrap, JQuery, Mapbox, χάρτες από OpenStreetMap, OpenTopoMap και ESRI.

Με βάση τα πειράματά μας συμπεραίνουμε ότι όταν αυξάνεται η ακτίνα, μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης, αυξάνεται και ο χρόνος εκτέλεσης του αλγόριθμου και ο βαθμός βιωσιμότητας. Επίσης, όταν αυξάνεται η απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής μειώνεται ο χρόνος εκτέλεσης του αλγόριθμου και αυξάνεται ελάχιστα ή παραμένει ίδιος ο βαθμός βιωσιμότητας. Ακόμη, όταν οι φορτιστές φορτώνονται από το αρχείο που περιέχει όλους τους φορτιστές αυξάνεται κατά πολύ ο χρόνος εκτέλεσης του αλγόριθμου σε σχέση με το αρχείο που περιέχει τους φορτιστές κάθε πόλης/χώρας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Σε αυτό το εισαγωγικό κεφάλαιο παρουσιάζονται το κίνητρο για την ανάπτυξη του συστήματος EcoCharge, το πρόβλημα που πρέπει να αντιμετωπιστεί, η συνεισφορά του EcoCharge και μια συνοπτική περιγραφή της διπλωματικής αυτής εργασίας.

1.1 Κίνητρο

Η αγορά ηλεκτρικών οχημάτων έχει εκθετική αύξηση τα τελευταία χρόνια, λόγω των εξαιρετικών πλεονεκτημάτων τους έναντι των συμβατικών οχημάτων εσωτερικής καύσης, όσον αφορά τη βιώσιμη μετακίνηση και τη σχέση κόστους αποδοτικότητας. Οι πόλεις διαδραματίζουν καθοριστικό ρόλο στην επίτευξη της κλιματικής ουδετερότητας μέχρι το 2050, τον στόχο της Ευρωπαϊκής Πράσινης Συμφωνίας, καθώς ευθύνονται για πάνω από το 65% της παγκόσμιας κατανάλωσης ενέργειας και το 70% των παγκόσμιων εκπομπών CO2 [1].

Τα τελευταία χρόνια, υπάρχει αυξανόμενο ενδιαφέρον για την ενσωμάτωση των ανανεώσιμων πηγών ενέργειας στις υποδομές φόρτισης των ηλεκτρικών οχημάτων (φωτοβολταϊκά πάνελ, ανεμογεννήτριες) [2], [3].

Οι άνθρωποι συχνά φορτίζουν τα ηλεκτρικά τους οχήματα κατά τις ελεύθερες τους ώρες, ακόμη και όταν η μπαταρία δεν είναι σημαντικά εξαντλημένη, για να εξασφαλίσουν ότι το όχημα θα φορτιστεί επαρκώς όταν θα χρειαστεί να ταξιδέψουν. Αυτή τη συνήθεια την ονομάζουμε συσσώρευση ενέργειας. Τα ηλεκτρικά οχήματα αποτελούν ένα τρόπο για τη προστασία του περιβάλλοντος και τη μείωση των εκπομπών αερίων του θερμοκηπίου. Η συσσώρευση ενέργειας με μη ανανεώσιμη ενέργεια αναιρεί τα περιβαλλοντικά οφέλη. Η ζήτηση για ενέργεια για φόρτιση ηλεκτρικών οχημάτων το 2020 στις ΗΠΑ ήταν περίπου 4.7 Terawatt hours και αναμένεται να αυξηθεί σε περίπου 107 Terawatt hours ως το 2035 [4].

Η βελτιστοποίηση της φόρτισης των ηλεκτρικών οχημάτων χρησιμοποιώντας ανανεώσιμες πηγές ενέργειας, αποτελεί έναν σημαντικό τομέα έρευνας αυτή την περίοδο [5]. Οι τρέχουσες εφαρμογές (π.χ. Plugshare, IONITY, Tesla Supercharger, EnBW) επικεντρώνονται στο να βοηθούν τους χρήστες να εντοπίζουν τα σημεία φόρτισης των οχημάτων τους και χρησιμοποιούν ανανεώσιμες πηγές ενέργειας.

Διάφορες τεχνικές βελτιστοποίησης, όπως η έξυπνη φόρτιση [6] και τα συστήματα vehicle-to-grid (V2G) [7], εξετάζονται με στόχο να βελτιώσουν το περιβαλλοντικό αποτύπωμα της διαδικασίας φόρτισης και να ελαχιστοποιήσουν την υπερφόρτωση στο δίκτυο ηλεκτρικής ενέργειας.

Επιπλέον, οι μέθοδοι για αναπαραγωγή ενέργειας εφαρμόζονται σε φορτωμένα ηλεκτρικά φορτηγά κατά τη διάρκεια της καθοδικής τους πορείας μέσω ανάκτησης ενέργειας, δημιουργώντας έτσι επαρκή ενέργεια για την ανάβαση του κενού φορτηγού [8].

Μια τεχνική ανανεώσιμης συσσώρευσης ενέργειας μπορεί να εφαρμοστεί σε περιπτώσεις που ο χρήστης του ηλεκτρικού οχήματος έχει ελεύθερο χρόνο (όταν περιμένει ή είναι σταθμευμένος):

1. ηλεκτρικά ταξί (π.χ., Lyft, Uber, Bolt) κατά τις περιόδους που είναι ελεύθερα
2. γονείς που περιμένουν στα ηλεκτρικά οχήματά τους, ενώ τα παιδιά τους κάνουν δραστηριότητες μετά το σχολείο
3. χρήστες ηλεκτρικών οχημάτων που πηγαίνουν για ψώνια

4. χρήστες ηλεκτρικών οχημάτων που βρίσκονται σε συνεδρίαση, εκδήλωση ή συνέδριο

Συνεπώς, σε όλες τις προαναφερθείσες περιπτώσεις, οι χρήστες θα μπορούσαν να επισκεφθούν κάποιον κοντινό φορτιστή για να φορτίσουν αποδοτικά τα ηλεκτρικά τους οχήματα χρησιμοποιώντας ενέργεια από ανανεώσιμες πηγές, μειώνοντας έτσι το αποτύπωμα άνθρακα από την καθημερινή τους ρουτίνα.

Η απόφαση για το πού θα γίνει η βιώσιμη συσσώρευση ενέργειας εξαρτάται από διάφορα εκτιμώμενα στοιχεία, όπως το κόστος παρέκκλισης για να φτάσει κανείς στον φορτιστή που εξαρτάται από την εκτιμώμενη κυκλοφοριακή κίνηση, η (διαθέσιμη καθαρή) ενέργεια στο φορτιστή που εξαρτάται από τον εκτιμώμενο καιρό και η διαθεσιμότητα του φορτιστή που εξαρτάται από τα εκτιμώμενα χρονοδιαγράμματα που δείχνουν πότε ένας φορτιστής είναι πολυσύχναστος. Για το σκοπό αυτό, μπορεί να χρησιμοποιηθεί ένα συνεχές ερώτημα k πλησιέστερων γειτόνων [9], το οποίο μπορεί να ανακτήσει τους k πλησιέστερους γείτονες μιας δοθείσας διαδρομής, ωστόσο δεν λαμβάνει υπόψη την εκτίμηση διάφορων στοιχείων.

1.2 Πρόβλημα

Η μετάβαση στη χρήση ηλεκτρικών οχημάτων αποτελεί ένα σημαντικό βήμα προς την προστασία του περιβάλλοντος και τη μείωση των εκπομπών αερίων του θερμοκηπίου. Ωστόσο, ένα σημαντικό πρόβλημα που πρέπει να αντιμετωπιστεί είναι η πηγή της ενέργειας με την οποία φορτίζονται τα οχήματα αυτά. Αν η ενέργεια προέρχεται από μη ανανεώσιμη ενέργεια, δηλαδή ενέργεια που προέρχεται από την καύση ορυκτών καυσίμων, όπως το πετρέλαιο και το μαζούτ, μπορεί να αναιρεθούν τα περιβαλλοντικά οφέλη της ηλεκτροκίνησης. Η αποτελεσματική χρήση ηλεκτρικών οχημάτων για την ελάχιστη επίδραση στο περιβάλλον απαιτεί την ανάπτυξη και την ευρεία χρήση ανανεώσιμων πηγών ενέργειας, όπως η ηλιακή και η αιολική ενέργεια, για τη φόρτισή τους. Μόνο με αυτόν τον τρόπο μπορούμε να εξασφαλίσουμε τη μακροπρόθεσμη βιωσιμότητα της ηλεκτροκίνησης και τα οφέλη της για το περιβάλλον.

Επίσης, για να βρεθούν οι k πλησιέστεροι φορτιστές σε ένα σημείο χρησιμοποιείται ένα συνεχές ερώτημα k πλησιέστερων γειτόνων, ωστόσο δεν λαμβάνει υπόψη την εκτίμηση διάφορων στοιχείων. Επομένως, απαιτείται ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που λαμβάνει υπόψη τα εκτιμώμενα στοιχεία.

Το EcoCharge [10] είναι ένα καινοτόμο πλαίσιο εργασίας, το οποίο έχει ως στόχο τη βελτιστοποίηση της φόρτισης των ηλεκτρικών οχημάτων, με φορτιστές που χρησιμοποιούν μόνο ανανεώσιμες πηγές ενέργειας και επικεντρώνεται αποκλειστικά στις μετακινήσεις μικρής διάρκειας εντός των αστικών περιοχών, αγνοώντας το σχεδιασμό μετακινήσεων με πολλές στάσεις (π.χ., A Better Routeplanner).

Αυτοί οι φορτιστές μεγιστοποιούν την αυτόνομη κατανάλωση ανανεώσιμης ενέργειας (π.χ., ηλιακή), μειώνοντας έτσι την παραγωγή διοξειδίου του άνθρακα και επίσης την ανάγκη για ακριβές σταθερές μπαταρίες στο δίκτυο ηλεκτρικής ενέργειας για την αποθήκευση ανανεώσιμης ενέργειας που δεν μπορεί να χρησιμοποιηθεί διαφορετικά.

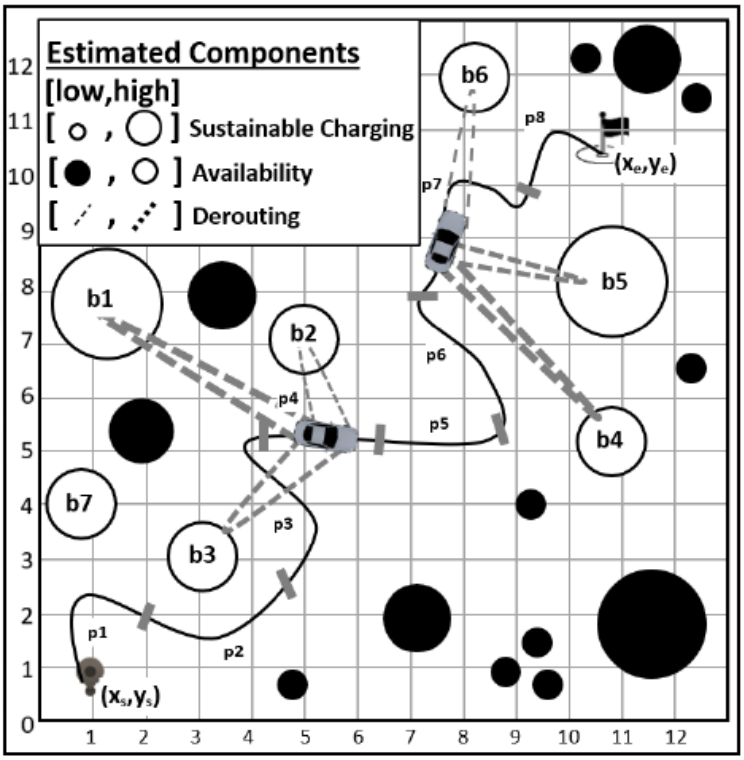
Το πρόβλημα μοντελοποιήθηκε ως ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που ανακτά τους k πλησιέστερους γείτονες κάθε σημείου κατά μήκος της διαδρομής, όπου η συνάρτηση απόστασης υπολογίζεται χρησιμοποιώντας εκτιμώμενα στοιχεία. Ένα εκτιμώμενο στοιχείο ορίζει μια συνάρτηση, η οποία μπορεί να έχει μια ασαφή τιμή βασισμένη σε κάποιες εκτιμήσεις. Τα εκτιμώμενα στοιχεία που χρησιμοποιούνται στο EcoCharge είναι:

1. το κόστος παρέκκλισης, το οποίο είναι ο χρόνος που απαιτείται για να φτάσει κανείς στο φορτιστή, εξαρτώμενος από την εκτιμώμενη κυκλοφοριακή κίνηση
2. η διαθεσιμότητα του φορτιστή, η οποία εξαρτάται από τα εκτιμώμενα χρονοδιαγράμματα που δείχνουν πότε ένας φορτιστής είναι πολυσύχναστος
3. η (διαθέσιμη καθαρή) ενέργεια στο φορτιστή, η οποία εξαρτάται από τον εκτιμώμενο καιρό

Αφού ο χρήστης επιλέξει την βαρύτητα που θέλει να έχουν τα εκτιμώμενα στοιχεία, ο αλγόριθμος του EcoCharge προτείνει τους κορυφαία καταταγμένους φορτιστές ηλεκτρικών οχημάτων και τους παρουσιάζει σε ένα χάρτη.

Ας υποθέσουμε ότι B είναι ένα σύνολο φορτιστών ηλεκτρικών οχημάτων σε ένα οδικό δίκτυο, όπου ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που λαμβάνει υπόψη τα εκτιμώμενα στοιχεία ανακτά τον πλησιέστερο γείτονα (k=1) κάθε τμήματος p της

διαδρομής P. Συγκεκριμένα, το αποτέλεσμα είναι ένα σύνολο <b, p>, όπου ο b είναι ένας φορτιστής από το B. Ως ένα απλοποιημένο παράδειγμα, ας θεωρήσουμε το Σχήμα 1.1, όπου B={b1,...,b20}. Έτσι, λαμβάνοντας υπόψη τη διαδρομή P και επικεντρώνοντας την προσοχή μας στη μετακίνηση προς τον τελικό προορισμό, το αποτέλεσμα 1NN του ερωτήματος είναι {< b7, p1 >, < b3, p2 >, < b3, p3 >, < b1, p4 >, < b4, p5 >, < b5, p6 >, < b5, p7 >, < b5, p8 >}, που σημαίνει ότι ο φορτιστής b7 είναι ο πλησιέστερος γείτονας για το τμήμα διαδρομής p1.



Σχήμα 1. 1 Παράδειγμα ανάκτησης πλησιέστερου γείτονα

Το EcoCharge χρησιμοποιεί ένα συνεχές ερώτημα k πλησιέστερων γειτόνων με εκτιμώμενα στοιχεία και μια δυναμική τεχνική κρυφής μνήμης για τη δημιουργία πινάκων με βιώσιμους φορτιστές για μετακινήσεις στην πόλη, σε λογικό χρόνο απόκρισης, ελαχιστοποιώντας την παρέκκλιση των διαδρομών και μεγιστοποιώντας το επίπεδο βιώσιμης φόρτισης. Η κατάταξη προέρχεται από τη συνάρτηση κόστους, η οποία χρησιμοποιεί μια επαναλαμβανόμενη διαδικασία για την καθορισμό των συνόλων των k

πλησιέστερων γειτόνων με φορτιστές ηλεκτρικών οχημάτων από το σημείο ερωτήματος εντός ενός προκαθορισμένου παραθύρου χρόνου, λαμβάνοντας υπόψη τα εκτιμώμενα στοιχεία. Το EcoCharge μπορεί να χρησιμοποιηθεί με τρεις διαφορετικούς τρόπους:

1. στο ενσωματωμένο λειτουργικό σύστημα ενός οχήματος (π.χ., Android Automotive OS, Volkswagen OS3)
2. οι υπολογισμοί πραγματοποιούνται κεντρικά σε ένα server
3. η διαχείριση της λειτουργικότητας γίνεται από μια συσκευή άκρου (π.χ., έξυπνο τηλέφωνο με χρήση Android Auto ή Apple CarPlay)

Η αποτελεσματικότητα του EcoCharge μετριέται με βάση:

1. το επίπεδο βιώσιμης φόρτισης
2. τη διαθεσιμότητα του φορτιστή
3. το κόστος παρέκκλισης
4. το χρόνο εκτέλεσης της CPU

Σκοπός του EcoCharge είναι να συνεισφέρει σημαντικά στην αγορά της περιβαλλοντικά συνειδητής φόρτισης των ηλεκτρικών οχημάτων, ιδιαίτερα σε ευθυγράμμιση με τους περιβαλλοντικούς στόχους της Ευρωπαϊκής Ένωσης για το 2030 [11].

1.3 Συνεισφορά

Η διπλωματική αυτή εργασία ασχολείται με την υλοποίηση και την αποτίμηση ενός γεωγραφικού συστήματος για την ηλιακή φόρτιση των ηλεκτρικών οχημάτων.

Το EcoCharge είναι ένα καινοτόμο σύστημα που υπολογίζει και παρουσιάζει σε ένα χάρτη τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης ηλεκτρικών οχημάτων σε κάθε σημείο του χάρτη. Το πρόβλημα μοντελοποιήθηκε ως ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που ανακτά τους k πλησιέστερους γείτονες κάθε σημείου λαμβάνοντας υπόψη τα εκτιμώμενα στοιχεία (κόστος παρέκκλισης από προγραμματισμένη διαδρομή, διαθεσιμότητα φορτιστή, βιώσιμο επίπεδο φόρτισης). Το EcoCharge μπορεί να χρησιμοποιηθεί με τρεις διαφορετικούς τρόπους: στο ενσωματωμένο λειτουργικό σύστημα ενός οχήματος, οι υπολογισμοί πραγματοποιούνται κεντρικά σε ένα server και η διαχείριση της λειτουργικότητας γίνεται

από μια συσκευή άκρου (π.χ., έξυπνο τηλέφωνο). Σκοπός του EcoCharge είναι να συνεισφέρει σημαντικά στην αγορά της περιβαλλοντικά συνειδητής φόρτισης των ηλεκτρικών οχημάτων.

Με βάση τα πειράματά μας συμπεραίνουμε ότι όταν αυξάνεται η ακτίνα, μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης, αυξάνεται και ο χρόνος εκτέλεσης του αλγόριθμου και ο βαθμός βιωσιμότητας. Επίσης, όταν αυξάνεται η απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής μειώνεται ο χρόνος εκτέλεσης του αλγόριθμου και αυξάνεται ελάχιστα ή παραμένει ίδιος ο βαθμός βιωσιμότητας. Ακόμη, όταν οι φορτιστές φορτώνονται από το αρχείο που περιέχει όλους τους φορτιστές αυξάνεται κατά πολύ ο χρόνος εκτέλεσης του αλγόριθμου σε σχέση με το αρχείο που περιέχει τους φορτιστές κάθε πόλης/χώρας.

Στο μέλλον, σχεδιάζεται η ενσωμάτωση του EcoCharge με τεχνολογίες έξυπνων δικτύων και η εκμετάλλευση των τιμών της ηλεκτρικής ενέργειας τις ώρες μη υψηλής ζήτησης και των υπηρεσιών σταθεροποίησης του δικτύου. Επιπλέον, σχεδιάζεται η παρακολούθηση της κυκλοφοριακής συμφόρησης για την ανακατεύθυνση των οδηγών προς εναλλακτικούς σταθμούς φόρτισης ηλεκτρικών οχημάτων.

1.4 Συνοπτική περιγραφή εργασίας

Η διπλωματική αυτή εργασία αποτελείται από επτά κεφάλαια.

Στο Κεφάλαιο 1 παρουσιάζονται το κίνητρο για την ανάπτυξη του συστήματος EcoCharge, το πρόβλημα που πρέπει να αντιμετωπιστεί και η συνεισφορά του EcoCharge.

Στο Κεφάλαιο 2 περιγράφονται διάφορες υπηρεσίες και επιστημονικές μελέτες που ασχολούνται με την φόρτιση και την διαδρομή των ηλεκτρικών οχημάτων.

Το Κεφάλαιο 3 αναφέρεται στο μοντέλο, τη λειτουργικότητα, τον αλγόριθμο και την αρχιτεκτονική του συστήματος EcoCharge.

Στο Κεφάλαιο 4 περιγράφεται η υλοποίηση του συστήματος EcoCharge για το backend και το frontend.

Στο Κεφάλαιο 5 παρουσιάζεται το γραφικό περιβάλλον χρήστη του συστήματος EcoCharge.

Το Κεφάλαιο 6 αναφέρεται στην μεθοδολογία και αποτίμηση των πειραμάτων μας.

Τέλος, στο Κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα της διπλωματικής αυτής εργασίας και η μελλοντική εργασία που μπορεί να γίνει για την περαιτέρω ανάπτυξη του EcoCharge.

Κεφάλαιο 2

Σχετική εργασία

Σε αυτό το κεφάλαιο παρουσιάζονται διάφορες υπηρεσίες και επιστημονικές μελέτες που ασχολούνται με την φόρτιση και την διαδρομή των ηλεκτρικών οχημάτων.

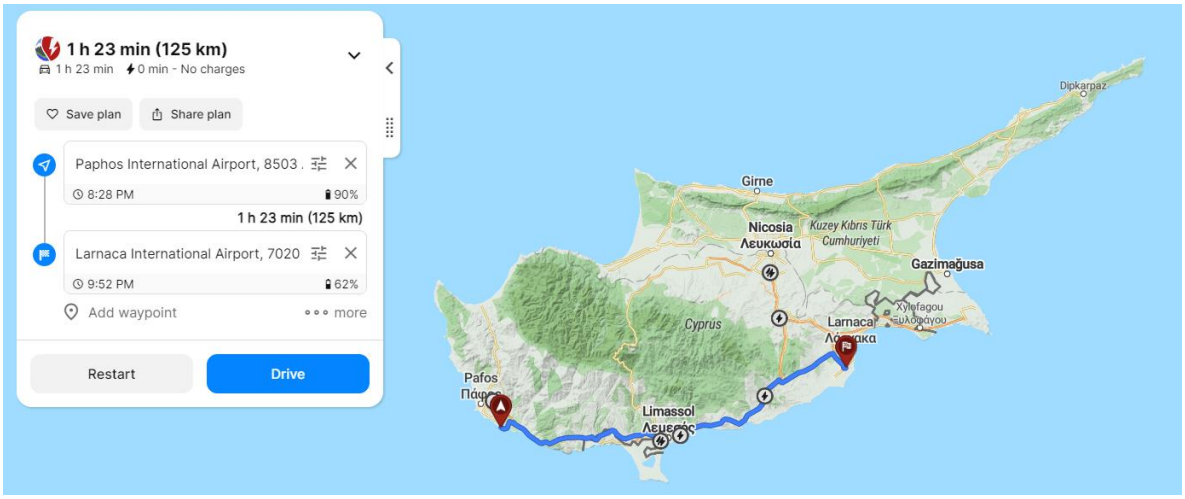
2.1 Υπηρεσίες

Υπάρχουν αρκετές υπηρεσίες που ασχολούνται με την φόρτιση και την διαδρομή των ηλεκτρικών οχημάτων. Στη συνέχεια, παρουσιάζονται οι κυριότερες από αυτές.

2.1.1 A Better Routeplanner (ABRP)

Ο ABRP [12] είναι ένας σχεδιαστής διαδρομών (route planner) που σχεδιάζει διαδρομές για ηλεκτρικά οχήματα με βάση διάφορους παράγοντες, συμπεριλαμβανομένων της διαθέσιμης φόρτισης, της πρόβλεψης της κυκλοφοριακής κίνησης, της ταχύτητας, της πρόβλεψης καιρού και άλλων παραγόντων που επηρεάζουν την απόδοση του οχήματος. Επιτρέπει στους οδηγούς να προγραμματίζουν το ταξίδι τους με ηλεκτρικά οχήματα, λαμβάνοντας υπόψη την απόσταση, τον τύπο του οχήματος, τη διαθέσιμη φόρτιση και άλλες παραμέτρους. Ο σχεδιασμός της διαδρομής γίνεται με γνώμονα την εξοικονόμηση ενέργειας και την ελαχιστοποίηση του χρόνου φόρτισης, προσφέροντας ένα ισορροπημένο σχέδιο ταξιδιού. Οι χρήστες μπορούν να χρησιμοποιήσουν τον ABRP είτε μέσω της ιστοσελίδας του, είτε μέσω εφαρμογής σε κινητά τηλέφωνα, επιτρέποντάς τους να σχεδιάσουν τη διαδρομή τους εκ των προτέρων ή να προσαρμόσουν το ταξίδι τους σε πραγματικό χρόνο κατά τη διάρκεια του. Το γραφικό περιβάλλον χρήστη του ABRP φαίνεται στο Σχήμα 2.1. Παράδειγμα σχεδιασμού της διαδρομής από το αεροδρόμιο Πάφου στο αεροδρόμιο Λάρνακας βλέπουμε στο Σχήμα 2.2.

Σχήμα 2. 1 Γραφικό περιβάλλον χρήστη του ABRP



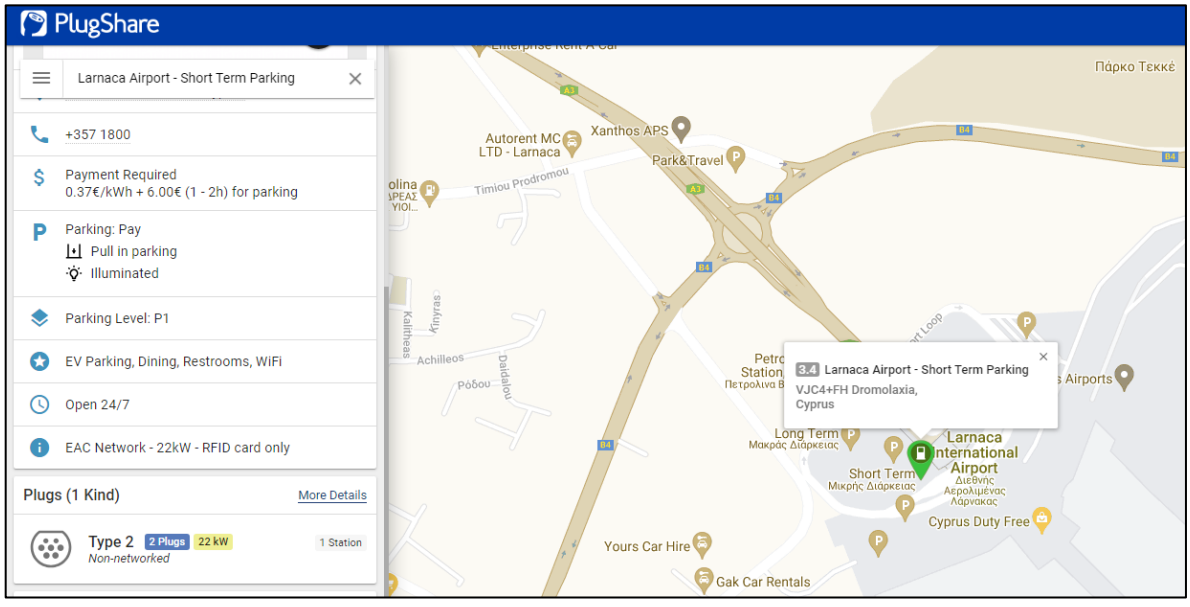
Σχήμα 2. 2 Παράδειγμα σχεδιασμού διαδρομής από τον ABRP

2.1.2 PlugShare

Η PlugShare [13] είναι μια διαδικτυακή πλατφόρμα και κοινότητα που παρέχει πληροφορίες σχετικά με τα σημεία φόρτισης για ηλεκτρικά οχήματα παγκοσμίως. Η εφαρμογή και η ιστοσελίδα της παρέχουν στους χρήστες πληροφορίες σχετικά με την τοποθεσία, τον τύπο και τη διαθεσιμότητα των σημείων φόρτισης, καθώς και αξιολογήσεις

και σχόλια από άλλους χρήστες. Οι χρήστες μπορούν να συμμετέχουν στην κοινότητα PlugShare, να συνδεθούν με άλλους χρήστες και να μοιραστούν εμπειρίες και συμβουλές σχετικά με τα ηλεκτρικά οχήματα και τη φόρτισή τους. Το γραφικό περιβάλλον χρήστη της PlugShare φαίνεται στο Σχήμα 2.3. Πληροφορίες για τον σταθμό φόρτισης ηλεκτρικών οχημάτων στο αεροδρόμιο Λάρνακας βλέπουμε στο Σχήμα 2.4.

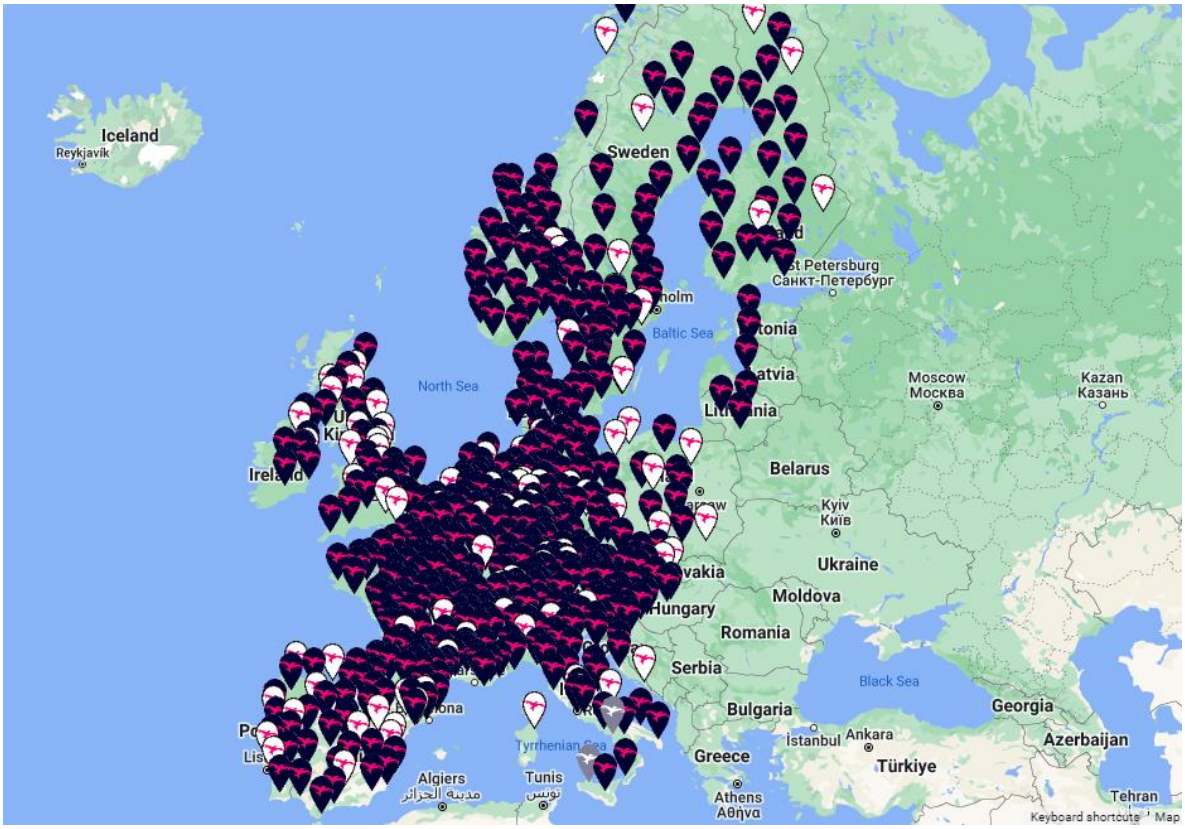
Σχήμα 2. 3 Γραφικό περιβάλλον χρήστη του PlugShare



Σχήμα 2. 4 Πληροφορίες για τον σταθμό φόρτισης στο αεροδρόμιο Λάρνακας

2.1.3 IONITY

Η IONITY [14] είναι μια εταιρεία που δημιουργήθηκε από μια συνεργασία μεταξύ μεγάλων αυτοκινητοβιομηχανιών, στόχος της οποίας είναι η ανάπτυξη και η διαχείριση ενός ευρωπαϊκού δικτύου πολύ γρήγορης φόρτισης για ηλεκτρικά οχήματα. Το δίκτυό της περιλαμβάνει σταθμούς φόρτισης, με πολλαπλά σημεία φόρτισης ανά τοποθεσία, κατά μήκος των αυτοκινητοδρόμων σε 24 ευρωπαϊκές χώρες, κάνοντας τη φόρτιση ηλεκτρικών οχημάτων κατάλληλη για μακρινές διαδρομές. Επιπλέον, χρησιμοποιεί την τεχνολογία CCS (Combined Charging System), η οποία είναι ένα πρότυπο φόρτισης που χρησιμοποιείται ευρέως στην Ευρώπη, για ευρεία συμβατότητα μεταξύ των ηλεκτρικών οχημάτων. Αξίζει να σημειωθεί ότι η IONITY χρησιμοποιεί 100% ανανεώσιμη ενέργεια. Στο Σχήμα 2.5 φαίνεται το δίκτυο φορτιστών της IONITY.

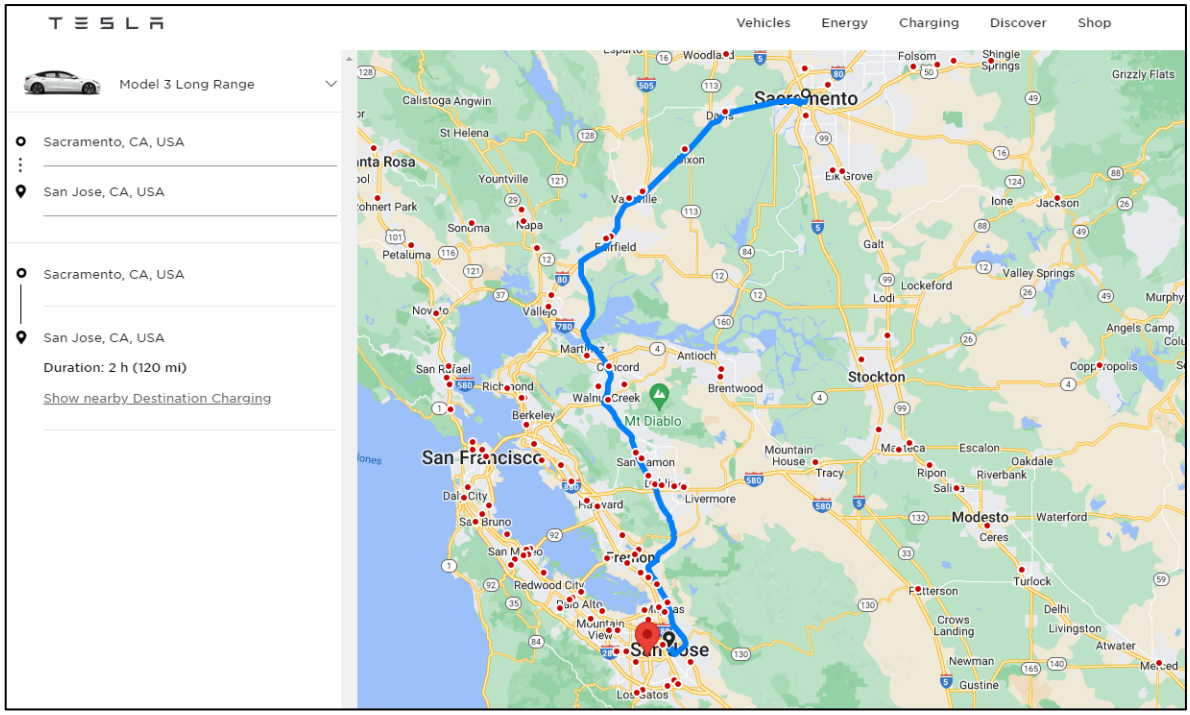


Σχήμα 2. 5 Δίκτυο φορτιστών της IONITY

2.1.4 Tesla Supercharger

Το Tesla Supercharger [15] είναι ένα δίκτυο φορτιστών υψηλής ισχύος που έχουν αναπτυχθεί και διαχειρίζονται από την εταιρεία Tesla. Οι φορτιστές αυτοί σχεδιάστηκαν ειδικά για τα ηλεκτρικά οχήματα της Tesla και παρέχουν τη δυνατότητα για γρήγορη φόρτιση των μπαταριών τους. Με περισσότερους από 50.000 φορτιστές υψηλής ισχύος, η Tesla κατέχει και λειτουργεί το μεγαλύτερο παγκόσμιο δίκτυο γρήγορης φόρτισης στον κόσμο. Οι φορτιστές βρίσκονται κυρίως κοντά σε αυτοκινητόδρομους και κύριες διαδρομές, κάνοντας τις μακρινές διαδρομές πιο εφικτές, αφού μπορούν να προσθέσουν ως και 200 μίλια σε μόλις 15 λεπτά. Η Tesla παρέχει και ένα σχεδιαστή ταξιδιών (trip planner), ο οποίος υπολογίζει αυτόματα τη διαδρομή που επιλέγει ο χρήστης και εμφανίζει τους φορτιστές κατά μήκος της. Στο Σχήμα 2.6 φαίνεται το δίκτυο φορτιστών της Tesla. Παράδειγμα σχεδιασμού της διαδρομής από το Sacramento στο San Jose βλέπουμε στο Σχήμα 2.7.

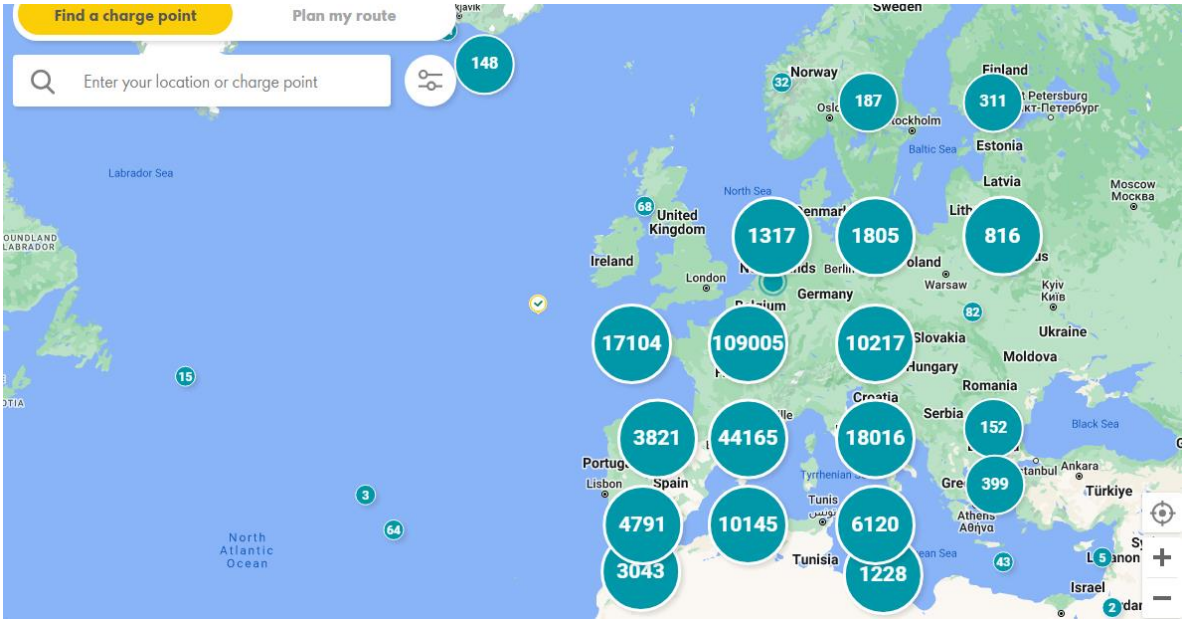
Σχήμα 2. 6 Δίκτυο φορτιστών της Tesla



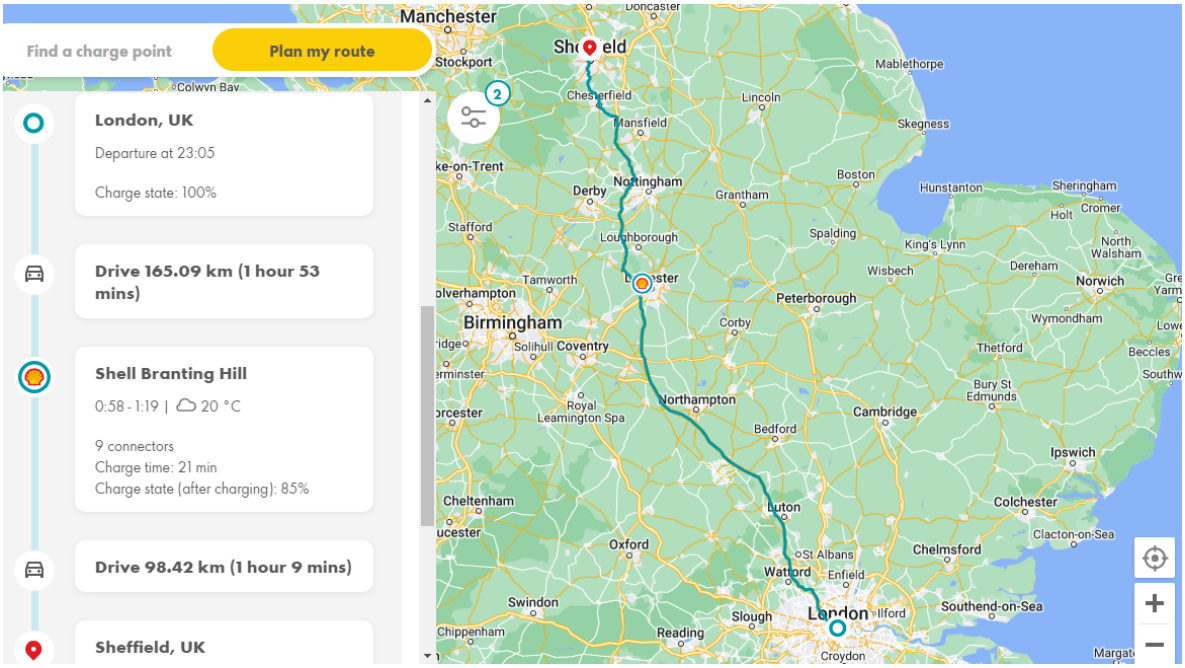
Σχήμα 2. 7 Παράδειγμα σχεδιασμού διαδρομής από τον σχεδιαστή ταξιδιών της Tesla

2.1.5 Shell Recharge

Το Shell Recharge [16] είναι ένα δίκτυο φορτιστών που έχουν αναπτυχθεί και διαχειρίζονται από την εταιρεία Shell. Τα σημεία φόρτισης του Shell Recharge βρίσκονται συνήθως κοντά σε βενζινάδικα της Shell ή σε άλλες τοποθεσίες που διαχειρίζεται η εταιρεία. Οι υπηρεσίες φόρτισης μπορεί να περιλαμβάνουν τόσο αργή όσο και γρήγορη φόρτιση, ανάλογα με τον τύπο του φορτιστή. Οι οδηγοί ηλεκτρικών οχημάτων μπορούν να εντοπίζουν και να χρησιμοποιούν σημεία φόρτισης, να πληρώνουν για τη φόρτιση και να διαχειρίζονται τις φορτίσεις τους μέσω της αντίστοιχης εφαρμογής ή της ιστοσελίδας του Shell Recharge. Η Shell παρέχει και ένα σχεδιαστή διαδρομών (route planner), ο οποίος υπολογίζει τη διαδρομή που επιλέγει ο χρήστης και εμφανίζει τους φορτιστές κατά μήκος της. Στο Σχήμα 2.8 φαίνεται το δίκτυο φορτιστών της Shell. Παράδειγμα σχεδιασμού της διαδρομής από το London στο Sheffield βλέπουμε στο Σχήμα 2.9.



Σχήμα 2. 8 Δίκτυο φορτιστών της Shell



Σχήμα 2. 9 Παράδειγμα σχεδιασμού διαδρομής από τον σχεδιαστή ταξιδιών της Shell

2.1.6 EnBW mobility+

Η EnBW mobility+ [17] είναι μια εφαρμογή που παρέχει υπηρεσίες φόρτισης για ηλεκτρικά οχήματα, καθώς και άλλες υπηρεσίες σχετικά με την φιλική προς το περιβάλλον κινητικότητα, η οποία παρέχεται από τη γερμανική εταιρεία EnBW. Η εταιρεία αυτή δραστηριοποιείται σε διάφορους τομείς της ενέργειας και έχει επεκτείνει τη δραστηριότητά της στον τομέα των ανανεώσιμων πηγών ενέργειας, όπως η αιολική και η ηλιακή ενέργεια, καθώς και στον τομέα των υποδομών φόρτισης για ηλεκτρικά οχήματα. Η EnBW mobility+ προσφέρει πρόσβαση σε ένα ευρύ δίκτυο φορτιστών για ηλεκτρικά οχήματα σε διάφορες τοποθεσίες σε όλη τη Γερμανία και άλλα μέρη της Ευρώπης. Οι χρήστες μπορούν να αποκτήσουν πρόσβαση σε αυτούς τους φορτιστές μέσω της εφαρμογής αυτής, η οποία τους επιτρέπει να βρίσκουν και να κλείνουν θέσεις φόρτισης, να πληρώνουν για τη φόρτιση και να διαχειρίζονται τις φορτίσεις τους.

2.1.7 Σύγκριση υπηρεσιών

Στον Πίνακα 2.1 συγκρίνουμε τις υπηρεσίες αναφορικά με την διαθεσιμότητα τους, τον τύπο φόρτισης των φορτιστών τους, τη χρήση ανανεώσιμων πηγών ενέργειας και την παροχή route planner.

Υπηρεσία	Διαθεσιμότητα	Τύπος φόρτισης	Ανανεώσιμη ενέργεια	Route planner
A Better Routeplanner	Παγκόσμια	Όλοι οι τύποι	Δίνει πληροφορίες	Ναι
PlugShare	Παγκόσμια	Όλοι οι τύποι	Δίνει πληροφορίες	Όχι
IONITY	Ευρώπη	Πολύ γρήγορη	100%	Όχι
Tesla Supercharger	Παγκόσμια	Πολύ γρήγορη	Ναι	Ναι
Shell Recharge	Ευρώπη	Όλοι οι τύποι	Ναι	Ναι
EnBW mobility+	Ευρώπη	Όλοι οι τύποι	Ναι	Όχι

Πίνακας 2. 1 Πίνακας σύγκρισης υπηρεσιών

2.2 Επιστημονικές μελέτες

Υπάρχουν αρκετές επιστημονικές μελέτες που ασχολούνται με την φόρτιση και την διαδρομή των ηλεκτρικών οχημάτων. Στη συνέχεια, παρουσιάζονται οι κυριότερες από αυτές.

2.2.1 Φόρτιση ηλεκτρικών οχημάτων

Η έρευνα σχετικά με την φόρτιση ηλεκτρικών οχημάτων έχει αυξηθεί σημαντικά τα τελευταία χρόνια, προκαλώντας την αύξηση της δημοτικότητας των ηλεκτρικών οχημάτων. Έχουν διεξαχθεί αρκετές μελέτες για την ανάπτυξη των υποδομών για τη φόρτιση ηλεκτρικών οχημάτων, τον σχεδιασμό της τοποθεσίας των σταθμών φόρτισης, και αλγορίθμους βελτιστοποίησης για τη φόρτιση ηλεκτρικών οχημάτων [6], [18]. Επιπλέον, ορισμένες μελέτες έχουν επικεντρωθεί στη συμπεριφορά και τις προτιμήσεις των χρηστών σχετικά με τη φόρτιση ηλεκτρικών οχημάτων και στην ανάπτυξη έξυπνων συστημάτων φόρτισης για τη διαχείριση της ζήτησης και την ισορροπία της ενέργειας [3]. Η έρευνα στην [3] παρουσιάζει ένα πλαίσιο εργασίας για τη στρατηγική ανάπτυξη σταθμών φόρτισης σε μια πόλη, χρησιμοποιώντας έναν αλγόριθμο πολυωνυμικού χρόνου. Αποτελείται από δύο συνιστώσες βελτιστοποίησης, την βέλτιστη τοποθέτηση του σταθμού φόρτισης που ελαχιστοποιεί τον μέσο χρόνο ταξιδιού προς τον πλησιέστερο φορτιστή και την βέλτιστη ανάθεση του σημείου φόρτισης που ελαχιστοποιεί τον μέσο χρόνο αναμονής για ένα διαθέσιμο σημείο φόρτισης. Στην [6], προτείνεται μια μέθοδος τοποθέτησης σταθμών φόρτισης για ηλεκτρικά οχήματα που ονομάζεται Social-Aware Optimal Electric Vehicle Charger Deployment (SOCD), η οποία λαμβάνει υπόψη πολλαπλές πολύπλοκες κοινωνικές επιρροές της διάταξης των φορτιστών ηλεκτρικών οχημάτων. Οι συγγραφείς χρησιμοποίησαν δύο αλγόριθμους για να αντιμετωπίσουν το θέμα, λαμβάνοντας υπόψη τόσο τις αστικές όσο και τις αγροτικές περιοχές, αλλά χωρίς καμία πρόβλεψη για τη βιωσιμότητα. Επιπλέον, στην [19], οι συγγραφείς εφάρμοσαν μια προσέγγιση με χρήση αλυσίδων Markov για να εκτιμήσουν ακριβώς την κατανάλωση ενέργειας σε ταξίδια με ηλεκτρικά οχήματα χρησιμοποιώντας πυκνότητα του πυρήνα, και ανέπτυξαν ένα εργαλείο

σχεδιασμού ταξιδιού που μπορεί να προσομοιώσει αποτελεσματικά την κατανάλωση ενέργειας των ηλεκτρικών οχημάτων σε μελλοντικά ταξίδια κατά μήκος μιας διαδρομής. Ένα σύστημα σχεδιασμού διαδρομής για ηλεκτρικά οχήματα παρουσιάζεται στην [20], όπου λαμβάνεται υπόψη ο χρόνος αναμονής στους σταθμούς φόρτισης, αλλά όχι η βιωσιμότητα.

2.2.2 KNN για χωροχρονικά δεδομένα

Σε εφαρμογές που αφορούν χωροχρονικά δεδομένα, τα σύνολα δεδομένων περιλαμβάνουν αντικείμενα και ερωτήματα που διατρέχουν τον ευκλείδειο χώρο. Οι τρέχουσες μελέτες σε αυτό τον τομέα αντιμετωπίζουν μόνο την πρόκληση της απάντησης σε ένα ερώτημα k-πλησιέστερου γείτονα για έναν μόνο χρήστη σε όλη την χρονική διάσταση (αναφέρεται ως ερώτημα CkNN). Τυπικά, οι kNN ενός αντικειμένου ο από ένα σύνολο δεδομένων O, οι οποίοι δηλώνονται ως $kNN(o, O)$, είναι τα k αντικείμενα που έχουν τα πιο παρόμοια χαρακτηριστικά με το ο. Ειδικότερα, δεδομένων των αντικειμένων $o_a \neq o_b \neq o_c$, $\forall o_b \in kNN(o_a, O)$ και$\forall o_c \in O - kNN(o_a, O)$, πάντα ισχύει ότι $dist(o_a, o_b) \leq dist(o_a, o_c)$.
Στο πλαίσιο των συνόλων δεδομένων μεγάλης κλίμακας που είναι αποθηκευμένα σε δίσκους, οι Frentzos et al. [21], Benetis et al. [22], Tao et al. [9], Iwerks et al. [23], και Raptopoulou et al. [24], υποθέτουν ότι η ταχύτητα των κινούμενων αντικειμένων παραμένει σταθερή, επιτρέποντας την εκτίμηση της μελλοντικής θέσης ενός αντικειμένου. Οι Huan et al. [25] υποθέτουν την ύπαρξη ενός βαθμού αβεβαιότητας στα χαρακτηριστικά της ταχύτητας και της κατεύθυνσης των κινούμενων αντικειμένων. Ανταποκρινόμενοι σε αυτή την υπόθεση, εισάγουν αλγόριθμους που σχεδιάζονται για τη βελτιστοποίηση σεναρίων, όπου η αβεβαιότητα εκτείνεται στην εκτίμηση μελλοντικών θέσεων. Αυτή η έρευνα χρησιμοποιεί δέντρα R για τη διεξαγωγή αποδοτικών αναζητήσεων για τους πλησιέστερους γείτονες. Οι Kollios et al. [26] εισάγουν μια προσέγγιση για την αντιμετώπιση ερωτημάτων πλησιέστερου γείτονα σχετικά με κινούμενα αντικείμενα εντός ενός μονοδιάστατου χωρικού πλαισίου. Η βάση της μεθόδου τους βρίσκεται σε ένα διπλό μετασχηματισμό, όπου ένα τμήμα γραμμής στο φυσικό χώρο αντιστοιχεί σε ένα σημείο στον μετασχηματισμένο χώρο και αντίστροφα. Οι Xiong et al. [27] επικεντρώνονται στην

αντιμετώπιση πολλαπλών kNN ερωτημάτων και προτείνουν μια προσέγγιση αυξητικής αναζήτησης που βασίζεται στο hashing των αντικειμένων σε ένα κανονικό πλέγμα, με ειδική έμφαση στη βελτιστοποίηση του χρόνου CPU. Όσον αφορά τα δεδομένα που είναι αποθηκευμένα σε δίσκους, ο κύριος στόχος είναι η μείωση των λειτουργιών εισόδου/εξόδου (I/O) του δίσκου και ο δευτερεύων στόχος ο χρόνος CPU.

Η επεξεργασία στη κύρια μνήμη είναι επιτακτική για εφαρμογές χωροχρονικών δεδομένων, ιδιαίτερα όταν ασχολούνται με αντικείμενα μεγάλης κινητικότητας. Η συχνότητα των ενημερώσεων για την τοποθεσία προκαλεί σημαντικούς περιορισμούς για την αποθήκευση και ευρετηρίαση σε δίσκο, επιβάλλοντας την βελτιστοποίηση του χρόνου CPU. Η βελτιστοποίηση των kNN ερωτημάτων, παρόμοια με την προσέγγιση που χρησιμοποίησαν οι Xiong et al. στο πλαίσιο των δεδομένων που είναι αποθηκευμένα σε δίσκους, αντιμετωπίζεται επίσης από τους Mouratidis et al. [28], Hu et al. [29] και Yu et al. [30]. Τα αντικείμενα δεδομένων ευρετηριάζονται από ένα πλέγμα στη κύρια μνήμη, δεδομένης μιας παραμέτρου μεγέθους του πλέγματος που καθορίζεται από το σύστημα. Όλες οι προσεγγίσεις χρησιμοποιούν μια επαναληπτική διαδικασία επέκτασης του εύρους αναζήτησης, για να αναγνωρίσουν τους kNN για κάθε ερώτημα. Υποθέτοντας ένα πλέγμα με $\sqrt{n}$ κελιά, η λύση τους χωρίς κατάσταση έχει πολυπλοκότητα χρόνου $O(n^{1.5})$ για ομοιόμορφες κατανομές και $O(n^{2.5})$ για τη κατανομή στη χειρότερη περίπτωση , όπου η αναζήτηση για τους περισσότερους χρήστες πρέπει να επεκταθεί επαναληπτικά μέχρι να καλύψει το μεγαλύτερο μέρος του χώρου.

Τυπικά, ένα ερώτημα All kNN (AkNN) δημιουργεί ένα kNN γράφο και υπολογίζει το αποτέλεσμα $kNN(o, O)$ για κάθε $o \in O$. Ένα ερώτημα AkNN μπορεί εναλλακτικά να θεωρηθεί ως ένα kNN Self-Join: δεδομένου ενός συνόλου δεδομένων O και ενός ακεραίου k, το kNN Self-Join του O συνδυάζει κάθε αντικείμενο $o_a \in O$ με τους k πλησιέστερους γείτονες του από το O, δηλαδή $O \ltimes_{kNN} O = \{(o_a, o_b) | o_a, o_b \in O,\ o_b \in kNN(o_a, O)\}$. Στην [31], οι συγγραφείς σχεδιάζουν έναν υψηλής απόδοσης κατανεμημένο αλγόριθμο στην κύρια μνήμη με το όνομα Spitfire, ο οποίος πραγματοποιεί τον υπολογισμό των AkNN στο cloud. Ένας τέτοιος τελεστής θα μπορούσε να είναι χρήσιμος, αν αποφασίσουμε να υλοποιήσουμε το EcoCharge στο cloud.

2.2.3 Βιώσιμα συστήματα διαχείρισης οικιακής ενέργειας

Στις [32] και [33] οι συγγραφείς παρουσίασαν τα Energy Planner (EP) και Green Planner (GP), ενσωματωμένα σε ένα σύστημα που ονομάζεται IMCF^{+}. Και τα δύο, προσαρμόζουν αλγόριθμους τεχνητής νοημοσύνης (hill climbing και simulated annealing) και επικεντρώνονται στον "μακροπρόθεσμο" σχεδιασμό, πράγμα που σημαίνει ότι υπολογίζουν έναν σχεδιασμό για ολόκληρο το έτος με λιγότερους πολύπλοκους υπολογισμούς καθημερινά. Για παράδειγμα, το IMCF^{+} δημιουργεί έναν οικιακό σχεδιασμό λαμβάνοντας υπόψη το καθορισμένο ετήσιο ενεργειακό προϋπολογισμό της οικογένειας (π.χ., 11500 kWh) και το Rule Automation Workflow (RAW). Ο στόχος του συστήματος είναι ο εντοπισμός των κανόνων που πρέπει να απορριφθούν ώστε οι χρήστες να παραμένουν εντός του επιθυμητού ετήσιου ενεργειακού προϋπολογισμού τους. Επιπλέον, ανέπτυξαν ένα σύστημα που ονομάζεται GreenCap [34], το οποίο αναφέρεται στον "καθημερινό" σχεδιασμό καθώς προσπαθεί να βρει τον καλύτερο συνδυασμό για την κατανομή και μετατόπιση των οικιακών συσκευών κατά τη διάρκεια της ημέρας με την ελαχιστοποίηση της εισαγόμενης ενέργειας από το δίκτυο, λαμβάνοντας υπόψη τις ώρες υψηλής ζήτησης και υψηλής παραγωγής ενέργειας.

2.2.4 Κινούμενα αντικείμενα και συντομότερες διαδρομές

Πολλές τεχνικές εισήχθησαν για τον υπολογισμό της συντομότερης διαδρομής σε οδικά δίκτυα, συμπεριλαμβανομένων των αλγορίθμων του Dijkstra και A* [35], [36]. Ο αλγόριθμος του Dijkstra κάνει αναζήτηση σε ολόκληρο το δίκτυο, κάτι που μπορεί να είναι αναποτελεσματικό λόγω των μη απαραίτητων επισκέψεων στους κόμβους. Ο αλγόριθμος A* βελτιώνει την αποδοτικότητα εισάγοντας ευρετική απόσταση και κατευθύνοντας τη διάσχιση του γράφου προς τον προορισμό. Η αναζήτηση προς δύο κατευθύνσεις μπορεί να μειώσει περαιτέρω τον χώρο αναζήτησης. Αυτοί οι αλγόριθμοι είναι ανεξάρτητοι από τα ευρετήρια και κατάλληλοι για δυναμικά περιβάλλοντα. Πρόσφατες εργασίες επικεντρώνονται στη βελτίωση των αλγορίθμων συντομότερης διαδρομής για

συγκεκριμένους τύπους γράφων, όπως οι γράφοι που εξαρτώνται από το χρόνο, τα πολυτροπικά δίκτυα και οι δυναμικοί γράφοι.

Οι συγγραφείς στην [37] πρότειναν ένα ευρετήριο (V-Tree), το οποίο είναι ένα ισορροπημένο δέντρο αναζήτησης που μπορεί να υποστηρίξει αποτελεσματικά την kNN αναζήτηση και δυναμικές ενημερώσεις των κινούμενων αντικειμένων. Σχεδιάστηκε ένας αλγόριθμος kNN αναζήτησης χρησιμοποιώντας τα όρια για τον αποτελεσματικό υπολογισμό των k πλησιέστερων αντικειμένων, ο οποίος χρησιμοποιεί μια μέθοδο best-first και μπορεί να μειώσει τα άσχετα αντικείμενα. Στην [38], οι συγγραφείς δίνουν έμφαση στο πρόβλημα επεξεργασίας των ερωτημάτων συντομότερης διαδρομής σε υπηρεσίες που βασίζονται στην τοποθεσία. Πρότειναν τρεις μεθόδους αποσύνθεσης ερωτημάτων για να ομαδοποιήσουν τα ερωτήματα και δύο batch αλγόριθμους που εκμεταλλεύονται τα αποσυνθεμένα σύνολα ερωτημάτων. Η εργασία στην [39] επικεντρώνεται στο πρόβλημα των συγκεντρωτικών πλησιέστερων γειτόνων σε οδικά δίκτυα ($FANN_R$) και προτείνει μια σειρά αλγορίθμων για την απάντηση σχετικών ερωτημάτων, συμπεριλαμβανομένου ενός αλγορίθμου που βασίζεται στον αλγόριθμο του Dijkstra, ενός αλγορίθμου βασισμένου στην ουρά και ενός πλαισίου εργασίας που συνδυάζει το Incremental Euclidean Restriction και το kNN.

Ένα ιεραρχικό ευρετήριο που ονομάζεται G* δέντρο για τη βελτιστοποίηση χωρικών ερωτημάτων σε οδικά δίκτυα προτάθηκε στην [40]. Δημιουργεί συντομεύσεις μεταξύ επιλεγμένων φύλλων των κόμβων για να αντιμετωπίσει το πρόβλημα της ανεπάρκειας του G δέντρου. Οι συγγραφείς πρότειναν τρεις αλγορίθμους βασισμένους σε συντομεύσεις για την απάντηση ερωτημάτων απόστασης, k πλησιέστερων γειτόνων και εύρους αντίστοιχα, οι οποίοι είναι πιο αποδοτικοί από τους υπάρχοντες αλγορίθμους στο G δέντρο. Η εργασία που δημοσιεύτηκε στην [41], πρότεινε ένα πρακτικό σχήμα κοινής χρήσης ταξί με την ονομασία mT-Share που εκμεταλλεύεται πλήρως τις πληροφορίες κινητικότητας των αιτημάτων για διαδρομές και των ταξί για να επιτύχει αποτελεσματική ευρετηρίαση και καλύτερη αντιστοίχιση επιβατών-ταξί, ενώ ικανοποιεί τους περιορισμούς στις προθεσμίες των επιβατών και τις χωρητικότητες των ταξί. Το mT-Share ευρετηριάζει τα ταξί και τα αιτήματα διαδρομής με γεωγραφικές πληροφορίες και κατευθύνσεις ταξιδιού και υποστηρίζει δρομολόγηση βασισμένη στη συντομότερη διαδρομή και πιθανολογική δρομολόγηση για να εξυπηρετήσει τα online και τα offline αιτήματα διαδρομής.

2.2.5 Διαστασιμότητα και ετερογένεια

Η διαθεσιμότητα των πληροφοριών και των συσκευών κινητικότητας έχει οδηγήσει σε πολύπλοκα δεδομένα διαδρομών, τα οποία έχουν πολλές διαστάσεις και σχετίζονται με ετερογενείς πηγές σημασιολογικών δεδομένων. Οι συγγραφείς στην [42], παρουσίασαν ένα εργαλείο με την ονομασία AUTOMATISE, το οποίο βοηθά στην ταξινόμηση διαδρομών με πολλαπλά χαρακτηριστικά. Η ανάλυση δεδομένων για την παρακολούθηση της κίνησης και την ανταπόκριση σε καταστάσεις έκτακτης ανάγκης σε ένα ενσωματωμένο εσωτερικό και εξωτερικό χώρο έχει υψηλή σημασία. Το UrbanGen [43] είναι ένα σύστημα που δημιουργεί δεδομένα ακολουθώντας τους περιορισμούς των οδικών δικτύων για τον εξωτερικό χώρο και τις τοπολογίες για τον εσωτερικό χώρο. Ενσωματώνει ένα μοντέλο των εσωτερικών τοπολογιών με τα υπερσύγχρονα οδικά δίκτυα, παραμετροποιεί την κίνηση των αντικειμένων στο ενσωματωμένο μοντέλο και παρέχει λειτουργικότητες για την σειριοποίηση και οπτικοποίηση των δημιουργημένων διαδρομών. Στην [44], ένα πλαίσιο εργασίας βασισμένο σε υπολογιστική όραση, με την ονομασία MapsVision, ανιχνεύει ασυμφωνίες σε δεδομένα κειμένων σε διαφορετικούς παρόχους χαρτών. Συγκρίνει labels σε κείμενα και εντοπίζει τα labels που λείπουν ή έχουν μετατοπιστεί, καθώς και ασυμφωνίες στο μέγεθος και το χρώμα της γραμματοσειράς. Το MapsVision βελτιώνει την ποιότητα των χαρτών όσον αφορά το κείμενο και βοηθά τους εκδότες χαρτών να αναγνωρίζουν ασυνέπειες σε διάφορες πλατφόρμες χαρτών.

2.2.6 Ασφάλεια και ιδιωτικότητα των δεδομένων των διαδρομών

Οι συγγραφείς στην [45] ανέπτυξαν ένα σύστημα διοδίων που κρυπτογραφεί και ανεβάζει τις πληροφορίες των διαδρομών των οχημάτων στις μονάδες που βρίσκονται στην άκρη του δρόμου, οι οποίες στη συνέχεια προωθούνται στο κέντρο ελέγχου κυκλοφορίας. Το σύστημα εγγυάται την ιδιωτικότητα του geolocation των οχημάτων και εμποδίζει τους οδηγούς από εσφαλμένη συμπεριφορά. Η εργασία στην [46] επικεντρώνεται σε ένα lightweight σύστημα πλοήγησης που διατηρεί την ιδιωτικότητα μέσω του σχεδιασμού μιας

πρωτότυπης υπογραφής και του συνδυασμού άλλων κρυπτογραφικών πρωτοκόλλων. Επιτυγχάνει την ανωνυμία των οχημάτων για τρίτους ως προς την τοποθεσία, το σημείο εκκίνησης, τον προορισμό και άλλες πληροφορίες διαδρομής. Το TP^3 είναι ένα πλαίσιο εργασίας για τη διατήρηση της ιδιωτικότητας των διαδρομών σε Mobile Cloud Environments [47], το οποίο αξιολογεί την έκθεση της ιδιωτικότητας των χρηστών που υποβάλλουν τις διαδρομές τους. Χρησιμοποιεί serverless λειτουργίες για να επιτύχει χαμηλό latency, υψηλό throughput και χαμηλά λειτουργικά κόστη. Το TP^3 χρησιμοποιεί έναν αλγόριθμο για την επίλυση ενός προβλήματος βελτιστοποίησης πολλαπλών στόχων, ο οποίος επικεντρώνεται στην ελαχιστοποίηση της ομοιότητας του προφίλ κινητικότητας των χρηστών και στην μεγιστοποίηση του ποσοστού επιτυχίας των αιτημάτων.

Κεφάλαιο 3

Αρχιτεκτονική του συστήματος EcoCharge

Σε αυτό το κεφάλαιο παρουσιάζονται το μοντέλο, η λειτουργικότητα, ο αλγόριθμος και η αρχιτεκτονική του συστήματος EcoCharge, καθώς και οι μετρικές που χρησιμοποιούνται.

3.1 Μοντέλο του EcoCharge

Έστω ότι ένα οδικό δίκτυο είναι ένας **κατευθυνόμενος γράφος με βάρη G = (V,E)**, όπου **V** είναι ένα **σύνολο κόμβων (κορυφών)** και **E** είναι ένα **σύνολο ακμών**. Κάθε κόμβος **v** έχει μια χωρική συντεταγμένη *(v.x, v.y)* που καθορίζει το γεωγραφικό του μήκος και πλάτος. Σε κάθε μια από τις ακμές $(u, v) \in E(u, v \in V)$, ανατίθεται ένα **βάρος w(u, v)**, που αντιπροσωπεύει το ενεργειακό κόστος για τη μετάβαση από το σημείο *u* στο *v*. Μπορεί κανείς να θεωρήσει το κόστος της μετακίνησης ως το μήκος του τμήματος του δρόμου, το χρόνο που απαιτείται για τη διέλευση του τμήματος του δρόμου, ή άλλα κόστη όπως η

κατανάλωση ενέργειας ή οι εκπομπές CO_2. Κάθε **όχημα** που κινείται στο οδικό δίκτυο δηλώνεται με **m** και έχει συγκεκριμένες **γεωγραφικές συντεταγμένες**, $\mathbf{loc_m}$.

Έστω ότι υπάρχει ένα **σύνολο δημόσιων σημείων φόρτισης B** που συνδέονται με κοντινές ανανεώσιμες πηγές ενέργειας σε αστικό περιβάλλον. Επίσης, έστω ότι πολλά ηλεκτρικά οχήματα κινούνται αυθαίρετα στο οδικό δίκτυο *G* και ότι κάθε όχημα *m* είναι εξοπλισμένο με μια εφαρμογή όπως το EcoCharge, που μπορεί να προτείνει σημεία φόρτισης για κάθε **τμήμα διαδρομής p** ενός **προγραμματισμένου ταξιδιού P**.

Η **διαθεσιμότητα της καθαρής ενέργειας** (μετρημένη σε kWh) στον **φορτιστή b ∈ B**, σε ένα **χρονικό παράθυρο t** που καθορίζεται από τον χρήστη, συμβολίζεται με **L**. Η τιμή του *t* προκύπτει από το εκτιμώμενο χρόνο άφιξης που λαμβάνει ο χρήστης μέσω μιας συνεργαζόμενης εφαρμογής πλοήγησης (π.χ., Google Maps ή Waze). Έστω ότι η καθαρή ενέργεια μπορεί να προέρχεται είτε από τοπικές πηγές (π.χ., φωτοβολταϊκά πάνελ που είναι τοποθετημένα τοπικά σε υπόστεγα αυτοκινήτων) είτε εικονικά μέσω net-metered από απομακρυσμένες μονάδες παραγωγής ανανεώσιμης ενέργειας. Δεδομένου ότι η καθαρή ενέργεια μπορεί να παρουσιάζει διακυμάνσεις λόγω των καιρικών συνθηκών, το *L* είναι μια πρόβλεψη, ή εκτιμώμενο στοιχείο. Η **φυσική διαθεσιμότητα ενός φορτιστή** στο χρονικό σημείο *t,* συμβολίζεται με **A**. Επίσης, δεδομένου ότι η *A* βασίζεται στο πόσο πολυσύχναστος είναι ο φορτιστής σε διαφορετικές ώρες της ημέρας, η *A* είναι επίσης ένα εκτιμώμενο στοιχείο. Η ενέργεια που απαιτείται για να φτάσει το όχημα *m* σε ένα φορτιστή *b* είναι το **ενεργειακό κόστος παρέκκλισης** (μετρημένο σε kWh) και συμβολίζεται με **D**. Δεδομένου ότι η κίνηση στο οδικό δίκτυο μπορεί να παρουσιάζει διακυμάνσεις (π.χ., βάσει της ώρας και της ημέρας), το *D* είναι επίσης ένα εκτιμώμενο στοιχείο.

Η εφαρμογή EcoCharge εμφανίζει συνεχώς, καθώς το όχημα *m* είναι σε κίνηση, έναν **πίνακα O** (π.χ., κάθε λίγα λεπτά) που υπολογίζεται είτε στο cloud είτε στο άκρο (δηλαδή, στο όχημα *m*). Σε περιπτώσεις όπου η απόσταση από την προηγούμενη στην τρέχουσα τοποθεσία του ηλεκτρικού οχήματος βρίσκεται εντός μιας διαμορφωμένης παραμέτρου, τότε αποφεύγεται η αναπαραγωγή ενός πίνακα *O* και γίνεται προσαρμογή μιας προηγουμένως δημιουργημένης λύσης.

3.2 Διατύπωση προβλήματος

Σκοπός του EcoCharge είναι να υπολογίζει και να παρέχει σε όλους τους οδηγούς ηλεκτρικών οχημάτων σε ένα οδικό δίκτυο έναν πίνακα *O* με βιώσιμους φορτιστές μέσα σε ένα μεγάλο χώρο αναζήτησης. Η έννοια του πίνακα *O* βασίζεται στον ελαχιστοποιητή της απόστασης από το ηλεκτρικό όχημα στον φορτιστή, ενώ ταυτόχρονα λαμβάνει υπόψη τη διαθεσιμότητα και την πράσινη φόρτιση, εφαρμόζοντας την αυτοκατανάλωση των ανανεώσιμων πηγών ενέργειας.

Η πρόθεση της εφαρμογής EcoCharge είναι να βελτιστοποιήσει μια συνάρτηση για να επιτύχει ένα συμβιβασμό μεταξύ του επιπέδου βιώσιμης φόρτισης του οχήματος *L*, της διαθεσιμότητας των φορτιστών *A* και της απόστασης *D* που παρεκκλίνει προς ένα φορτιστή.

3.3 Λειτουργικότητα του EcoCharge

Το EcoCharge χρησιμοποιεί ένα ερώτημα συνεχούς αναζήτησης των k πλησιέστερων γειτόνων με εκτιμώμενα στοιχεία (CkNN-EC) για να προσδιορίσει τους πλησιέστερους φορτιστές για όλα τα τμήματα της διαδρομής *P*. Ειδικότερα, μπορούμε να σκεφτούμε αυτό το ερώτημα ως την παραγωγή ταυτόχρονα πολλών kNN αποτελεσμάτων, δηλαδή ένα για κάθε τμήμα μιας διαδρομής *P*. Τα αποτελέσματα του υπολογισμού των kNN παράγουν τον πίνακα *O*, ο οποίος αποτελείται από πολλά υποαποτελέσματα O_{p1}, ..., O_{pn}. Για να αποφευχθούν πολλαπλές σαρώσεις στη βάση δεδομένων, περνάει μια φορά ολόκληρη η συλλογή των τμημάτων (μαζί με την αντίστοιχη κάλυψή τους). Η μέθοδος ξεκινά με ένα αρχικό σύνολο σημείων διαχωρισμού (σημεία εντός του τμήματος διαδρομής, στα οποία συμβαίνει μια μετάβαση στη γειτνίαση) που αποτελείται μόνο από δύο σημεία διαχωρισμού (x_s, y_s) και (x_e, y_e) με τα σημεία κάλυψής τους να έχουν οριστεί σε 0 (υποδεικνύοντας ότι οι πλησιέστεροι γείτονες όλων των σημείων εντός του τμήματος p_i είναι ακόμη άγνωστοι). Σε κάθε βήμα, το σύνολο σημείων διαχωρισμού περιλαμβάνει το τρέχον αποτέλεσμα σύμφωνα με όλα τα επεξεργασμένα σημεία μέχρι στιγμής. Για κάθε σημείο διαχωρισμού $(x_i, y_i) \in SL\ (0 \leq i < |SL| - 1) : (x_i, y_i) \in p_i$ και όλα τα σημεία στο p_i

έχουμε τους ίδιους πλησιέστερους γείτονες, εκφρασμένους ως NN_{pi}. Το τελικό αποτέλεσμα περιλαμβάνει κάθε σημείο διαχωρισμού (x_i, y_i) που παραμένει στο σύνολο των σημείων διαχωρισμού μετά τον τερματισμό, μαζί με τους πλησιέστερους γείτονές του NN_{pi}.
Το EcoCharge επεξεργάζεται κάθε όχημα m στο οδικό δίκτυο σε οποιαδήποτε χρονική στιγμή βάσει των ακόλουθων βημάτων:

1. λαμβάνει το προγραμματισμένο ταξίδι *P* του χρήστη σε ένα οδικό δίκτυο *G* και το διαμερίζει σε τμήματα $p \approx 3\text{-}5\ km$ ή αντίστοιχο χρονικό παράθυρο (αυτή η ρύθμιση μπορεί να τροποποιηθεί ανάλογα με τις ανάγκες και τις προτιμήσεις του χρήστη)
2. αναζητά τον πλησιέστερο *b* λαμβάνοντας υπόψη τα *L*, *A*, *D* και υπολογίζει ένα **βαθμό βιώσιμης φόρτισης (SC)**
3. προβάλλει στον οδηγό έναν πίνακα *O* για να αποφασίσει και να επιλέξει ένα σταθμό φόρτισης ηλεκτρικών οχημάτων από τις διαθέσιμες επιλογές

3.4 Μετρικές

Η αποτελεσματικότητα του EcoCharge μετριέται με βάση τις ακόλουθες μετρικές:

1. **επίπεδο βιώσιμης φόρτισης (L)**
2. **βαθμός διαθεσιμότητας (A)**
3. **κόστος παρέκκλισης (D)**
4. **χρόνος εκτέλεσης CPU (F_t)**

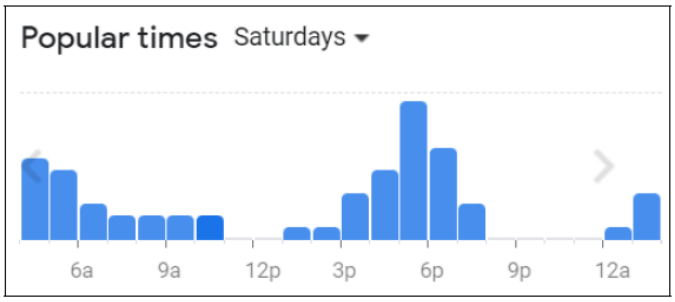
Κάθε σταθμός φόρτισης ηλεκτρικών οχημάτων *b* έχει διαφορετικό ρυθμό φόρτισης και **επίπεδα παραγωγής ενέργειας s_t** που εξαρτώνται από τον χρόνο και την τοποθεσία. Στο EcoCharge, δεν λαμβάνεται υπόψη η ενέργεια που εισάγεται από το δίκτυο, αλλά μόνο το ηλιακό πλεόνασμα που παράγεται, καθώς στόχος είναι η παραγωγή μηδενικών εκπομπών CO_2 κατά τη φάση φόρτισης των ηλεκτρικών οχημάτων (δηλαδή συσσώρευση ανανεώσιμων πόρων). Επιπλέον, το επίπεδο βιώσιμης φόρτισης λαμβάνει υπόψη την πρόγνωση καιρού (ηλιόλουστος, συννεφιασμένος) σε συγκεκριμένη ώρα και τοποθεσία, ο οποίος ανακτάται από μια υπηρεσία στο cloud (π.χ., OpenWeather, Windy, WindFinder), η οποία χρησιμοποιεί μοντέλα καιρού όπως το Global Forecast System (GFS) [48] και το

European Centre for Medium-Range Weather Forecasts (ECMWF) [49], και τα δύο με ακρίβεια 95-96% για μέχρι 12 ώρες και 85-95% για τρεις ημέρες.

$L(B) = \max\{s_t^b \,|\forall b \in B\}$

Η διαθεσιμότητα κάθε φορτιστή ηλεκτρικών οχημάτων εκτιμάται χρησιμοποιώντας κάποια υπηρεσία (π.χ. Google Maps POI busy timetables), όπως φαίνεται στο Σχήμα 3.1.

$A(B) = \max\{A_b|\forall b \in B\}$



Σχήμα 3. 1 Παράδειγμα εκτίμησης της διαθεσιμότητας ενός φορτιστή με βάση τις δημοφιλείς ώρες

Μια διαδρομή από το σημείο εκκίνησης v_0 προς τον φορτιστή v_k είναι μια ακολουθία κόμβων $P = < v_0, v_1, ..., v_k >$, ενώ το w αντιπροσωπεύει το βάρος της ακμής όσον αφορά τις εκπομπές CO_2 (ή kWh που καταναλώθηκαν). Στην περίπτωση που απαιτείται παρέκκλιση από την αρχική προγραμματισμένη διαδρομή για να φτάσει σε έναν επιλεγμένο φορτιστή, θα προστεθούν εκπομπές CO_2 στο συνολικό υπολογισμένο κόστος ταξιδιού. Στην περίπτωση που ο φορτιστής βρίσκεται στη διαδρομή του προγραμματισμένου ταξιδιού, δεν χρειάζεται παρέκκλιση, άρα δεν προστίθενται επιπλέον εκπομπές CO_2. Επιπλέον, η παρέκκλιση λαμβάνει υπόψη πληροφορίες πραγματικού χρόνου για την κυκλοφοριακή κίνηση (π.χ., συμφόρηση) σε συγκεκριμένο χρόνο και τοποθεσία που ανακτώνται από μια cloud υπηρεσία γεωγραφικού συστήματος πληροφοριών (π.χ., Google Maps, Waze, HERE Maps).

Το ελάχιστο κόστος παρέκκλισης για ένα τμήμα p της διαδρομής P προς όλους τους φορτιστές B είναι:

$$D(B) = \min\left\{\sum_{i=0}^{|p|} w_{v_{i,b}} \cdot dis\tan c\, e(v_i, b)\middle| v_i \in p, \forall b \in B\right\}$$

Η εξίσωση εξασφαλίζει την ελαχιστοποίηση του *D* και κατ' επέκταση τη μείωση των εκπομπών CO_2 καθώς υπάρχει συσχέτιση μεταξύ τους.

Ο βαθμός βιωσιμότητας (*SC*) είναι μια συνάρτηση άθροισης με βάρη, όπου το w_1 είναι το βάρος του επιπέδου βιώσιμης φόρτισης (*L*), το w_2 το βάρος της διαθεσιμότητας (*A*) και το w_3 το βάρος του κόστους παρέκκλισης (*D*), αντίστοιχα. Αυτά τα βάρη μπορούν να διαμορφωθούν από τον χρήστη για να προσαρμοστεί η διαδικασία λήψης αποφάσεων σύμφωνα με τις προτιμήσεις και προτεραιότητές του. Χρησιμοποιώντας το συνεχές ερώτημα k πλησιέστερων γειτόνων με το βαθμό βιωσιμότητας ως συνάρτηση απόστασης, το EcoCharge δημιουργεί δύο σύνολα αποτελεσμάτων, ένα βασισμένο στο SC_{min} και ένα άλλο στο SC_{max} και το τελικό αποτέλεσμα είναι η τομή τους, η οποία περιέχει k φορτιστές.

$$SC_{min} = (L_{min} \cdot w_1) + (A_{min} \cdot w_2) + \big((1 - D_{min}) \cdot w_3\big)$$

$$SC_{max} = (L_{max} \cdot w_1) + (A_{max} \cdot w_2) + \big((1 - D_{max}) \cdot w_3\big)$$

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

Επιπλέον, η προτεινόμενη προσέγγιση σχετίζεται και με τον χρόνο εκτέλεσης CPU (F_t), ο οποίος είναι ο χρόνος επεξεργασίας που απαιτείται από τον αλγόριθμο για την εκτέλεση της συνάρτησης βελτιστοποίησης και τη δημιουργία του αποτελέσματος.

3.5 Αλγόριθμος του EcoCharge

Για να παρακολουθεί συνεχώς το αποτέλεσμα των kNN, η μέθοδος CkNN-EC απαιτεί το χωρισμό της απόστασης της διαδρομής σε ξεχωριστά τμήματα, τα οποία εξετάζονται σειριακά για τον καθορισμό των kNN. Η διαδικασία χωρισμού είναι υπεύθυνη για το χωρισμό του προγραμματισμένου ταξιδιού *P* σε τμήματα. Οι αποστάσεις του οδικού δικτύου μεταξύ όλων των φορτιστών και του ηλεκτρικού οχήματος πρέπει να ενημερώνονται κάθε φορά που το ηλεκτρικό όχημα φτάνει σε ένα σημείο διαχωρισμού του προγραμματισμένου ταξιδιού. Για κάθε τμήμα p_i, η διαδικασία εύρεσης των kNN αποτελείται από δύο φάσεις. Η πρώτη ονομάζεται filtering phase και χρησιμοποιείται για να απορρίψει τους μη κατάλληλους φορτιστές. Στη δεύτερη φάση που ονομάζεται refinement, πραγματοποιείται μια αξιολόγηση για να καθοριστεί αν οι υποψήφιοι φορτιστές είναι κατάλληλοι ως CkNN-EC, χρησιμοποιώντας την εξίσωση $SC(B) =$

$sort\big(SC_{max}(b) \cap SC_{min}(b)\big)$, $\forall b \in B$. Στο Σχήμα 3.2 βλέπουμε τον αλγόριθμο του EcoCharge.

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

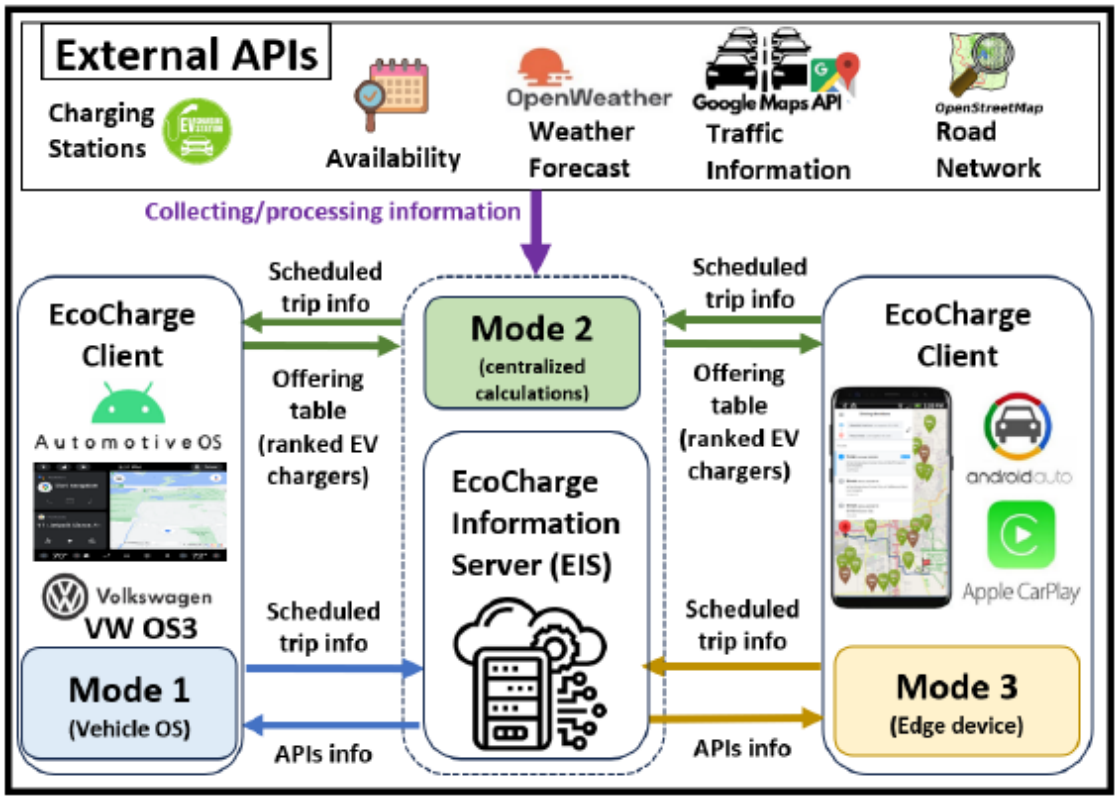
Input: B: set of all EV chargers; P: scheduled trip; m: EV's information; s: solar energy production data;
Output: A generated Offering Table O, consisted of sustainable EV chargers (i.e., CkNN-EC of B)

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

Σχήμα 3. 2 Αλγόριθμος του EcoCharge [10]

3.6 Αρχιτεκτονική του EcoCharge

Ο πυρήνας του συστήματος βρίσκεται στον EcoCharge client που υποστηρίζεται από έναν κεντρικό server, ο οποίος αλληλοεπιδρά με εξωτερικά APIs για την ανάκτηση των δεδομένων (βλ. Σχήμα 3.3). Εκμεταλλευόμενος εξωτερικά APIs, ο EcoCharge Information Server (EIS) διαθέτει δεδομένα πρόβλεψης καιρού σε πραγματικό χρόνο (από OpenWeatherMap [50]), λεπτομερείς πληροφορίες οδικού δικτύου και κυκλοφοριακής κίνησης, τη διαθεσιμότητα των φορτιστών και μια περιεκτική λίστα με όλους τους διαθέσιμους φορτιστές ηλεκτρικών οχημάτων με βάση την τοποθεσία του χρήστη (από PlugShare). Αυτή η κεντρική προσέγγιση επιτρέπει στο server να συγκεντρώνει αποδοτικά τα απαιτούμενα δεδομένα και να τα διανέμει σε μεμονωμένους πελάτες όταν ζητηθούν. Το EcoCharge περιορίζει την ανάγκη για περιττά αιτήματα κλήσης API με την χρήση μηχανισμού έξυπνου caching, που ονομάζεται dynamic caching.



Σχήμα 3. 3 Αρχιτεκτονική του EcoCharge [10]

Η υπηρεσία παρέχεται στους χρήστες με τρεις λειτουργίες:

1. Λειτουργία 1, όπου το EcoCharge λειτουργεί στο ενσωματωμένο λειτουργικό σύστημα ενός οχήματος (π.χ., Automotive OS, Volkswagen OS3)
2. Λειτουργία 2, όπου ο EIS αναλαμβάνει τους υπολογισμούς κεντρικά
3. Λειτουργία 3, όπου η διαχείριση της λειτουργικότητας γίνεται από μια συσκευή άκρου (π.χ., έξυπνο τηλέφωνο με χρήση Android Auto ή Apple CarPlay)

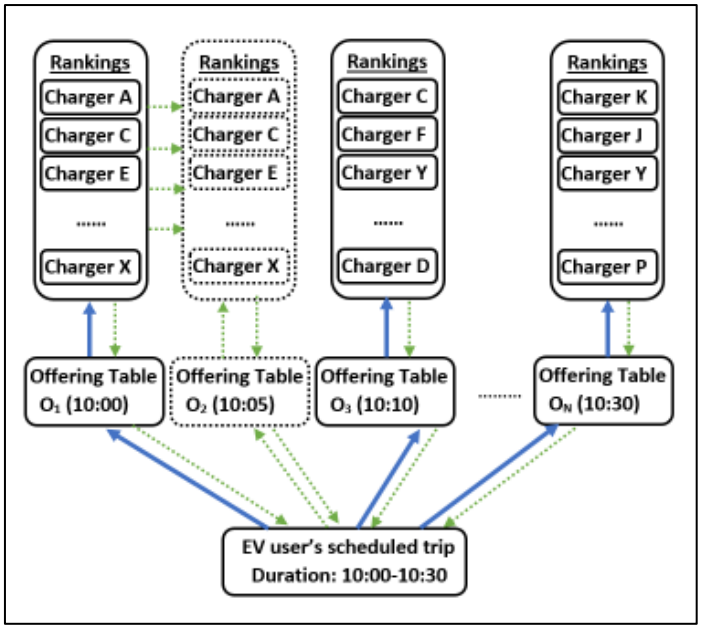
Μετά τη λήψη, μέσω μιας κλήσης API, της πρόβλεψης καιρού, των λεπτομερειών του οδικού δικτύου και των πληροφοριών σχετικά με τις τοποθεσίες φόρτισης ηλεκτρικών οχημάτων από τον EIS, η εφαρμογή του πελάτη αναλαμβάνει τον ρόλο επεξεργασίας αυτών των δεδομένων. Έχοντας ως αρμοδιότητα την βελτιστοποίηση της διαδρομής, η εφαρμογή του πελάτη χρησιμοποιεί τον αλγόριθμο του EcoCharge για να υπολογίσει την πιο αποδοτική διαδρομή λαμβάνοντας υπόψη τη βιώσιμη φόρτιση και το κόστος παρέκκλισης βάσει του προγραμματισμένου ταξιδιού του χρήστη. Αυτή η διαδικασία

περιλαμβάνει τον δυναμικό εντοπισμό των τοποθεσιών φόρτισης ηλεκτρικών οχημάτων κατά μήκος της διαδρομής, λαμβάνοντας υπόψη παράγοντες όπως η ηλιοφάνεια σε πραγματικό χρόνο, η πολυπλοκότητα του οδικού δικτύου και η διαθεσιμότητα των φορτιστών. Επομένως, οι χρήστες λαμβάνουν ενημερωμένες αποφάσεις σχετικά με τη στρατηγική φόρτισης των ηλεκτρικών τους οχημάτων ελαχιστοποιώντας την περιβαλλοντική τους επίδραση.

Ο EcoCharge client συνεχώς υπολογίζει ξανά τη διαδρομή χρησιμοποιώντας ένα παράθυρο περίπου 3-5 λεπτών που θα μπορούσε να αλλάξει την αρχική διαδρομή για να επισκεφθεί ένα σταθμό φόρτισης. Αυτή η απόκλιση γίνεται με στόχο την εύρεση μιας πιο αποδοτικής συνολικής διαδρομής (δηλαδή, από την τρέχουσα τοποθεσία προς το φορτιστή και από το φορτιστή προς τον προορισμό), η οποία περιλαμβάνει την επιπλέον απόσταση προς και από ένα σταθμό φόρτισης ηλεκτρικών οχημάτων.

Το EcoCharge χρησιμοποιεί dynamic caching χωρίζοντας τον χώρο του προβλήματος σε μικρότερα, πιο απλά υποπροβλήματα και επιλύοντας κάθε ένα μόνο μια φορά. Αποθηκεύει τις λύσεις και τις αποκρίσεις του API σε έναν πίνακα και τις χρησιμοποιεί για να επιλύσει παρόμοια προβλήματα μέχρις ότου το συνολικό πρόβλημα επιλυθεί πλήρως. Το πλεονέκτημα της τεχνικής αυτής είναι η δυνατότητα αποφυγής περιττών υπολογισμών και μείωσης της συνολικής πολυπλοκότητας του προβλήματος με την επαναχρησιμοποίηση ήδη δημιουργημένων λύσεων σε αντίστοιχες περιπτώσεις, οδηγώντας έτσι σε σημαντική επιτάχυνση. Υπόθεση για caching μπορεί να γίνει, καθώς ο χώρος απόφασης έχει προσωρινή συμπεριφορά, δηλαδή μια λύση θα ακυρωθεί φυσικά μετά από ένα συγκεκριμένο χρονικό σημείο (t), αφού τα L, A, D θα είναι φυσικά άκυρα μετά το t. Ειδικότερα, χρησιμοποιήθηκε μια προσέγγιση από κάτω προς τα πάνω (bottom-up), όπου τα υποπροβλήματα επιλύονται ξεχωριστά και τα αποτελέσματά τους αποθηκεύονται σε έναν πίνακα, ο οποίος μπορεί στη συνέχεια να χρησιμοποιηθεί για να επιλυθούν μελλοντικά ερωτήματα.

Έστω ότι έχουμε την τρέχουσα τοποθεσία p_i ενός χρήστη ηλεκτρικού αυτοκινήτου σε ένα οδικό δίκτυο με διάφορους σταθμούς φόρτισης B. Το EcoCharge θα ξεκινήσει κανονικά να υπολογίζει το βαθμό βιωσιμότητας του κάθε φορτιστή SC_b, επιλύοντας υποπροβλήματα, και θα δημιουργήσει ένα πίνακα O_1 με βάση την τοποθεσία του χρήστη (βλ. Σχήμα 3.4).



Σχήμα 3. 4 Παράδειγμα dynamic caching bottom-up [10]

Ο πίνακας O_1 αποτελείται από τους κορυφαία καταταγμένους σταθμούς φόρτισης, με τον σταθμό φόρτισης που έχει τον υψηλότερο βαθμό βιωσιμότητας να κατατάσσεται πρώτος. Στη συνέχεια, ακολουθώντας την επόμενη τοποθεσία του οδηγού p_{i+1} σύμφωνα με το προγραμματισμένο ταξίδι του, θα πρέπει να αντιμετωπιστεί ένα νέο πρόβλημα. Λαμβάνοντας υπόψη την απόσταση μεταξύ της τελευταίας και της τρέχουσας τοποθεσίας, μια ήδη υπολογισμένη λύση μπορεί να επαναχρησιμοποιηθεί στο πλαίσιο της επίλυσης ενός μεγαλύτερου προβλήματος. Για να αντιμετωπιστεί αυτό, χρησιμοποιείται η **ακτίνα R**, η οποία επιτρέπει στους χρήστες να λαμβάνουν φορτιστές ηλεκτρικών οχημάτων εντός της επιθυμητής γεωγραφικής τους ακτίνας και η **παράμετρος Q**, η οποία υποδεικνύει την επιθυμητή από τους χρήστες απόσταση από την προηγούμενη στην τρέχουσα τοποθεσία, για τη λήψη ενημερώσεων από τον server και τον υπολογισμό νέων λύσεων. Με αυτό τον τρόπο, ο αλγόριθμός του EcoCharge δεν χρειάζεται να επαναλαμβάνει τον πλήρη χώρο αναζήτησης των φορτιστών για κάθε νέο τμήμα διαδρομής, σε περίπτωση που οι απαιτήσεις της προηγούμενης τοποθεσίας των ηλεκτρικών οχημάτων ταιριάζουν με τις παραμέτρους R και Q. Επομένως, ο O_1 μπορεί να προσαρμοστεί σε O_2, στην περίπτωση που οι προϋποθέσεις των παραμέτρων ικανοποιούνται, και αυτό συνεχίζεται στα επόμενα τμήματα της διαδρομής μέχρι τον τελικό προορισμό του χρήστη. Συγκεκριμένα, πριν

δημιουργηθεί ένας νέος πίνακας και δοθεί στον χρήστη, το EcoCharge εξετάζει την προηγούμενη και την τρέχουσα τοποθεσία προκειμένου να αποφασίσει εάν πρέπει να δημιουργηθεί ξανά μια νέα λύση ή μπορεί να εφαρμόσει την λύση που δημιουργήθηκε προηγουμένως. Ως εκ τούτου, ελαχιστοποιείται ο χρόνος εκτέλεσης CPU εξαλείφοντας περιττούς υπολογισμούς στο μεγάλο χώρο αναζήτησης των φορτιστών.

Κεφάλαιο 4

Υλοποίηση του συστήματος EcoCharge

Σε αυτό το κεφάλαιο παρουσιάζεται η υλοποίηση του συστήματος EcoCharge για το backend και το frontend, η οποία θα παρουσιαστεί ως demo στο 25ο Διεθνές Συνέδριο του IEEE για τη Διαχείριση Δεδομένων Κινητού Υπολογισμού [51].

4.1 Υλοποίηση για το backend

Το αρχείο που περιέχει τον κώδικα για το backend ονομάζεται "ecocharge_demo.py". Ο κώδικας για το backend είναι γραμμένος σε Python 3.

4.1.1 Python 3

Η Python 3 [52] είναι η τρίτη κύρια έκδοση της γλώσσας προγραμματισμού Python. Είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού που απολαμβάνει μεγάλη δημοτικότητα λόγω της απλότητάς της και της ευκολίας μάθησής της. Η Python 3 έχει πολλές βελτιώσεις σε σχέση με την Python 2, κυκλοφόρησε αρχικά τον Δεκέμβριο του 2008 και έκτοτε έχει υιοθετηθεί ευρέως από την κοινότητα των προγραμματιστών.

4.1.2 Flask

Για την επικοινωνία του backend με το frontend χρησιμοποιήθηκε το Flask [53]. Το Flask είναι ένα ελαφρύ και ευέλικτο πλαίσιο εφαρμογών για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών σε γλώσσα προγραμματισμού Python. Αναπτύχθηκε από τον Armin Ronacher και κυκλοφόρησε για πρώτη φορά το 2010. Είναι γνωστό για την απλότητά του και την ευκολία στην εκμάθηση και χρήση, καθώς και για την επεκτασιμότητά του που του επιτρέπει να χρησιμοποιείται για τη δημιουργία εφαρμογών μικρής ή μεγάλης κλίμακας. Οι βασικές αρχές του είναι η απλότητα, η ευελιξία και η ελαφρότητα, καθώς προωθεί την ανάπτυξη διαδικτυακών εφαρμογών με έναν ελάχιστο αριθμό απαραίτητων συνιστωσών, επιτρέποντας στους προγραμματιστές να επιλέγουν και

να ενσωματώνουν τις δικές τους λύσεις ανάπτυξης. Η διεύθυνση όπου ο server του Flask τρέχει την εφαρμογή μας είναι `http://localhost:5000`.

4.1.3 CORS

Για την εξασφάλιση της ασφάλειας και της ιδιωτικότητας των δεδομένων στο διαδίκτυο χρησιμοποιήθηκε ο μηχανισμός Cross-Origin Resource Sharing (CORS). Το CORS είναι ένας μηχανισμός που επιτρέπει σε μια ιστοσελίδα να έχει πρόσβαση σε πόρους περιορισμένης πρόσβασης από ένα server σε ένα domain που είναι διαφορετικό από το domain που εξυπηρέτησε την ιστοσελίδα.

4.1.4 Flask-CORS

Το Flask-CORS είναι ένα extension του Flask, το οποίο υποστηρίζει τον μηχανισμό CORS στις εφαρμογές Flask, επιτρέποντας στους χρήστες να αιτούνται πόρους από διαφορετικά origins από αυτά που υποστηρίζεται η εφαρμογή Flask. Η χρήση του Flask-CORS επιτρέπει στους προγραμματιστές να διαμορφώνουν τις ρυθμίσεις CORS για τις διάφορες διαδρομές (routes) της εφαρμογής τους, προσδίδοντας ευελιξία και ασφάλεια στις αιτήσεις που γίνονται σε διαφορετικά domains. Με αυτόν τον τρόπο, το Flask-CORS διευκολύνει την ανάπτυξη διαδικτυακών εφαρμογών που αλληλοεπιδρούν απρόσκοπτα με διαφορετικά domains και πηγές δεδομένων στο διαδίκτυο.

4.1.5 GeoPy

Για τον υπολογισμό της απόστασης μεταξύ δύο σημείων χρησιμοποιήθηκε η συνάρτηση geodesic από την βιβλιοθήκη GeoPy [54]. Το GeoPy είναι μια βιβλιοθήκη της Python που παρέχει διάφορα εργαλεία για τον χειρισμό γεωγραφικών δεδομένων και την εκτέλεση γεωγραφικών υπολογισμών. Η συνάρτηση geodesic χρησιμοποιείται για τον υπολογισμό της γεωδαιτικής απόστασης μεταξύ δύο γεωγραφικών σημείων. Αυτό είναι χρήσιμο όταν

χρειάζεται να υπολογίσουμε την απόσταση μεταξύ δύο σημείων πάνω στην επιφάνεια της Γης, λαμβάνοντας υπόψη την καμπυλότητα της Γης.

4.1.6 JSON

Επίσης, εισάγεται η μορφή JSON στη γλώσσα προγραμματισμού Python. Το JSON (JavaScript Object Notation) είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν και για τις μηχανές να το αναλύσουν και να το παράγουν. Το JSON είναι μια μορφή κειμένου που αναπαριστά τα δεδομένα σε μορφή ζευγών "κλειδί-τιμή" και είναι εντελώς ανεξάρτητη από τη γλώσσα προγραμματισμού, αλλά χρησιμοποιεί συμβάσεις που είναι γνωστές στους προγραμματιστές των γλωσσών της οικογένειας C.

4.1.7 Υλοποίηση

Αφού εισάγουμε τις βιβλιοθήκες (βλ. Σχήμα 4.1), δημιουργούμε ένα instance του Flask και ενεργοποιούμε το CORS (βλ. Σχήμα 4.2). Ακολούθως, ανοίγουμε το αρχείο που περιέχει τα ηλιακά δεδομένα για κάθε φορτιστή ηλεκτρικών οχημάτων (βλ. Σχήμα 4.3). Στη συνέχεια, ανοίγουμε το αρχείο "nodes.txt" που περιέχει τις συντεταγμένες των σημείων κατά μήκος της διαδρομής που επιλέγει ο χρήστης (βλ. Σχήμα 4.4). Το αρχείο αυτό ανανεώνεται κάθε φορά που ο χρήστης επιλέγει μια διαδρομή. Παράδειγμα αυτού του αρχείου για τη διαδρομή από το αεροδρόμιο Λάρνακας προς το αεροδρόμιο Πάφου βλέπουμε στο Σχήμα 4.5, όπου για κάθε σημείο παρατηρούμε το id του, το γεωγραφικό μήκος του (longitude) και το γεωγραφικό πλάτος του (latitude).

```
from flask import Flask, jsonify, request
from flask_cors import CORS
from geopy.distance import geodesic
import json
```

Σχήμα 4. 1 Εισαγωγή βιβλιοθηκών

```
# Create a Flask application instance named 'app'
app = Flask(__name__)

# Enable Cross-Origin Resource Sharing (CORS) for specific routes
CORS(app, resources={r"/chargers": {"origins": "*"},
                     r"/nearby_chargers": {"origins": "*"},
                     r"/nodes": {"origins": "*"}})
```

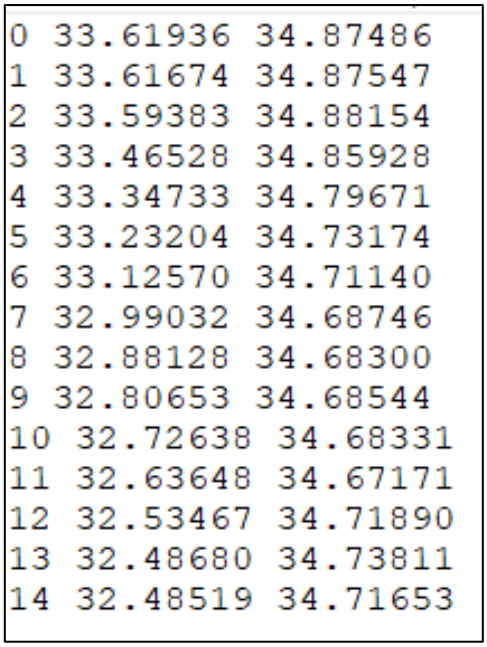
Σχήμα 4. 2 Δημιουργία instance του Flask και ενεργοποίηση του CORS

```
# Open the file "solarDataALL.txt" in read mode
fSolarData = open(r'solarDataALL.txt', "r")
# Read the contents of the "solarDataALL.txt" file and store them in fSolarData
fSolarData = fSolarData.readlines()
```

Σχήμα 4. 3 Άνοιγμα αρχείου ηλιακών δεδομένων

```
# Open the file "nodes.txt" in read mode
fNodes = open(r'nodes.txt', "r")
# Read the contents of the "nodes.txt" file and store them in fNodes
fNodes = fNodes.readlines()
```

Σχήμα 4. 4 Άνοιγμα αρχείου "nodes.txt"



Σχήμα 4. 5 Παράδειγμα αρχείου "nodes.txt"

Στον κώδικα προσδιορίσαμε τρία routes στο Flask για να διαχειρίζονται τα αιτήματα POST προς τα τρία endpoints ("/nodes", "/chargers", "/nearby_chargers"). Ένα REST API (Representational State Transfer Application Programming Interface) επιτρέπει την επικοινωνία μεταξύ διαφορετικών εφαρμογών ή συστημάτων μέσω του πρωτοκόλλου HTTP. Με ένα REST API, μπορούμε να πραγματοποιήσουμε διάφορες ενέργειες, όπως λήψη, δημιουργία, ενημέρωση ή διαγραφή δεδομένων, μέσω της αποστολής HTTP αιτημάτων σε συγκεκριμένες διευθύνσεις URL. Ένα HTTP POST αίτημα σε ένα REST API συνήθως χρησιμοποιείται για την αποστολή δεδομένων σε ένα server για επεξεργασία και τα δεδομένα του POST αιτήματος βρίσκονται στο σώμα του αιτήματος, σε μορφή JSON ή σε άλλη μορφή.

Στο Σχήμα 4.6 βλέπουμε το Flask route που διαχειρίζεται τα αιτήματα POST προς το endpoint "/nodes", όπου τα αιτήματα επεξεργάζονται από τις συναρτήσεις που ακολουθούν αμέσως μετά τη δήλωση του route. Συγκεκριμένα, εξάγονται τα δεδομένα JSON από το σώμα του αιτήματος (id, longitude, latitude των σημείων κατά μήκος της διαδρομής) και γράφονται στο αρχείο "nodes.txt".

```python
# Define a route for handling POST requests to '/nodes'
@app.route( rule: '/nodes', methods=['POST'])
def write_nodes():
    # Extract JSON data from the request
    data = request.get_json()

    if 'nodesList' in data:
        # Extract nodes list from the JSON data
        nodes_list = data['nodesList']

        # Write nodes to a text file
        write_to_file(nodes_list)

        return jsonify({'message': 'Data received and saved to file successfully'})

    return jsonify({'error': 'Invalid request'})


def write_to_file(data):
    # Open the file in write mode
    with open('nodes.txt', 'w') as file:
        # Write each item in the data list to the file
        for item in data:
            file.write(f"{item}\n")
```

Σχήμα 4. 6 Route για διαχείριση αιτημάτων POST προς το endpoint "/nodes"

Στο Σχήμα 4.7 φαίνεται το Flask route που διαχειρίζεται τα αιτήματα POST προς το endpoint "/chargers", όπου τα αιτήματα επεξεργάζονται από τη συνάρτηση που ακολουθεί αμέσως μετά τη δήλωση του route. Συγκεκριμένα, εξάγονται τα δεδομένα JSON από το σώμα του αιτήματος (συγκεκριμένη πόλη ή χώρα) και επιστρέφονται στον πελάτη σε μορφή JSON οι φορτιστές που βρίσκονται στη συγκεκριμένη πόλη ή χώρα. Απόσπασμα από το αρχείο "Cyprus.json", το οποίο περιέχει πληροφορίες για τους σταθμούς φόρτισης ηλεκτρικών οχημάτων στην Κύπρο φαίνεται στο Σχήμα 4.8.

```python
# Define a route for handling POST requests to '/chargers'
@app.route( rule: '/chargers', methods=['POST'])
def get_chargers():
    # Retrieve JSON data from the request
    data = request.get_json()

    # Extract the 'selected_value' from the JSON data
    selected_value = data.get('selected_value')

    # Open the JSON file corresponding to the selected value
    f = open(r'' + selected_value + '.json')
    # Load the JSON data from the file
    chargers = json.load(f)
    # Close the file after loading the data
    f.close()

    return jsonify(chargers)
```

Σχήμα 4. 7 Route για διαχείριση αιτημάτων POST προς το endpoint "/chargers"

```
{
    "access": 1,
    "address": "A1, Skarinou, Cyprus",
    "icon": "https://assets.plugshare.com/icons/Y.png",
    "icon_type": "Y",
    "id": 306199,
    "latitude": 34.817538,
    "longitude": 33.360926,
    "name": "McDonald's",
    "score": 10.0,
    "stations": [
        {
            "id": 710776,
            "outlets": [
                {
                    "connector": 13,
                    "id": 2140715,
                    "kilowatts": 50.0,
                    "power": 0,
                    "status": null
                },
                {
                    "connector": 3,
                    "id": 2142764,
                    "kilowatts": 50.0,
                    "power": 0,
                    "status": null
                },
                {
                    "connector": 7,
                    "id": 2142765,
                    "kilowatts": 43.0,
                    "power": 0,
                    "status": null
                }
            ]
        }
    ],
    "url": "http://api.plugshare.com/view/location/306199",
    "availabilityMIN": 77,
    "availabilityMAX": 100
},
```


Σχήμα 4. 8 Απόσπασμα από το αρχείο "Cyprus.json"

Στο Flask route που διαχειρίζεται τα αιτήματα POST προς το endpoint "/nearby_chargers" τα δεδομένα JSON από το σώμα του αιτήματος που εξάγονται είναι (βλ. Σχήμα 4.9):

1. το γεωγραφικό μήκος και πλάτος του σημείου, κοντά στο οποίο ψάχνουμε τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης,
2. το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή,

3. η διαθεσιμότητα του φορτιστή,
4. το βιώσιμο επίπεδο φόρτισης,
5. η πόλη ή η χώρα, στην οποία βρίσκονται οι φορτιστές
6. η ακτίνα μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων

Τα αιτήματα επεξεργάζονται από τις συναρτήσεις που ακολουθούν αμέσως μετά τη δήλωση του route, με αποτέλεσμα να επιστρέφονται στον πελάτη σε μορφή JSON μεταξύ άλλων οι κορυφαία καταταγμένοι, με βάση τον αλγόριθμο του EcoCharge, σταθμοί φόρτισης ηλεκτρικών οχημάτων.

```python
# Define a route for handling POST requests to '/nearby_chargers'
@app.route( rule: '/nearby_chargers', methods=['POST'])
def get_nearby_chargers():
    # Retrieve JSON data from the request
    data = request.get_json()

    lat = data['lat']
    lng = data['lng']
    derouting_cost = data['derouting_cost']
    charger_availability = data['charger_availability']
    sustainable_charging_level = data['sustainable_charging_level']
    selected_value = data['selected_value']
    radius = data['radius']
```

Σχήμα 4. 9 Απόσπασμα από το route για διαχείριση αιτημάτων POST προς το endpoint "/nearby_chargers"

Για να τρέξει το αρχείο "ecocharge_demo.py" πρέπει να εγκατασταθούν πρώτα οι απαραίτητες βιβλιοθήκες.

Οι απαραίτητες εντολές για να τρέξει το αρχείο "ecocharge_demo.py" είναι:

- `sudo apt update`
- `sudo apt install python3-pip`
- `sudo pip3 install Flask`
- `sudo pip3 install flask-cors`
- `sudo pip3 install geopy`
- `python3 ecocharge_demo.py`

4.2 Υλοποίηση για το frontend

Το αρχείο που περιέχει τον κώδικα για το frontend ονομάζεται "ecocharge_demo.html". Ο κώδικας για το frontend είναι γραμμένος σε HTML, CSS και JavaScript.

4.2.1 HTML

Η HTML (HyperText Markup Language) χρησιμοποιείται για τη δημιουργία και τον προσδιορισμό της δομής μιας ιστοσελίδας, όπως για παράδειγμα για τη δημιουργία διαφόρων στοιχείων όπως κείμενο, εικόνες, βίντεο, σύνδεσμοι και πίνακες. Η HTML αποτελείται από tags που περιβάλλουν το περιεχόμενο και παρέχουν πληροφορίες σχετικά με τη σημασία αυτού του περιεχομένου. Είναι η βασική γλώσσα για την κατασκευή ιστοσελίδων.

4.2.2 CSS

Η CSS (Cascading Style Sheets) χρησιμοποιείται για τον έλεγχο της εμφάνισης και της διάταξης του περιεχομένου που ορίζεται στην HTML. Με τη CSS, μπορούμε να καθορίσουμε το χρώμα, τη γραμματοσειρά, το μέγεθος και τη διάταξη των στοιχείων της ιστοσελίδας μας.

4.2.3 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία διαδραστικών και δυναμικών ιστοσελίδων. Μπορούμε να χρησιμοποιήσουμε τη JavaScript για να προσθέσουμε λειτουργίες όπως αλληλεπίδραση χρήστη, ανταπόκριση σε γεγονότα, δημιουργία δυναμικού περιεχομένου και πολλά άλλα. Είναι επίσης η γλώσσα πίσω από πολλές διαδικτυακές εφαρμογές που λειτουργούν στην πλευρά του πελάτη, όπως παιχνίδια και εφαρμογές διαδικτύου.

Αυτές οι τρεις γλώσσες συνήθως χρησιμοποιούνται μαζί για τη δημιουργία πλήρως λειτουργικών και ευέλικτων ιστοσελίδων. Η HTML ορίζει τη δομή της ιστοσελίδας, η CSS ελέγχει την εμφάνιση και τη διάταξη, ενώ η JavaScript προσθέτει διαδραστικότητα και λειτουργικότητα.

4.2.4 Leaflet

Για την ενσωμάτωση διαδραστικού χάρτη στην εφαρμογή μας χρησιμοποιήθηκε η Leaflet [55].

Η Leaflet είναι μια ανοικτού κώδικα JavaScript βιβλιοθήκη που προσφέρει ευέλικτες λύσεις για την ενσωμάτωση διαδραστικών χαρτών σε ιστοσελίδες και εφαρμογές. Αναπτύχθηκε από τον Volodymyr Agafonkin και κυκλοφόρησε για πρώτη φορά το 2011. Με μέγεθος μόλις περίπου 42 KB, παρέχει όλες τις λειτουργίες χαρτογράφησης που οι περισσότεροι προγραμματιστές χρειάζονται. Η Leaflet σχεδιάστηκε με έμφαση στην απλότητα, στην απόδοση και στην ευκολία χρήσης. Μπορεί επίσης να επεκταθεί με πολλά plugins και διαθέτει ένα API που είναι εύκολο να χρησιμοποιηθεί. Η ενεργή κοινότητά της και η εκτεταμένη τεκμηρίωσή της την καθιστούν μια κορυφαία επιλογή για προγραμματιστές που αναζητούν να ενσωματώσουν λειτουργίες χαρτογράφησης στις εφαρμογές τους.

Για να ενσωματώσουμε τα αρχεία CSS και JavaScript της Leaflet στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css"/>`
- `<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>`

4.2.5 Leaflet Routing Machine

Για την εύρεση του δρομολογίου μεταξύ δύο σημείων στο χάρτη χρησιμοποιήθηκε το Leaflet Routing Machine [56].

Το Leaflet Routing Machine είναι ένα plugin της βιβλιοθήκης Leaflet που προσφέρει προηγμένη λειτουργικότητα δρομολόγησης σε έναν χάρτη Leaflet για ιστοσελίδες και εφαρμογές. Επιτρέπει στους προγραμματιστές να προσθέτουν εύκολα δυνατότητες στις εφαρμογές τους όπως εύρεση της βέλτιστης διαδρομής με βάση την ποδηλασία, την οδήγηση, την πεζοπορία και άλλες επιλογές, εμφάνιση οδηγιών και σημείων ενδιαφέροντος στον χάρτη, καθώς επίσης και προσαρμογή του τρόπου εμφάνισης και των επιλογών δρομολόγησης.

Για να ενσωματώσουμε τα αρχεία CSS και JavaScript του Leaflet Routing Machine στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<link rel="stylesheet" href="https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.css"/>`
- `<script src="https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.js"></script>`

4.2.6 Leaflet.Locate

Για την εμφάνιση της τρέχουσας θέσης μας στο χάρτη χρησιμοποιήθηκε το Leaflet.Locate [57].

Το Leaflet.Locate είναι ένα plugin της βιβλιοθήκης Leaflet που προσφέρει δυνατότητες εντοπισμού της τρέχουσας θέσης του χρήστη σε έναν χάρτη. Έτσι, ο χρήστης μπορεί γρήγορα και εύκολα να βρίσκει τη θέση του στο χάρτη.

Για να ενσωματώσουμε τα αρχεία CSS και JavaScript του Leaflet.Locate στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol/dist/L.Control.Locate.min.css"/>`

- `<script src="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol/dist/L.Control.Locate.min.js" charset="utf-8"></script>`

4.2.7 Leaflet.markercluster

Για την ομαδοποίηση των φορτιστών που βρίσκονται κοντά μεταξύ τους χρησιμοποιήθηκε το Leaflet.markercluster [58].

Το Leaflet.markercluster είναι ένα plugin της βιβλιοθήκης Leaflet που παρέχει δυνατότητες ομαδοποίησης σημείων (markers) σε χάρτες Leaflet. Επιτρέπει στους προγραμματιστές να ομαδοποιήσουν σημεία που βρίσκονται κοντά μεταξύ τους, προσφέροντας έτσι μια καλύτερη οπτική αναπαράσταση των δεδομένων σε περιοχές με υψηλή πυκνότητα. Αυτό είναι ιδιαίτερα χρήσιμο όταν έχουμε μεγάλο αριθμό σημείων σε έναν χάρτη και θέλουμε να τα εμφανίσουμε με τρόπο που να μην καταστρέφει την αισθητική του χάρτη.

Για να ενσωματώσουμε τα αρχεία CSS και JavaScript του Leaflet.markercluster στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster/dist/MarkerCluster.css"/>`
- `<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster/dist/MarkerCluster.Default.css"/>`
- `<script src="https://unpkg.com/leaflet.markercluster/dist/leaflet.markercluster.js"></script>`

4.2.8 Bootstrap

Για την εμφάνιση και τη διάταξη του περιεχομένου της εφαρμογής μας χρησιμοποιήθηκε το Bootstrap 5.3.3.

Το Bootstrap [59] είναι το πιο δημοφιλές δωρεάν και ανοικτού κώδικα CSS πλαίσιο εργασίας για την ανάπτυξη responsive ιστοσελίδων και ιστοσελίδων με προτεραιότητα την κινητή συσκευή. Δημιουργήθηκε από τους προγραμματιστές του Twitter και κυκλοφόρησε για πρώτη φορά το 2011. Περιλαμβάνει πρότυπα σχεδίασης για τυπογραφία, φόρμες, κουμπιά, πλοήγηση και άλλα στοιχεία διεπαφής, τα οποία μπορούν να ενσωματωθούν εύκολα σε οποιοδήποτε ιστοσελίδα ή εφαρμογή για να βελτιωθεί η εμφάνιση και η λειτουργικότητα τους. Με τη χρήση του Bootstrap, οι προγραμματιστές μπορούν να αναπτύξουν γρήγορα και εύκολα επαγγελματικής ποιότητας ιστοσελίδες που προσαρμόζονται αυτόματα σε διάφορες συσκευές και μεγέθη οθονών.

Για να ενσωματώσουμε τα αρχεία CSS και JavaScript του Bootstrap στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">`
- `<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>`

4.2.9 JQuery

Για την ευκολότερη χρήση της JavaScript στην εφαρμογή μας χρησιμοποιήθηκε η JQuery 3.7.1.

Η jQuery [60] είναι μια lightweight, γρήγορη και "write less, do more" βιβλιοθήκη JavaScript. Παίρνει πολλές κοινές εργασίες που απαιτούν πολλές γραμμές κώδικα

JavaScript και τις αντικαθιστά με μεθόδους που μπορούμε να καλέσουμε με μία μόνο γραμμή κώδικα. Με τη βοήθειά της, οι προγραμματιστές μπορούν να δημιουργούν ιστοσελίδες με λιγότερο κώδικα και περισσότερη ευκολία. Από το 2006, η jQuery έχει γίνει ένα από τα πιο δημοφιλή εργαλεία για την ανάπτυξη ιστοσελίδων και εφαρμογών.
Για να ενσωματώσουμε το αρχείο JavaScript της JQuery στην εφαρμογή μας προσθέτουμε τα ακόλουθα στο "ecocharge_demo.html":

- `<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>`

4.2.10 Google Sign-In

Για να προσθέσουμε το κουμπί Google Sign-In στην εφαρμογή μας ακολουθούμε την εξής διαδικασία:

1. δημιουργούμε authorization credentials, με αποτέλεσμα να αποκτούμε ένα client ID (βλ. Σχήμα 4.10)
2. φορτώνουμε τη βιβλιοθήκη της πλατφόρμας Google στην εφαρμογή μας, γράφοντας τα ακόλουθα:
   ```
   <script src="https://apis.google.com/js/platform.js" async
   defer></script>
   ```
3. ορίζουμε το client ID που δημιουργήσαμε για την εφαρμογής μας, γράφοντας τα ακόλουθα:
   ```
   <meta name="google-signin-client_id"
   content="YOUR_CLIENT_ID.apps.googleusercontent.com">
   ```
4. προσθέτουμε το κουμπί Google Sign-In, γράφοντας τα ακόλουθα:
   ```
   <div class="g-signin2" data-onsuccess="onSignIn"></div>
   ```

```
Google Sign-in Client ID:

1. Sign in to Google Cloud Platform
   https://console.developers.google.com/apis/dashboard

2. Create a new project, or use an existing project if you already have one set up

3. Go to the Credentials page
   https://console.cloud.google.com/apis/credentials

4. Click OAuth consent screen
   User Type: External
   App name: EcoCharge
   Save

5. Click Create credentials > OAuth client ID
   Name:
   Authorized JavaScript origins: https://ecocharge-demo.cs.ucy.ac.cy
   Authorized redirect URIs:
   Save

6. Use your Client ID
```

Σχήμα 4. 10 Οδηγίες δημιουργίας authorization credentials

4.2.11 Mapbox

Χρησιμοποιήσαμε επίσης την Mapbox [61] για να μετατρέπει διευθύνσεις σε συντεταγμένες και αντίστροφα, καθώς επίσης και για να υποδεικνύει αυτόματα τις διευθύνσεις καθώς τις γράφουμε. Για να μπορέσουμε να χρησιμοποιήσουμε τις λειτουργίες της Mapbox στην εφαρμογή μας, έπρεπε πρώτα να δημιουργήσουμε ένα Mapbox Access Token, μέσω της σελίδας https://account.mapbox.com/access-tokens/.

Η Mapbox είναι μια πλατφόρμα δεδομένων τοποθεσίας για κινητές και διαδικτυακές εφαρμογές, η οποία ιδρύθηκε το 2010. Προσφέρει ένα ευέλικτο API που επιτρέπει στους προγραμματιστές να ενσωματώνουν διαφορετικούς τύπους χαρτών και λειτουργίες στις εφαρμογές τους, όπως δρομολόγηση, γεωκωδικοποίηση, εύρεση με τοποθεσία και πολλά άλλα.

4.2.12 Χάρτες

Ο χάρτης μας προέρχεται από την OpenStreetMap [62], μια δωρεάν γεωγραφική βάση δεδομένων που ενημερώνεται και συντηρείται από μια κοινότητα εθελοντών μέσω ανοικτής συνεργασίας. Η OpenStreetMap παρέχει δωρεάν χάρτες και γεωχωρικά δεδομένα που μπορούν να χρησιμοποιηθούν από οποιονδήποτε για οποιαδήποτε σκοπό.
Επίσης, ενσωματώσαμε και το χάρτη από το OpenTopoMap [63], ένα project που βασίζεται στην OpenStreetMap και παρέχει χάρτες που δείχνουν την ανάγλυφη διαμόρφωση του εδάφους.
Ακόμη, χρησιμοποιήσαμε γεωγραφικούς και τοπογραφικούς χάρτες από την Environmental Systems Research Institute [64], μια εταιρεία λογισμικού που εξειδικεύεται στα γεωγραφικά συστήματα πληροφοριών (GIS) και ιδρύθηκε το 1969.

4.2.13 Υλοποίηση

Αφού ενσωματώσαμε τους χάρτες στην εφαρμογή μας, προσθέσαμε την κλίμακα του χάρτη (βλ. Σχήμα 4.11). Ακολούθως, προσθέσαμε ένα κουμπί για την εμφάνιση της τρέχουσας θέσης του χρήστη στο χάρτη (βλ. Σχήμα 4.12).

```
// Add map scale
L.control.scale({
    position: "bottomright",
    maxWidth: 100,
    metric: true,
    imperial: true
}).addTo(map);
```

Σχήμα 4. 11 Προσθήκη κλίμακας χάρτη

```
// Show my location
L.control
    .locate({
        position: "bottomright",
        showCompass: "true",
        strings: {
            title: "Show my location"
        }
    })
    .addTo(map);
```

Σχήμα 4. 12 Προσθήκη κουμπιού για την εμφάνιση της τρέχουσας θέσης

Ο χρήστης δηλώνει την αρχή και το τέλος της διαδρομής του. Έτσι, προσθέτουμε event listeners στα inputs της αρχής και του τέλους (βλ. Σχήμα 4.13).

```
// Add an event listener to the 'input' event of the origin element
originInput.addEventListener('input', function () {
    updateAutocomplete(originInput.value, 'origin');
});

// Add an event listener to the 'input' event of the destination element
destinationInput.addEventListener('input', function () {
    updateAutocomplete(destinationInput.value, 'destination');
});
```

Σχήμα 4. 13 Προσθήκη event listeners στα inputs της αρχής και του τέλους της διαδρομής

Επίσης, ο χρήστης δηλώνει τη βαρύτητα που θέλει να έχουν τα εκτιμώμενα στοιχεία, δηλαδή το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης. Στο Σχήμα 4.14 φαίνεται ο τρόπος που βρίσκουμε το κόστος παρέκκλισης που επιλέγει ο χρήστης.

```
// Estimated Component: Derouting
var slider1 = document.getElementById("ec1");
var output1 = document.getElementById("range-value1");
output1.innerHTML = slider1.value;

slider1.oninput = function () {
    output1.innerHTML = this.value;
}
```

Σχήμα 4. 14 Τρόπος εύρεσης κόστους παρέκκλισης

Ακόμη, ο χρήστης επιλέγει την πόλη ή τη χώρα στην οποία βρίσκονται οι φορτιστές που ψάχνει. Για να προβάλλεται ο χάρτης στην περιοχή που επιλέγει ο χρήστης προσθέσαμε ένα event listener στην αλλαγή της επιλογής του (βλ. Σχήμα 4.15).

```
// Add an event listener to the 'change' event of the chargersLocation element
document.getElementById("chargersLocation").addEventListener('change', function () {
    var selectedValue = this.value;
    if (selectedValue === "All") {
        // Set default view when selected value is "All"
        map.setView([40, -105], 5);
    } else {
        // Call the geocode function with the selected value
        geocode(selectedValue, function (coordinates) {
            if (coordinates) {
                // Set the view of the map to the coordinates obtained from geocoding
                map.setView([coordinates[1], coordinates[0]], 8);
            } else {
                console.error('Unable to geocode the selected name.');
            }
        });
    }
});
```

Σχήμα 4. 15 Προσθήκη event listener στην αλλαγή τοποθεσίας

Ο χρήστης έχει την δυνατότητα να βλέπει ή όχι όλους τους φορτιστές που βρίσκονται στην πόλη ή στη χώρα που επέλεξε. Αυτό γίνεται με την προσθήκη ενός event listener στην αλλαγή της κατάστασης του checkbox (checked, unchecked) (βλ. Σχήμα 4.16).

```
// Add an event listener to the 'change' event of the checkbox element
document.getElementById('checkbox').addEventListener('change', function () {
    if (this.checked) {
        // Checkbox is checked, show all chargers
        showAllChargers();
    } else {
        // Checkbox is unchecked
        clearChargers();
    }
});
```

Σχήμα 4. 16 Προσθήκη event listener στην αλλαγή κατάστασης του checkbox

Στο Σχήμα 4.17 φαίνεται ο τρόπος με τον οποίο ορίζουμε την τρέχουσα θέση του χρήστη ως την αρχή της διαδρομής του.

```
// Make current location as origin address
map.on('locationfound', function (e) {
    // Update the value of the origin input field with the user's current location coordinates
    const currentLocation = `${e.latlng.lat}, ${e.latlng.lng}`;

    // Call reverse geocoding to get the address of the current location
    reverseGeocode(e.latlng.lat, e.latlng.lng, function (address) {
        if (address) {
            originInput.value = address;
        } else {
            originInput.value = currentLocation;
        }
    });
});
```

Σχήμα 4. 17 Ορισμός τρέχουσας θέσης ως αρχή της διαδρομής

Όταν εμφανίζεται στο χάρτη η διαδρομή που επιλέγει ο χρήστης, εμφανίζεται και ένα κίτρινο ταξί, το οποίο μπορούμε να μετακινήσουμε για να βρούμε τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης κοντά στο σημείο αυτό. Αυτό επιτυγχάνεται με την προσθήκη ενός event listener, ο οποίος ανανεώνει τις συντεταγμένες όταν το ταξί μετακινείται (βλ. Σχήμα 4.18).

```
// TaxiMarker
var taxiMarker = L.marker([originCoords[1], originCoords[0]], {
    icon: taxiIcon,
    draggable: true // Make the marker draggable
}).addTo(map);

// Event listener to update coordinates when marker is dragged
taxiMarker.on('dragend', function (event) {
    var marker = event.target;
    var position = marker.getLatLng();
    console.log('New Marker Position:', position);
    // Call fetchNearbyChargers function with updated coordinates
    fetchNearbyChargers(
        position.lat,
        position.lng,
        slider1.value / 100,
        slider2.value / 100,
        slider3.value / 100,
        selected_value,
        radius);
});

// Show an alert on the map instructing the user to move the cab
taxiMarker.bindPopup("Drag the cab to find nearby chargers.").openPopup();
```

Σχήμα 4. 18 Προσθήκη event listener που ανανεώνει τις συντεταγμένες όταν το ταξί μετακινείται

Στο Σχήμα 4.19 παρουσιάζεται η συνάρτηση που αποστέλλει τη λίστα με τους κόμβους με αίτημα POST στο endpoint "/nodes". Η λίστα (id, longitude, latitude των σημείων κατά μήκος της διαδρομής) βρίσκεται στο σώμα του αιτήματος σε μορφή JSON.

```
// Function to send the nodes list
no usages
function sendNodes(nodesList) {
    fetch('http://localhost:5000/nodes', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({nodesList: nodesList}),
    })
        .then(response => response.json())
        .then(data => {
            console.log('Data:', data);
        })
        .catch(error => console.error('Error sending coordinates to server:', error));
}
```

Σχήμα 4. 19 Συνάρτηση που αποστέλλει λίστα με κόμβους στο endpoint "/nodes"

Στο Σχήμα 4.20 βλέπουμε τη συνάρτηση που αποστέλλει τα δεδομένα JSON με αίτημα POST στο endpoint "/nearby_chargers". Τα δεδομένα που βρίσκονται στο σώμα του αιτήματος σε μορφή JSON είναι:

1. το γεωγραφικό μήκος και πλάτος του σημείου, κοντά στο οποίο ψάχνουμε τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης,
2. το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή,
3. η διαθεσιμότητα του φορτιστή,
4. το βιώσιμο επίπεδο φόρτισης,
5. η πόλη ή η χώρα, στην οποία βρίσκονται οι φορτιστές
6. η ακτίνα μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων

Αφού γίνει η επεξεργασία του αιτήματος στο backend, επιστρέφονται στον πελάτη σε μορφή JSON μεταξύ άλλων οι κορυφαία καταταγμένοι, με βάση τον αλγόριθμο του EcoCharge, σταθμοί φόρτισης ηλεκτρικών οχημάτων, με αποτέλεσμα η συγκεκριμένη συνάρτηση να τοποθετεί στο χάρτη markers στις θέσεις των συγκεκριμένων σταθμών φόρτισης.

```
// Function to fetch nearby chargers
no usages
function fetchNearbyChargers(lat, lng, derouting_cost, charger_availability, sustainable_charging_level,
                             selected_value, radius) {

    fetch('http://localhost:5000/nearby_chargers', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            lat: lat,
            lng: lng,
            derouting_cost: derouting_cost,
            charger_availability: charger_availability,
            sustainable_charging_level: sustainable_charging_level,
            selected_value: selected_value,
            radius: radius
        }),
    })
        .then(response => response.json())
        .then(data => {
            console.log('Data:', data);

            // Remove only markers having the class 'custom-marker'
            map.eachLayer(function (layer) {
                if (layer instanceof L.Marker && layer.options.icon.options.className === 'custom-marker') {
                    map.removeLayer(layer);
                }
            });

            // Add markers for the nearby chargers
            addMarkers(data.nearby_chargers);
        })
        .catch(error => console.error('Error sending coordinates to server:', error));
}
```

Σχήμα 4. 20 Συνάρτηση που αποστέλλει δεδομένα στο endpoint "/nearby_chargers"

Στο Σχήμα 4.21 παρουσιάζεται η συνάρτηση που αποστέλλει την πόλη ή τη χώρα στην οποία βρίσκονται οι φορτιστές με αίτημα POST στο endpoint "/chargers". Η πόλη ή η χώρα βρίσκεται στο σώμα του αιτήματος σε μορφή JSON. Στον πελάτη επιστρέφονται σε μορφή JSON οι φορτιστές που βρίσκονται στη συγκεκριμένη πόλη ή χώρα, με αποτέλεσμα η συγκεκριμένη συνάρτηση να τοποθετεί ομαδοποιημένα markers στο χάρτη στις θέσεις των σταθμών φόρτισης που βρίσκονται κοντά μεταξύ τους.

```
// Function to show all chargers
no usages
function showAllChargers() {
    var selected_value = document.getElementById("chargersLocation").value;

    fetch('http://localhost:5000/chargers', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({selected_value: selected_value}),
    })
        .then(response => response.json())
        .then(chargers => {
            var markerCluster = L.markerClusterGroup(); // Create a marker cluster group

            chargers.forEach(charger => {
                var marker = L.marker([charger.latitude, charger.longitude], {icon: greenMarkerIcon})
                    .bindPopup(charger.name);
                markerCluster.addLayer(marker); // Add marker to the cluster group
            });

            map.addLayer(markerCluster); // Add the marker cluster group to the map
        })
        .catch((error) => console.error('Error:', error));
}
```

Σχήμα 4. 21 Συνάρτηση που αποστέλλει πόλη ή χώρα στο endpoint "/chargers"

Ο χρήστης έχει επίσης την δυνατότητα με right-click σε ένα σημείο στο χάρτη να βλέπει τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης ηλεκτρικών οχημάτων που βρίσκονται κοντά στο σημείο αυτό (βλ. Σχήμα 4.22).

```
// Function to open a custom popup on right-click and show the nearby chargers
no usages
function showCustomPopup(lat, lng) {
    var selected_value = document.getElementById("chargersLocation").value;
    // console.log(selected_value);

    var radius = parseInt(document.getElementById("radius").value);
    // console.log(radius);

    // Create a custom popup
    var popupContent = `<p>Latitude: ${lat.toFixed(4)}</p><p>Longitude: ${lng.toFixed(4)}</p>
                    <p><a href="#" id="showNearbyChargers">Show nearby chargers</a></p>`;
    var popup = L.popup()
        .setLatLng([lat, lng])
        .setContent(popupContent)
        .openOn(map);

    // Attach a click event listener to the "Show nearby chargers" link
    document.getElementById("showNearbyChargers").addEventListener('click', function (event) {
        event.preventDefault(); // Prevent the default link behavior

        fetchNearbyChargers(
            lat,
            lng,
            slider1.value / 100,
            slider2.value / 100,
            slider3.value / 100,
            selected_value,
            radius
        );
        // Close the popup after clicking the link
        map.closePopup(popup);
    });
}
```

Σχήμα 4. 22 Συνάρτηση που εμφανίζει τους φορτιστές με right-click σε ένα σημείο

Κεφάλαιο 5

Γραφικό περιβάλλον χρήστη του συστήματος EcoCharge

Σε αυτό το κεφάλαιο παρουσιάζεται το γραφικό περιβάλλον χρήστη του συστήματος EcoCharge.

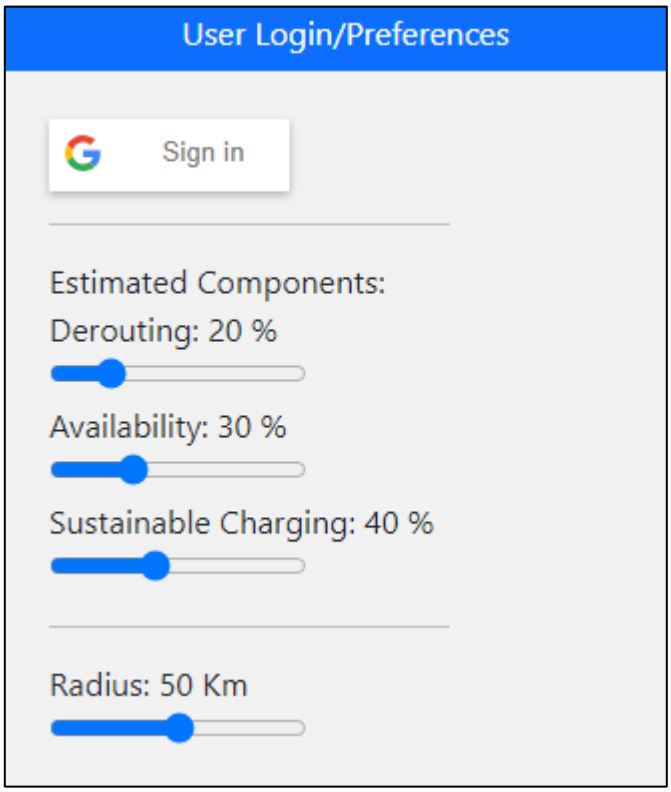
5.1 Panels

Το γραφικό περιβάλλον χρήστη του συστήματος EcoCharge περιλαμβάνει δύο panels. Στο Σχήμα 5.1 φαίνεται το γραφικό περιβάλλον με ανοικτά τα panels και στο Σχήμα 5.2 με κλειστά τα panels, στην ιστοσελίδα `https://ecocharge-demo.cs.ucy.ac.cy/`.

Σχήμα 5. 1 Γραφικό περιβάλλον χρήστη με ανοικτά τα panels

Σχήμα 5. 2 Γραφικό περιβάλλον χρήστη με κλειστά τα panels

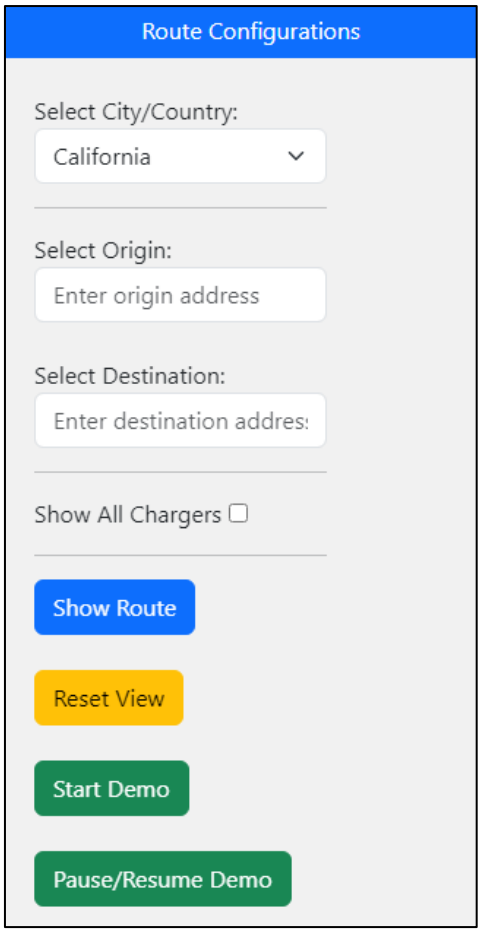
Το δεξί panel (βλ. Σχήμα 5.3) σχετίζεται με τη σύνδεση με το λογαριασμό Google και τις προτιμήσεις του χρήστη. Συγκεκριμένα περιέχει το κουμπί για να συνδεθεί ο χρήστης με το λογαριασμό που έχει στην Google, τρία range sliders για τα εκτιμώμενα στοιχεία, δηλαδή το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, τη διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης και ένα range slider για την ακτίνα μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων.



Σχήμα 5. 3 Δεξί panel

Το αριστερό panel (βλ. Σχήμα 5.4) σχετίζεται με τις ρυθμίσεις της διαδρομής. Συγκεκριμένα, ο χρήστης επιλέγει την πόλη/χώρα, την αρχή και το τέλος της διαδρομής και έχει τη δυνατότητα να δει όλους τους φορτιστές στη συγκεκριμένη πόλη/χώρα. Επίσης, με το πάτημα του κουμπιού "Show Route" εμφανίζεται η διαδρομή, ενώ με το πάτημα του "Reset View" ο χάρτης προβάλλεται με βάση τις συντεταγμένες της αρχής της διαδρομής. Ακόμη, αν ο χρήστης θέλει να δει το demo με την πορεία ενός κίτρινου ταξί από την αρχή

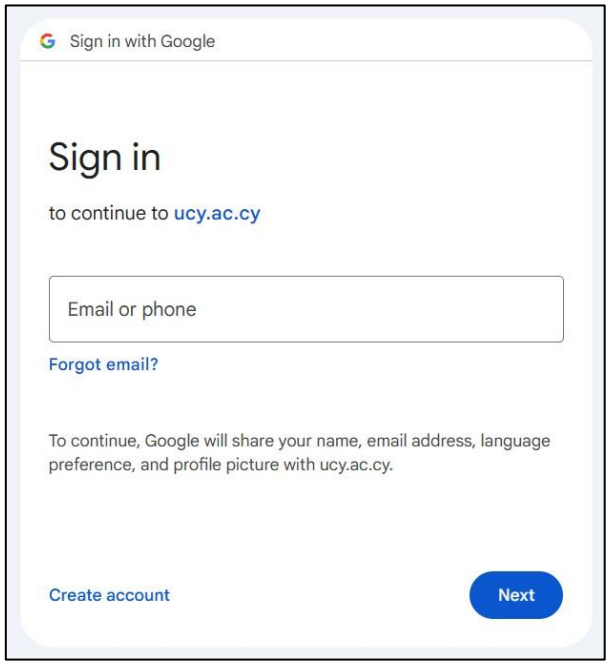
στο τέλος της διαδρομής, όπου εμφανίζονται σε κάθε σημείο κατά μήκος της διαδρομής οι κορυφαία καταταγμένοι σταθμοί φόρτισης, μπορεί να πατήσει το κουμπί "Start Demo". Αν θέλει να σταματήσει ή να ξαναρχίσει το demo μπορεί να πατήσει το "Pause/Resume Demo".



Σχήμα 5. 4 Αριστερό panel

5.2 Σύνδεση με το λογαριασμό της Google

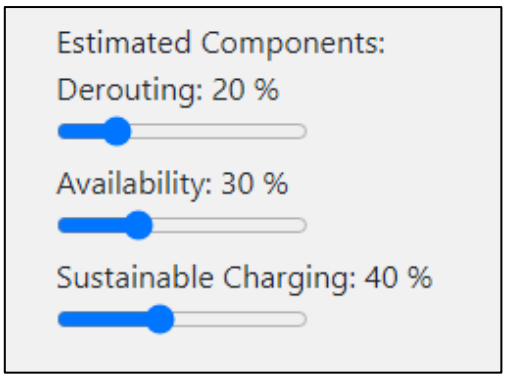
Αρχικά, ο χρήστης συνδέεται με το λογαριασμό που έχει στην Google (βλ. Σχήμα 5.5).



Σχήμα 5. 5 Σύνδεση με το λογαριασμό της Google

5.3 Βαρύτητα εκτιμώμενων στοιχείων

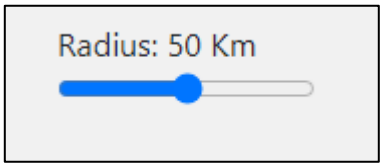
Στη συνέχεια, ο χρήστης δηλώνει τη βαρύτητα που θέλει να έχουν τα εκτιμώμενα στοιχεία, δηλαδή το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης. Στο Σχήμα 5.6 βλέπουμε ένα παράδειγμα, όπου το κόστος παρέκκλισης επιλέγεται 20%, η διαθεσιμότητα 30% και το βιώσιμο επίπεδο φόρτισης 40%.



Σχήμα 5. 6 Παράδειγμα βαρύτητας εκτιμώμενων στοιχείων

5.4 Επιλογή ακτίνας

Ακολούθως, ο χρήστης επιλέγει την ακτίνα μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων. Στο παράδειγμα του Σχήματος 5.7 επιλέγεται ακτίνα 50 km.

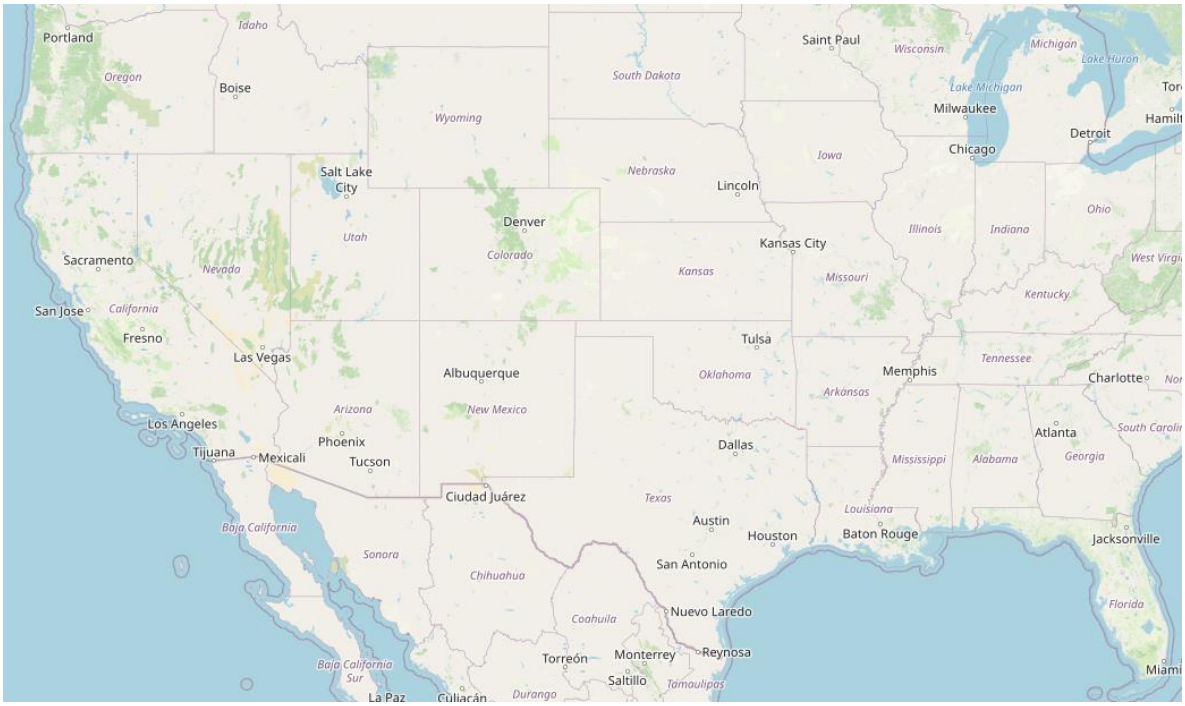


Σχήμα 5. 7 Παράδειγμα επιλογής ακτίνας

5.5 Επιλογή χάρτη

Ο χρήστης μπορεί να επιλέξει τον χάρτη που επιθυμεί από τους διαθέσιμους γεωγραφικούς και τοπογραφικούς χάρτες που υπάρχουν. Οι διαθέσιμοι χάρτες είναι οι ακόλουθοι:

1. γεωγραφικός χάρτης από OpenStreetMap (βλ. Σχήμα 5.8)
2. τοπογραφικός χάρτης από OpenTopoMap (βλ. Σχήμα 5.9)
3. γεωγραφικός χάρτης από ESRI (βλ. Σχήμα 5.10)
4. τοπογραφικός χάρτης από ESRI (βλ. Σχήμα 5.11)



Σχήμα 5. 8 Γεωγραφικός χάρτης από OpenStreetMap

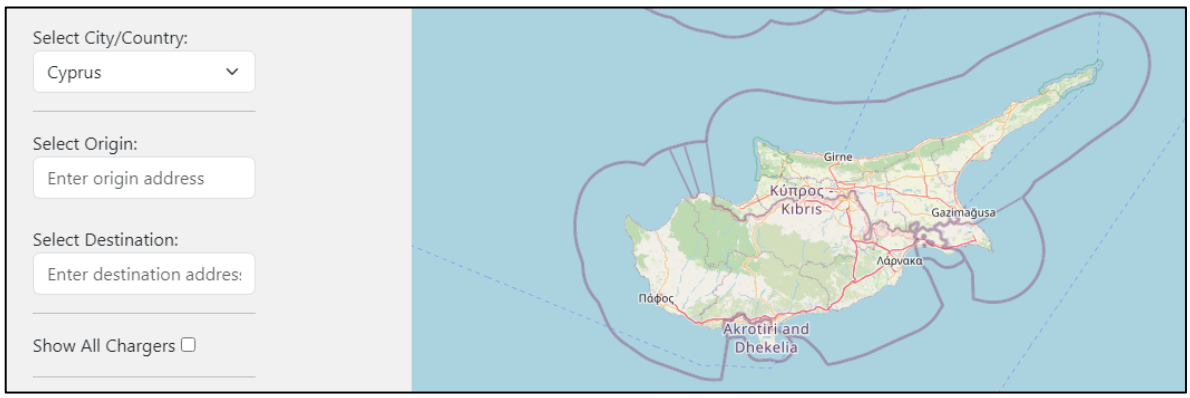
Σχήμα 5. 9 Τοπογραφικός χάρτης από OpenTopoMap

Σχήμα 5. 10 Γεωγραφικός χάρτης από ESRI

Σχήμα 5. 11 Τοπογραφικός χάρτης από ESRI

5.6 Επιλογή πόλης ή χώρας

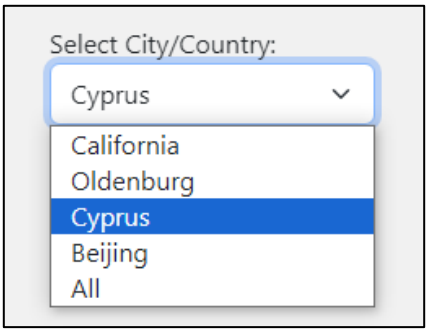
Μετά, ο χρήστης επιλέγει την πόλη ή τη χώρα, στην οποία βρίσκονται οι φορτιστές που ψάχνει για να φορτίσει το ηλεκτρικό του όχημα. Μόλις επιλέξει, ο χάρτης προβάλλεται στην περιοχή αυτή (βλ. Σχήμα 5.12).



Σχήμα 5. 12 Προβολή χάρτη στην περιοχή που επιλέγει ο χρήστης

Οι επιλογές που έχει ο χρήστης για πόλεις/χώρες είναι (βλ. Σχήμα 5.13):

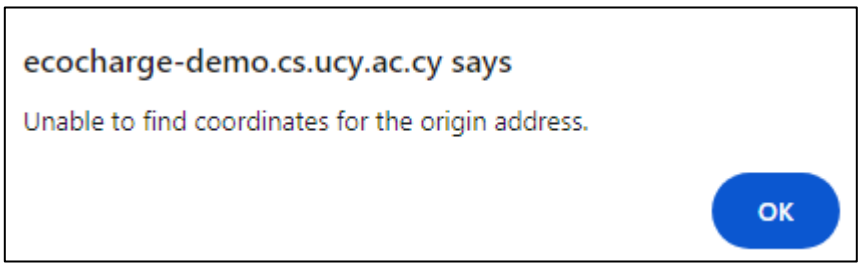
1. California
2. Oldenburg
3. Κύπρος
4. Beijing
5. Όλες



Σχήμα 5. 13 Διαθέσιμες πόλεις/χώρες

5.7 Επιλογή αρχής της διαδρομής

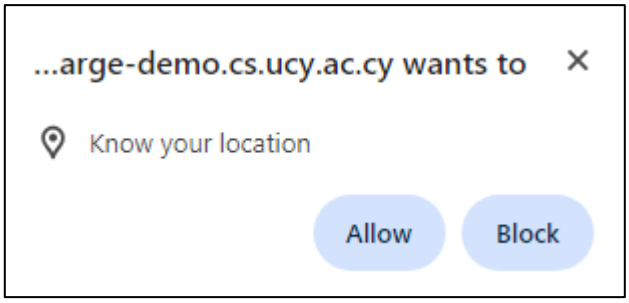
Στη συνέχεια, ο χρήστης επιλέγει την αρχή της διαδρομής του. Σε περίπτωση που δεν επιλέξει την αρχή και προσπαθήσει να δει την διαδρομή του, εμφανίζεται μήνυμα λάθους για μη επιλογή αρχής της διαδρομής (βλ. Σχήμα 5.14).



Σχήμα 5. 14 Μήνυμα λάθους για μη επιλογή αρχής της διαδρομής

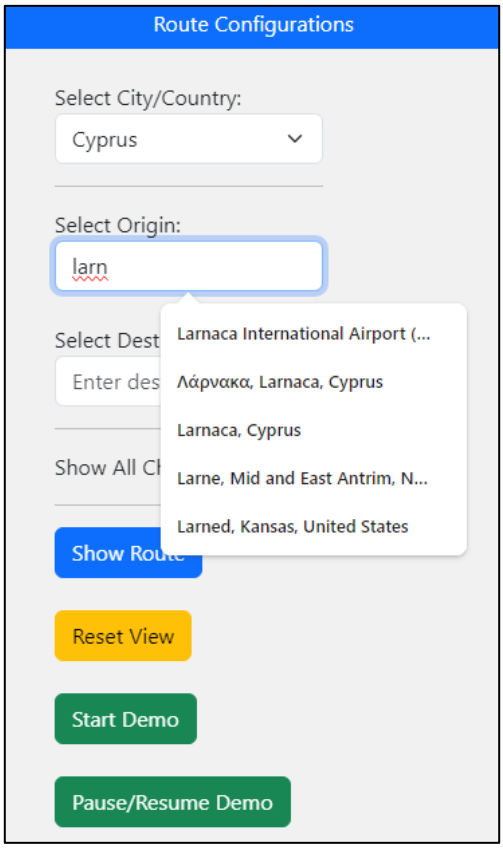
Ο χρήστης έχει την δυνατότητα να ορίσει την τρέχουσα θέση του ως την αρχή της διαδρομής του πατώντας το κουμπί του Σχήματος 5.15 και επιτρέποντας στο σύστημά μας να μάθει την θέση του (βλ. Σχήμα 5.16).

Σχήμα 5. 15 Κουμπί εμφάνισης τρέχουσας θέσης



Σχήμα 5. 16 Μήνυμα για να επιτραπεί η γνώση της τρέχουσα θέσης

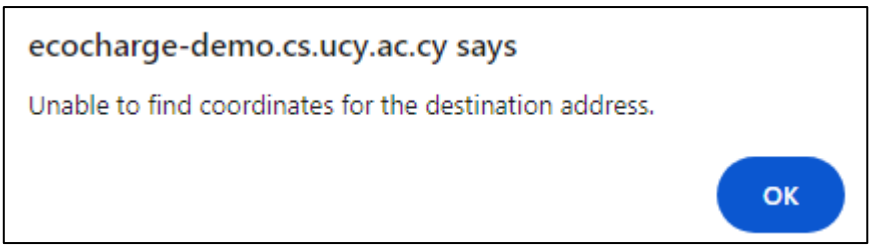
Έστω ότι ο χρήστης θέλει να επιλέξει ως αρχή της διαδρομής του το αεροδρόμιο Λάρνακας (LCA). Μόλις αρχίσει να γράφει, οι διευθύνσεις υποδεικνύονται αυτόματα, όπως φαίνεται στο Σχήμα 5.17.



Σχήμα 5. 17 Αυτόματη υπόδειξη διευθύνσεων για αρχή διαδρομής

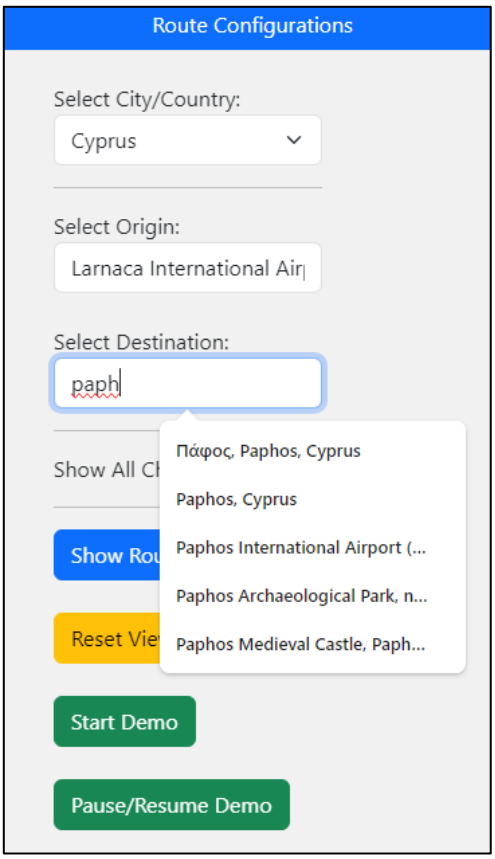
5.8 Επιλογή τέλους της διαδρομής

Ακολούθως, ο χρήστης επιλέγει το τέλος της διαδρομής του. Σε περίπτωση που δεν επιλέξει τέλος και προσπαθήσει να δει την διαδρομή του, εμφανίζεται μήνυμα λάθους για μη επιλογή τέλους της διαδρομής (βλ. Σχήμα 5.18).



Σχήμα 5. 18 Μήνυμα λάθους για μη επιλογή τέλους της διαδρομής

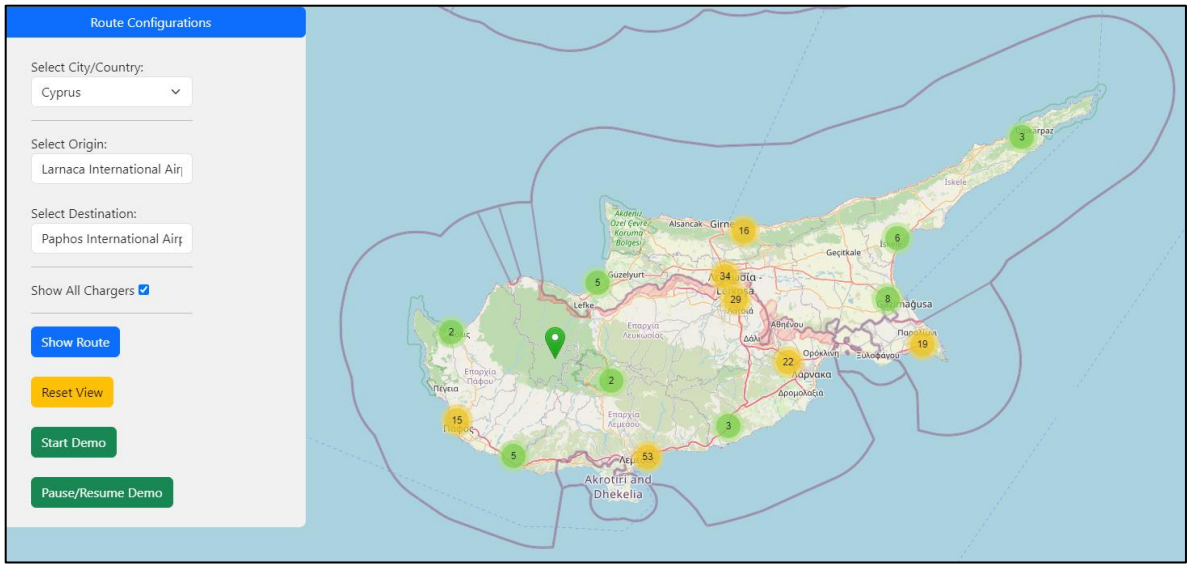
Έστω ότι ο χρήστης θέλει να επιλέξει ως τέλος της διαδρομής του το αεροδρόμιο Πάφου (PFO). Μόλις αρχίσει να γράφει, οι διευθύνσεις υποδεικνύονται αυτόματα, όπως φαίνεται στο Σχήμα 5.19.



Σχήμα 5. 19 Αυτόματη υπόδειξη διευθύνσεων για τέλος διαδρομής

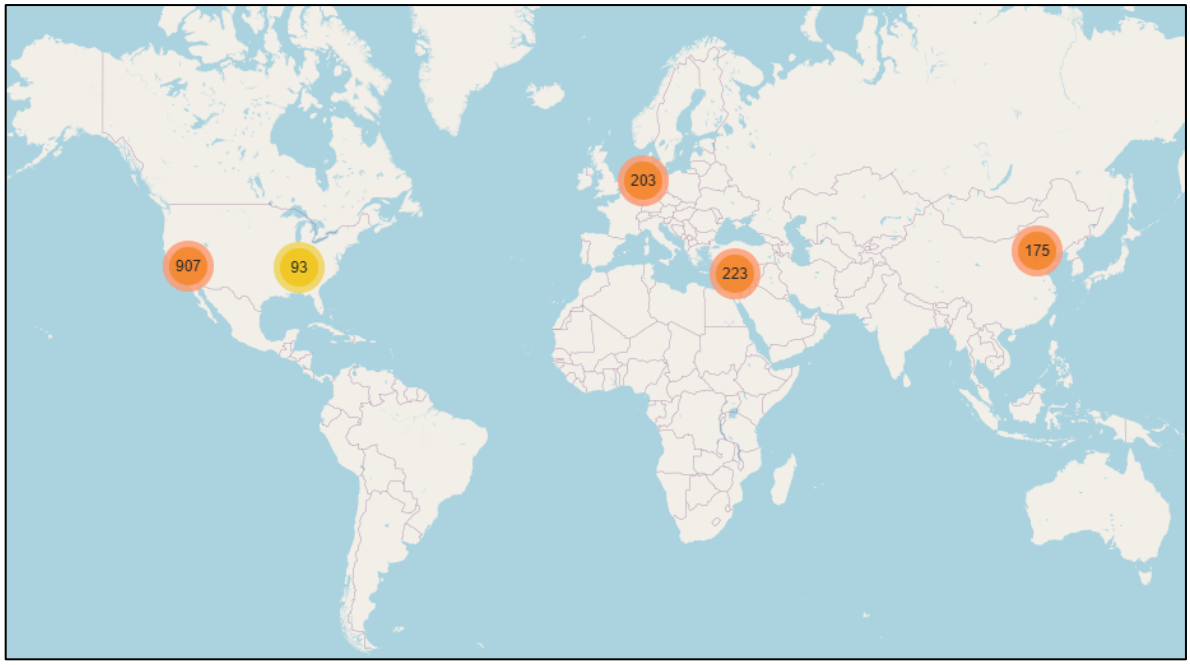
5.9 Εμφάνιση όλων των φορτιστών στην πόλη/χώρα

Ο χρήστης έχει την δυνατότητα να δει όλους τους φορτιστές στη συγκεκριμένη πόλη/χώρα που έχει επιλέξει, όταν τσεκάρει το checkbox για "Show All Chargers". Οι φορτιστές που βρίσκονται κοντά μεταξύ τους ομαδοποιούνται, με τρόπο που να μην καταστρέφει την αισθητική του χάρτη, όπως φαίνεται στο Σχήμα 5.20.



Σχήμα 5. 20 Εμφάνιση ομαδοποιημένων φορτιστών στην Κύπρο

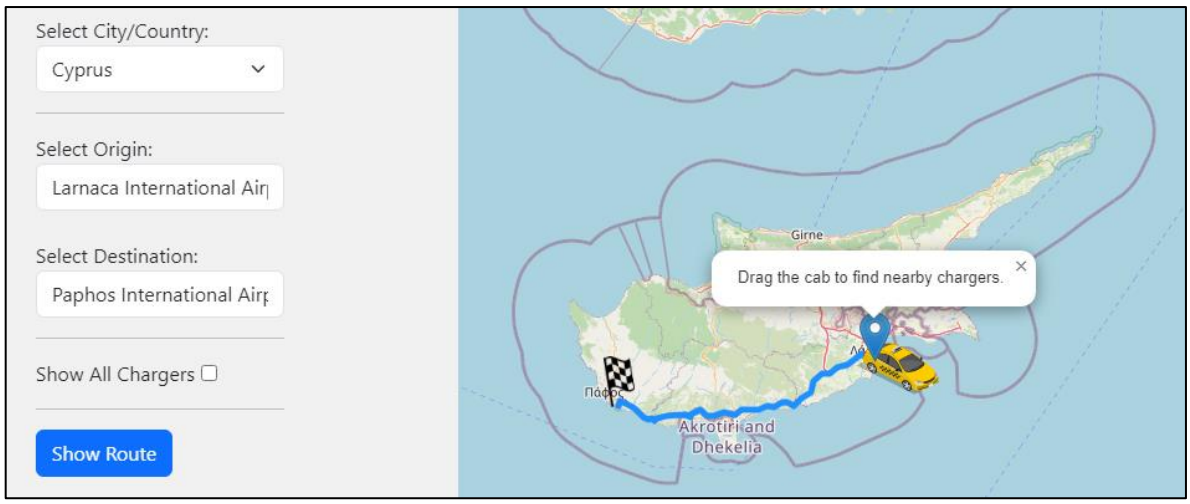
Όπως φαίνεται στο Σχήμα 5.21 είναι διαθέσιμοι 1601 συνολικά φορτιστές ηλεκτρικών οχημάτων, 175 στο Beijing, 203 στο Oldenburg, 223 στην Κύπρο, 907 στην περιοχή της California και 93 διάσπαρτοι στην υπόλοιπη Αμερική, οι οποίοι συλλέχθηκαν από την PlugShare.



Σχήμα 5. 21 Διαθέσιμοι φορτιστές ηλεκτρικών οχημάτων ανά περιοχή

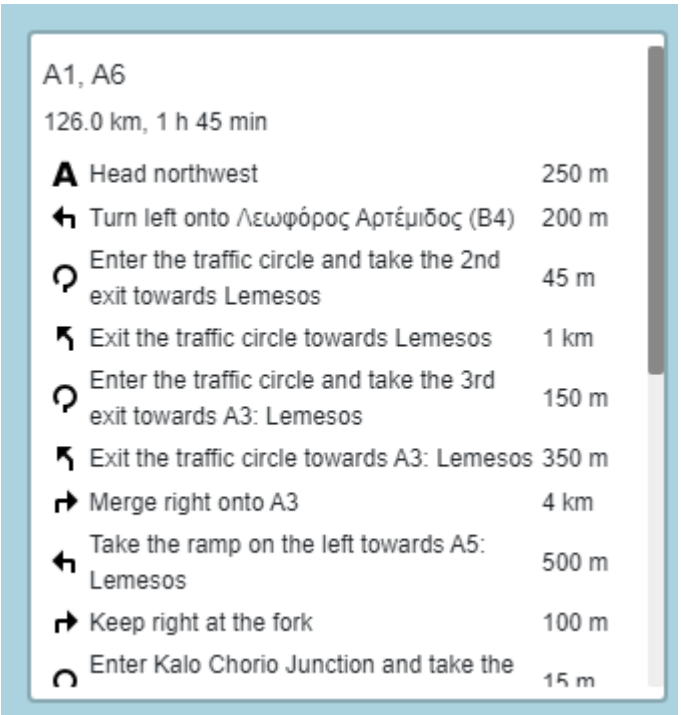
5.10 Εμφάνιση διαδρομής

Στο Σχήμα 5.22 φαίνεται η προτεινόμενη διαδρομή με μπλε γραμμή, όταν ο χρήστης πατήσει το κουμπί "Show Route". Επίσης, βλέπουμε στην αρχή της διαδρομής ένα μπλε marker και ένα κίτρινο ταξί και στο τέλος της διαδρομής μια σημαία. Ακόμη, εμφανίζεται και ένα μήνυμα που προτρέπει τον χρήστη να μετακινήσει το ταξί για να βρει τους κορυφαία καταταγμένους σταθμούς φόρτισης που βρίσκονται κοντά στη θέση του ταξί.



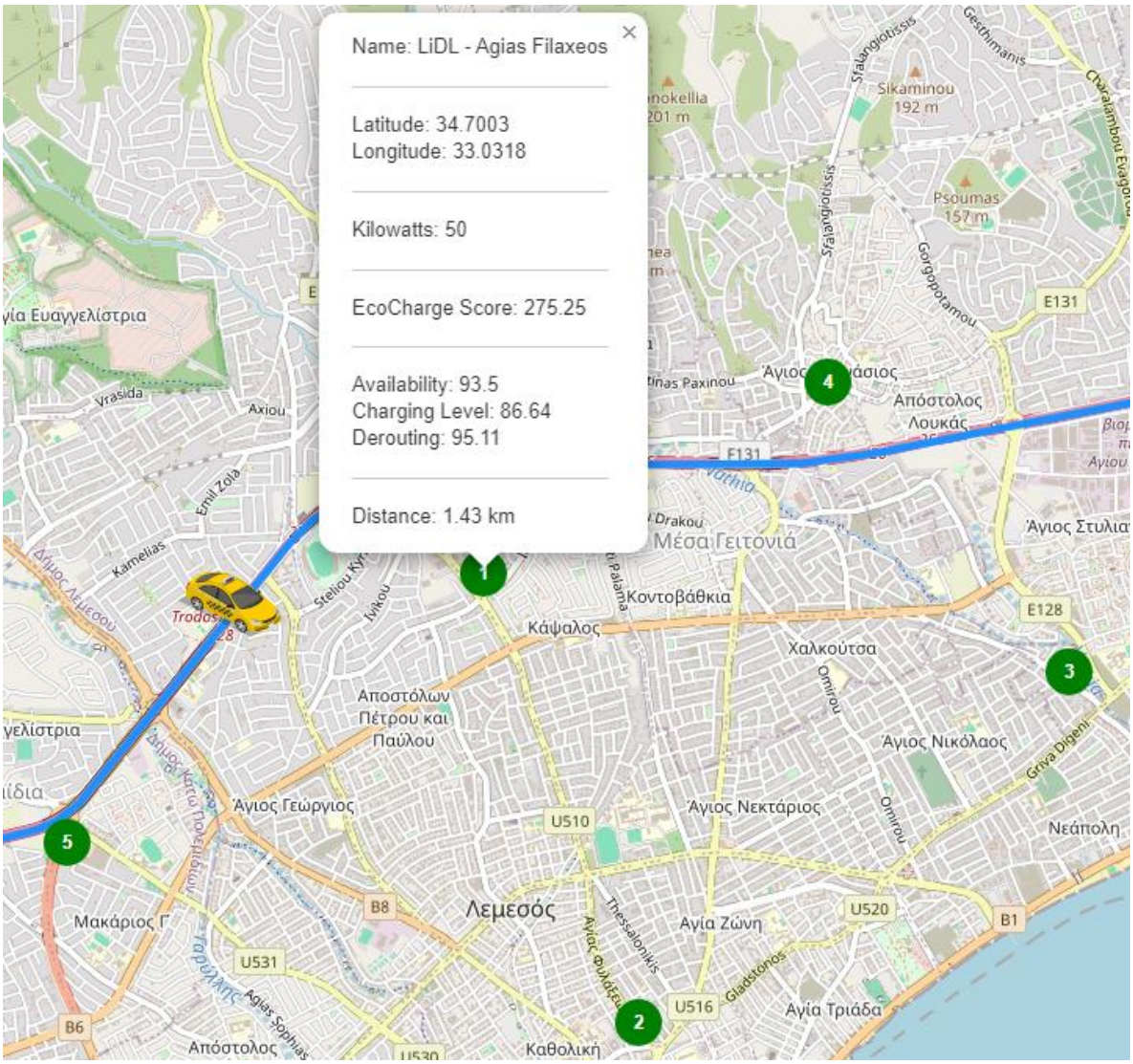
Σχήμα 5. 22 Εμφάνιση επιλεγμένης διαδρομής

Επίσης, όταν ο χρήστης πατήσει το κουμπί "Show Route" εμφανίζεται και ένα παράθυρο με πληροφορίες της διαδρομής (βλ. Σχήμα 5.23).



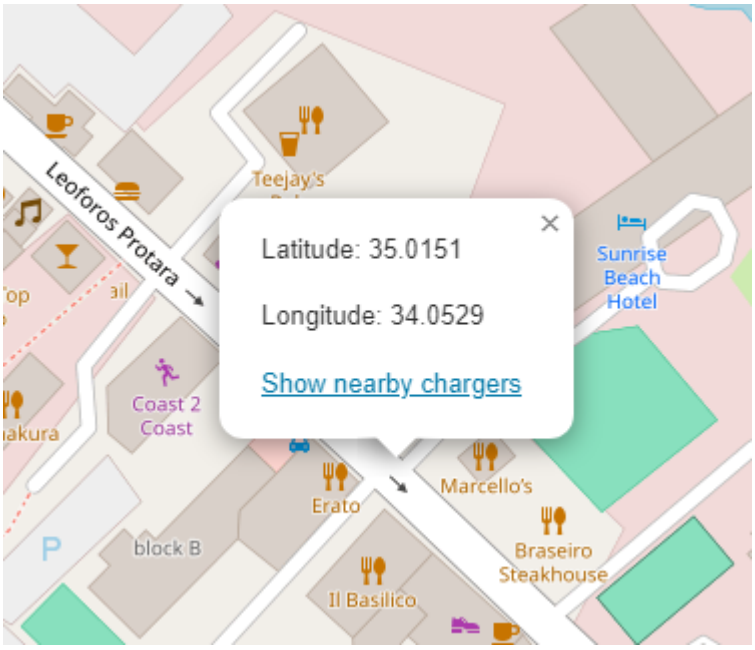
Σχήμα 5. 23 Παράθυρο πληροφοριών διαδρομής

Στο Σχήμα 5.24 βλέπουμε τους κορυφαία καταταγμένους σταθμούς φόρτισης που βρίσκονται κοντά στη θέση του ταξί, όταν το μετακινήσουμε. Ακόμη, όταν πατήσουμε πάνω στον πράσινο marker με την ένδειξη 1 βλέπουμε και πληροφορίες για τον σταθμό φόρτισης που κατατάχθηκε πρώτος, όπως για παράδειγμα το όνομα του, τις συντεταγμένες του, τα Kilowatts του και την απόσταση του από το ταξί.

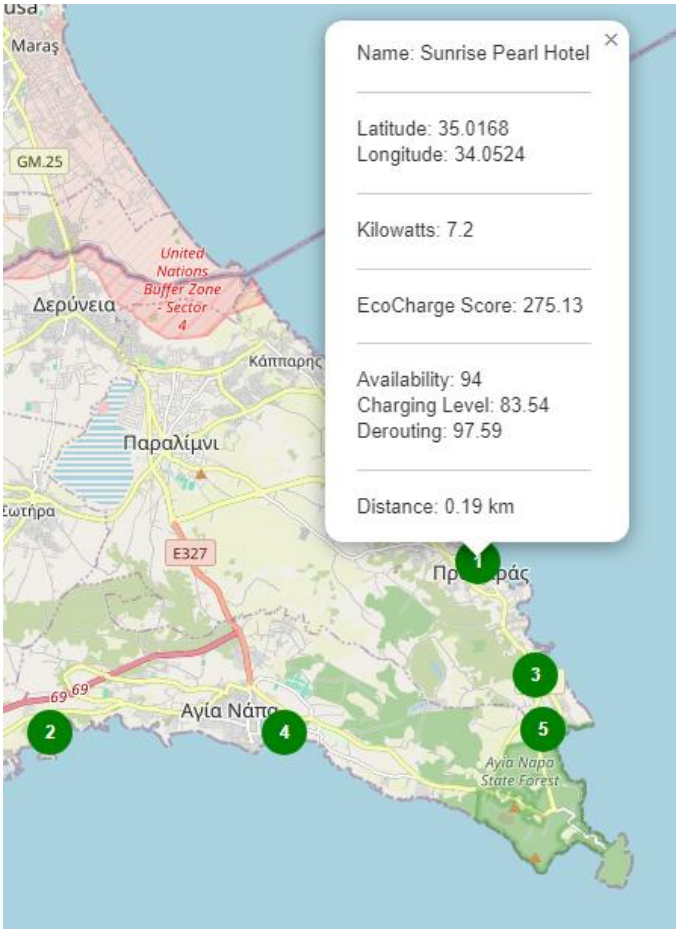


Σχήμα 5. 24 Παράδειγμα εμφάνισης κορυφαία καταταγμένων σταθμών φόρτισης, με μετακίνηση ταξί

Ο χρήστης έχει επίσης την δυνατότητα με right-click σε ένα σημείο στο χάρτη (βλ. Σχήμα 5.25) να βλέπει τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης ηλεκτρικών οχημάτων που βρίσκονται κοντά στο σημείο αυτό (βλ. Σχήμα 5.26).



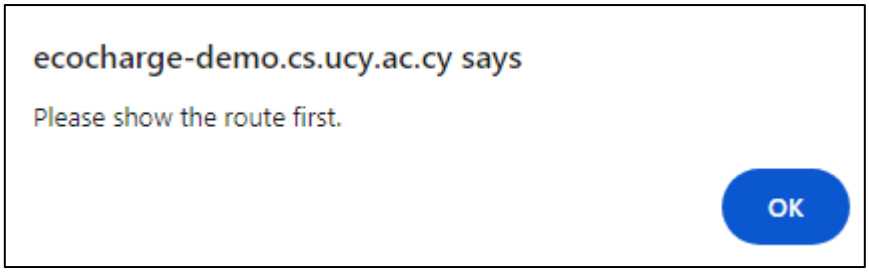
Σχήμα 5. 25 Right-click σε ένα σημείο στο χάρτη



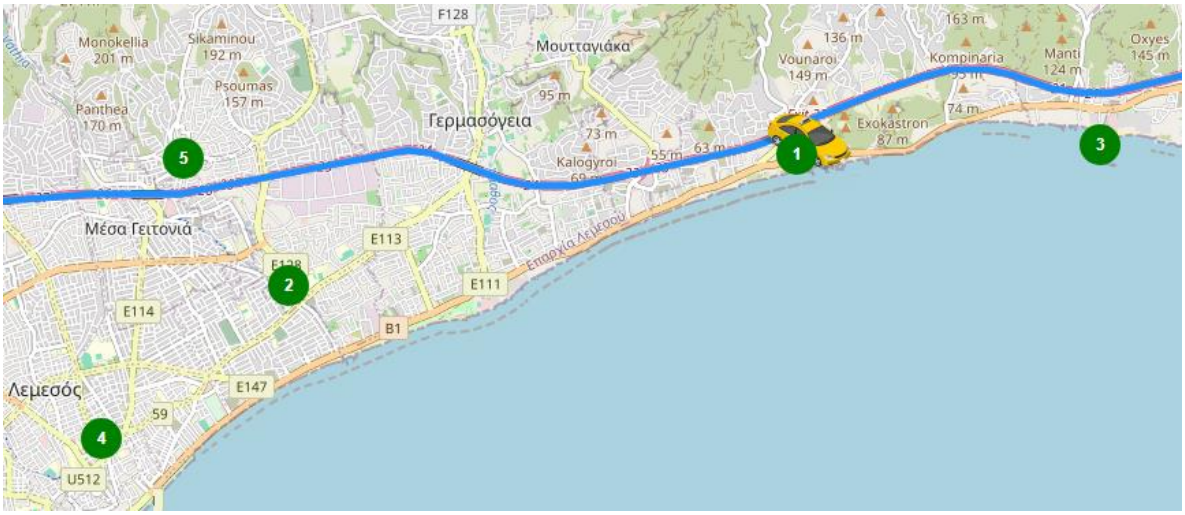
Σχήμα 5. 26 Παράδειγμα εμφάνισης κορυφαία καταταγμένων σταθμών φόρτισης, με right-click

5.11 Demo

Ακολούθως, αν ο χρήστης θέλει να δει το demo με την πορεία ενός κίτρινου ταξί από την αρχή στο τέλος της διαδρομής, όπου εμφανίζονται σε κάθε σημείο κατά μήκος της διαδρομής οι κορυφαία καταταγμένοι σταθμοί φόρτισης, μπορεί να πατήσει το κουμπί "Start Demo". Σε περίπτωση που δεν εμφανίστηκε η διαδρομή και προσπαθήσει ο χρήστης να τρέξει το demo, εμφανίζεται μήνυμα λάθους για μη εμφάνιση της διαδρομής (βλ. Σχήμα 5.27). Παράδειγμα εμφάνισης των κορυφαία καταταγμένων σταθμών φόρτισης κατά τη διάρκεια του demo βλέπουμε στο Σχήμα 5.28. Αν ο χρήστης θέλει να σταματήσει ή να ξαναρχίσει το demo μπορεί να πατήσει το κουμπί "Pause/Resume Demo".



Σχήμα 5. 27 Μήνυμα λάθους για μη εμφάνιση της διαδρομής



Σχήμα 5. 28 Παράδειγμα εμφάνισης κορυφαία καταταγμένων σταθμών φόρτισης κατά τη διάρκεια του demo

Κεφάλαιο 6

Πειραματική μεθοδολογία και αποτίμηση

Σε αυτό το κεφάλαιο παρουσιάζεται η μεθοδολογία και η αποτίμηση των πειραμάτων μας.

6.1 Χαρακτηριστικά εικονικής μηχανής

Η εικονική μηχανή (Virtual Machine) που χρησιμοποιήθηκε για την διεξαγωγή των πειραμάτων μας έχει τα εξής χαρακτηριστικά:

- Αριθμό CPUs: 2
- Συχνότητα CPU: 2.10 GHz
- Αρχιτεκτονική CPU: x86_64
- Μοντέλο CPU: Intel® Xeon® Silver 4208 CPU @ 2.10 GHz
- Μνήμη: 2.05 GB
- Αποθήκευση: 27.08 GB

- Λειτουργικό σύστημα: Ubuntu 22.04.4 LTS
- Πυρήνας: Linux 6.5.0-28-generic
- Hypervisor: VMware

6.2 Πειραματικές διαδρομές

Οι διαδρομές που χρησιμοποιήθηκαν στα πειράματά μας είναι οι ακόλουθες:

1. Oldenburg

- Αρχή: Varel, Kreis Friesland, Lower Saxony, Germany
- Τέλος: Bremen, Free Hanseatic City of Bremen, Germany

2. California

- Αρχή: Ceres, California, United States
- Τέλος: Selma, California, United States

3. Cyprus

- Αρχή: Larnaca International Airport (LCA)
- Τέλος: Paphos International Airport (PFO)

4. Beijing

- Αρχή: Beijing Capital Int'l Airport
- Τέλος: Cangzhou Shi, Hebei, China

Η επιλογή των συγκεκριμένων διαδρομών έγινε με βάση τους διαθέσιμους σταθμούς φόρτισης (ύπαρξη αρκετών φορτιστών κατά μήκος της διαδρομής) και τον αριθμό των σημείων κατά μήκος των διαδρομών (κάθε διαδρομή περιλαμβάνει 15 σημεία).

6.3 Χρόνος εκτέλεσης CPU και βαθμός βιωσιμότητας για διαφορετικές ακτίνες

Στο πείραμα αυτό υπολογίζουμε τον χρόνο εκτέλεσης CPU (βλ. Πίνακα 6.1) και τον βαθμό βιωσιμότητας (%) του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km, μέσα στις οποίες ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων, για τις τέσσερις διαδρομές. Ο βαθμός βιωσιμότητας είναι ουσιαστικά το κανονικοποιημένο άθροισμα των scores των προτεινόμενων σταθμών φόρτισης (βλ. Πίνακα 6.2). Το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης είναι 100%. Το Q (απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής) είναι 5 km.

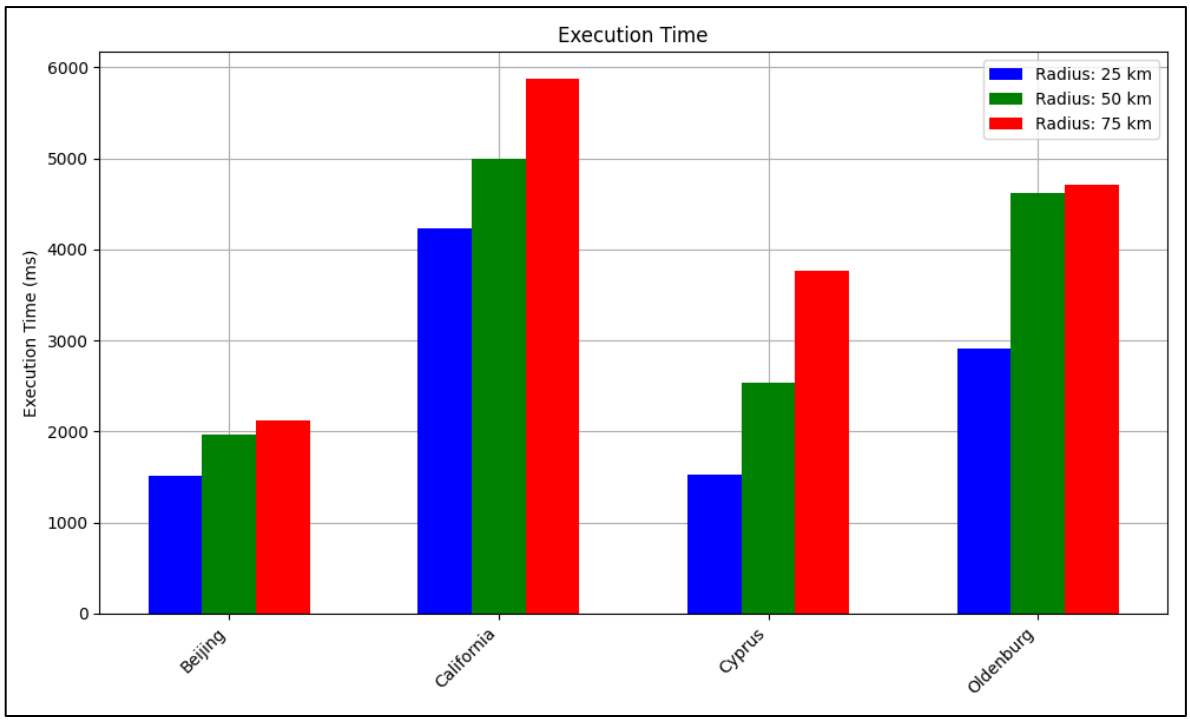
Χρόνος εκτέλεσης (ms)	Ακτίνα (km)	Πόλη/χώρα
1531.90	25	Cyprus
2541.85	50	Cyprus
3762.31	75	Cyprus
4234.76	25	California
5000.35	50	California
5876.41	75	California
2912.86	25	Oldenburg
4614.48	50	Oldenburg
4715.75	75	Oldenburg
1512.82	25	Beijing
1964.22	50	Beijing
2125.34	75	Beijing

Πίνακας 6. 1 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για διαφορετικές ακτίνες

Score	Ακτίνα (km)	Πόλη/χώρα
16795.04	25	Cyprus
18014.11	50	Cyprus
18402.02	75	Cyprus
12411.78	25	California
14824.02	50	California
15457.04	75	California
17838.03	25	Oldenburg
18792.02	50	Oldenburg
18914.12	75	Oldenburg
8841.54	25	Beijing
10762.82	50	Beijing
12805.93	75	Beijing

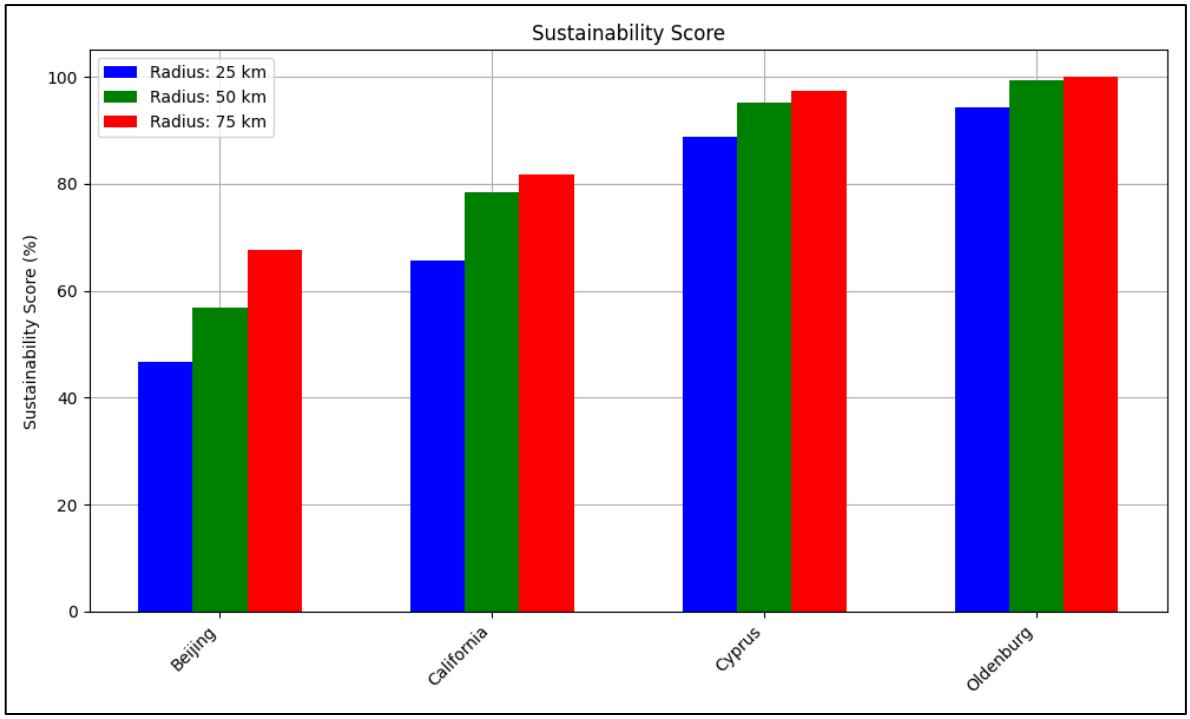
Πίνακας 6. 2 Scores αλγόριθμου EcoCharge για διαφορετικές ακτίνες

Η γραφική παράσταση του χρόνου εκτέλεσης του αλγόριθμου EcoCharge για ακτίνες 25 km, 50 km και 75 km για τις τέσσερις διαδρομές φαίνεται στο Σχήμα 6.1, ενώ στο Σχήμα 6.2 βλέπουμε την αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας (%) του αλγόριθμου.



Σχήμα 6. 1 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για διαφορετικές ακτίνες

Παρατηρώντας το Σχήμα 6.1 βλέπουμε ότι και στις τέσσερις διαδρομές όταν αυξάνεται η ακτίνα αυξάνεται και ο χρόνος εκτέλεσης. Αυτό είναι απολύτως λογικό αφού όσο μεγαλώνει η ακτίνα, μέσα στην οποία ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων, τόσο μεγαλώνει και ο χώρος αναζήτησης των φορτιστών. Επίσης, οι μεγαλύτεροι χρόνοι εμφανίζονται στην California, αφού στην περιοχή αυτή υπάρχουν οι περισσότεροι φορτιστές που διαθέτουμε (907) και οι μικρότεροι χρόνοι στο Beijing, λόγω του ότι εκεί υπάρχουν οι λιγότεροι φορτιστές που διαθέτουμε (175).



Σχήμα 6. 2 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικές ακτίνες

Παρατηρώντας το Σχήμα 6.2 βλέπουμε ότι και στις τέσσερις διαδρομές όταν αυξάνεται η ακτίνα αυξάνεται και ο βαθμός βιωσιμότητας. Αυτό είναι απολύτως λογικό αφού όσο μεγαλώνει η ακτίνα, μέσα στην οποία ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων, τόσο μεγαλώνει και ο χώρος αναζήτησης των φορτιστών, με αποτέλεσμα στο μεγαλύτερο χώρο αναζήτησης να υπάρχουν φορτιστές με μεγαλύτερο score. Επίσης, οι μεγαλύτεροι βαθμοί βιωσιμότητας εμφανίζονται στο Oldenburg και οι μικρότεροι στο Beijing, λόγω του ότι σε κάποια σημεία κατά μήκος της διαδρομής στο Beijing δεν υπήρχαν καθόλου φορτιστές ή υπήρχαν ένας ή δύο μέσα στην ακτίνα αναζήτησης.

6.4 Χρόνος εκτέλεσης CPU για φόρτωση φορτιστών από διαφορετικά αρχεία

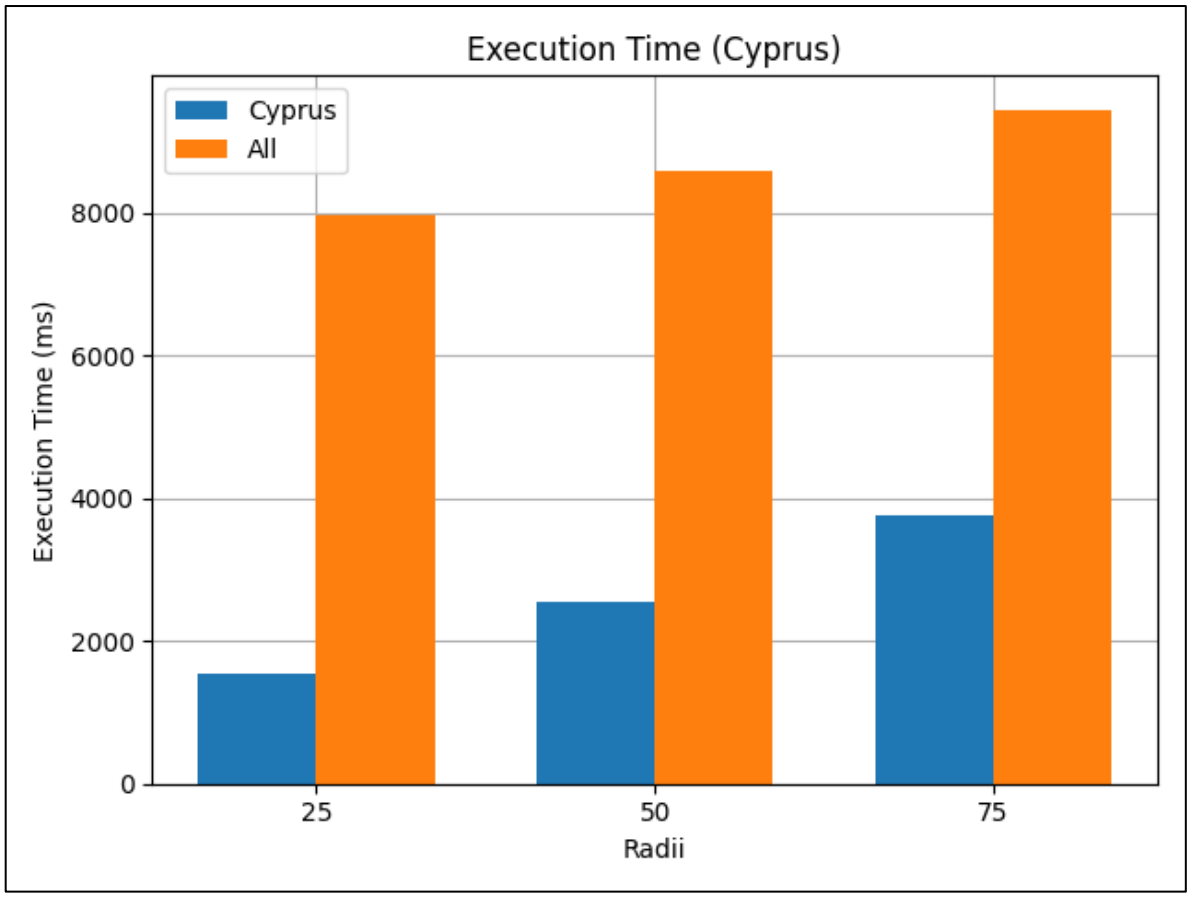
Στο πείραμα αυτό υπολογίζουμε τον χρόνο εκτέλεσης CPU του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km, μέσα στις οποίες ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων, για τις τέσσερις διαδρομές, όπου η φόρτωση των φορτιστών γίνεται από δύο διαφορετικά αρχεία. Το πρώτο είναι το αρχείο που περιέχει πληροφορίες για τους σταθμούς φόρτισης της συγκεκριμένης πόλης/χώρας και το δεύτερο περιέχει πληροφορίες για όλους τους σταθμούς που έχουμε στην διάθεσή μας (All.json). Το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης είναι 100%. Το Q (απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής) είναι 5 km.

Στον Πίνακα 6.3 φαίνεται ο χρόνος εκτέλεσης του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km όταν οι φορτιστές φορτώνονται από δύο διαφορετικά αρχεία, Cyprus.json και All.json. Στον Πίνακα 6.4 φαίνεται ο χρόνος εκτέλεσης του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km όταν οι φορτιστές φορτώνονται από δύο διαφορετικά αρχεία, California.json και All.json. Στον Πίνακα 6.5 φαίνεται ο χρόνος εκτέλεσης του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km όταν οι φορτιστές φορτώνονται από δύο διαφορετικά αρχεία, Oldenburg.json και All.json. Στον Πίνακα 6.6 φαίνεται ο χρόνος εκτέλεσης του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km όταν οι φορτιστές φορτώνονται από δύο διαφορετικά αρχεία, Beijing.json και All.json.

Χρόνος εκτέλεσης (ms)	Ακτίνα (km)	Αρχείο json
1531.90	25	Cyprus
2541.85	50	Cyprus
3762.31	75	Cyprus
7978.17	25	All
8594.44	50	All
9451.25	75	All

Πίνακας 6. 3 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Cyprus και All

Η αντίστοιχη γραφική παράσταση φαίνεται στο Σχήμα 6.3.

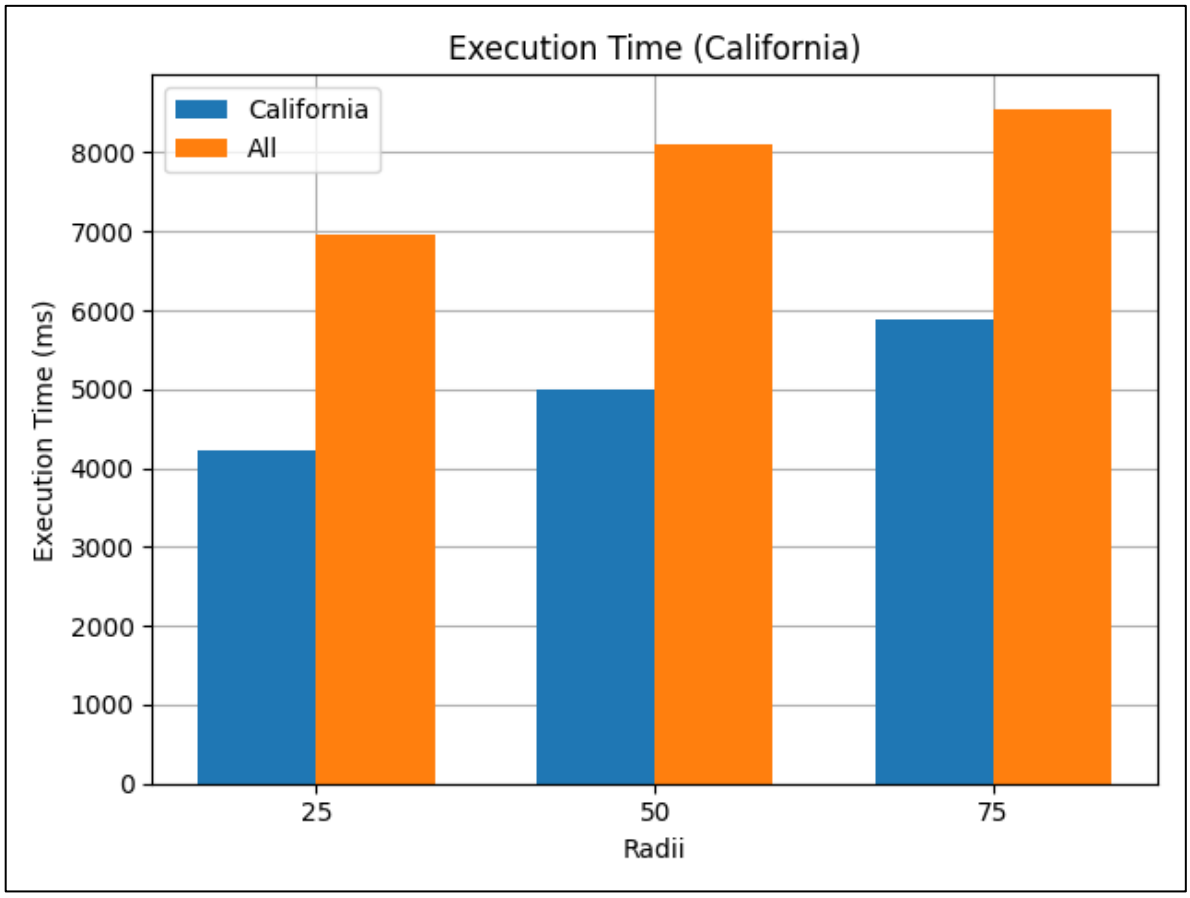


Σχήμα 6. 3 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Cyprus και All

Χρόνος εκτέλεσης (ms)	Ακτίνα (km)	Αρχείο json
4234.76	25	California
5000.35	50	California
5876.41	75	California
6968.31	25	All
8106.56	50	All
8556.71	75	All

Πίνακας 6. 4 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία California και All

Η αντίστοιχη γραφική παράσταση φαίνεται στο Σχήμα 6.4.

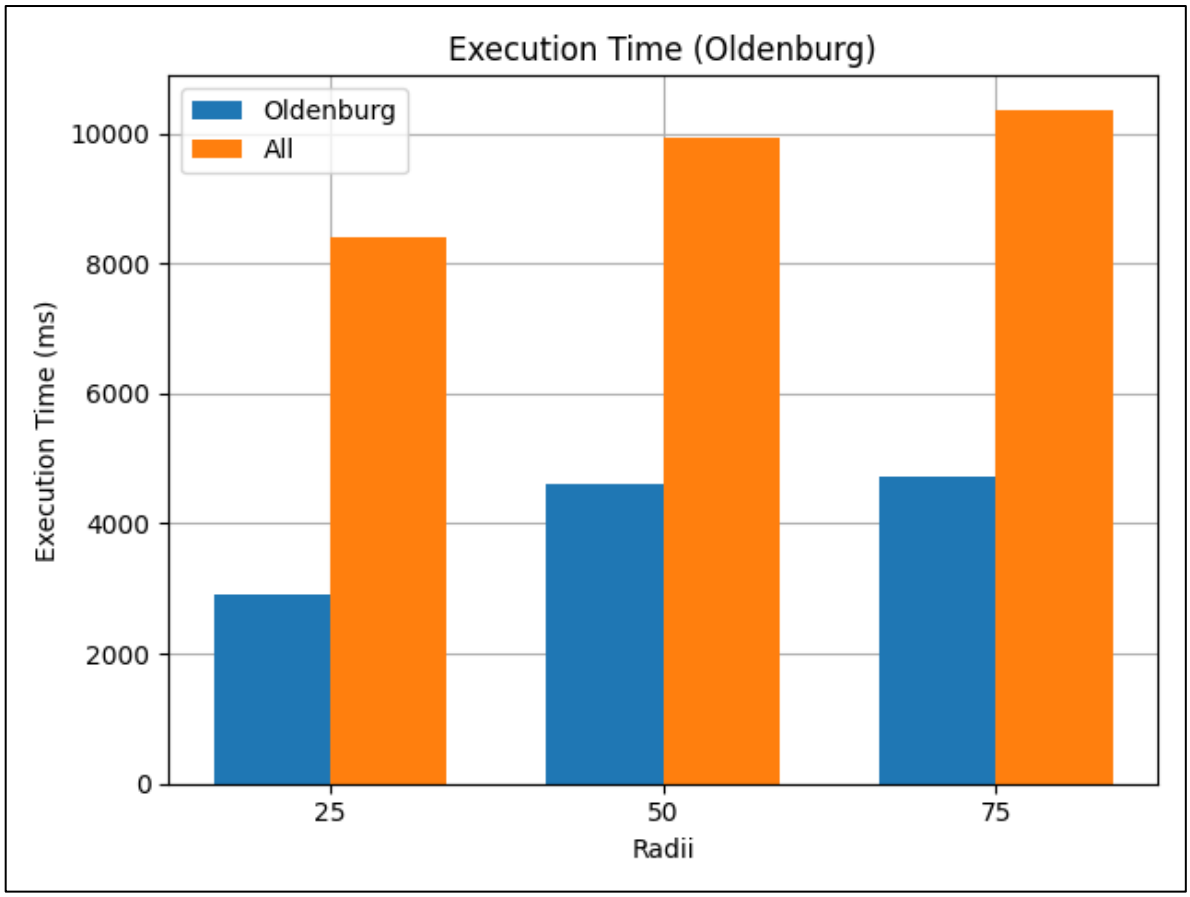


Σχήμα 6. 4 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία California και All

Χρόνος εκτέλεσης (ms)	Ακτίνα (km)	Αρχείο json
2912.86	25	Oldenburg
4614.48	50	Oldenburg
4715.75	75	Oldenburg
8403.94	25	All
9927.55	50	All
10368.07	75	All

Πίνακας 6. 5 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Oldenburg και All

Η αντίστοιχη γραφική παράσταση φαίνεται στο Σχήμα 6.5.

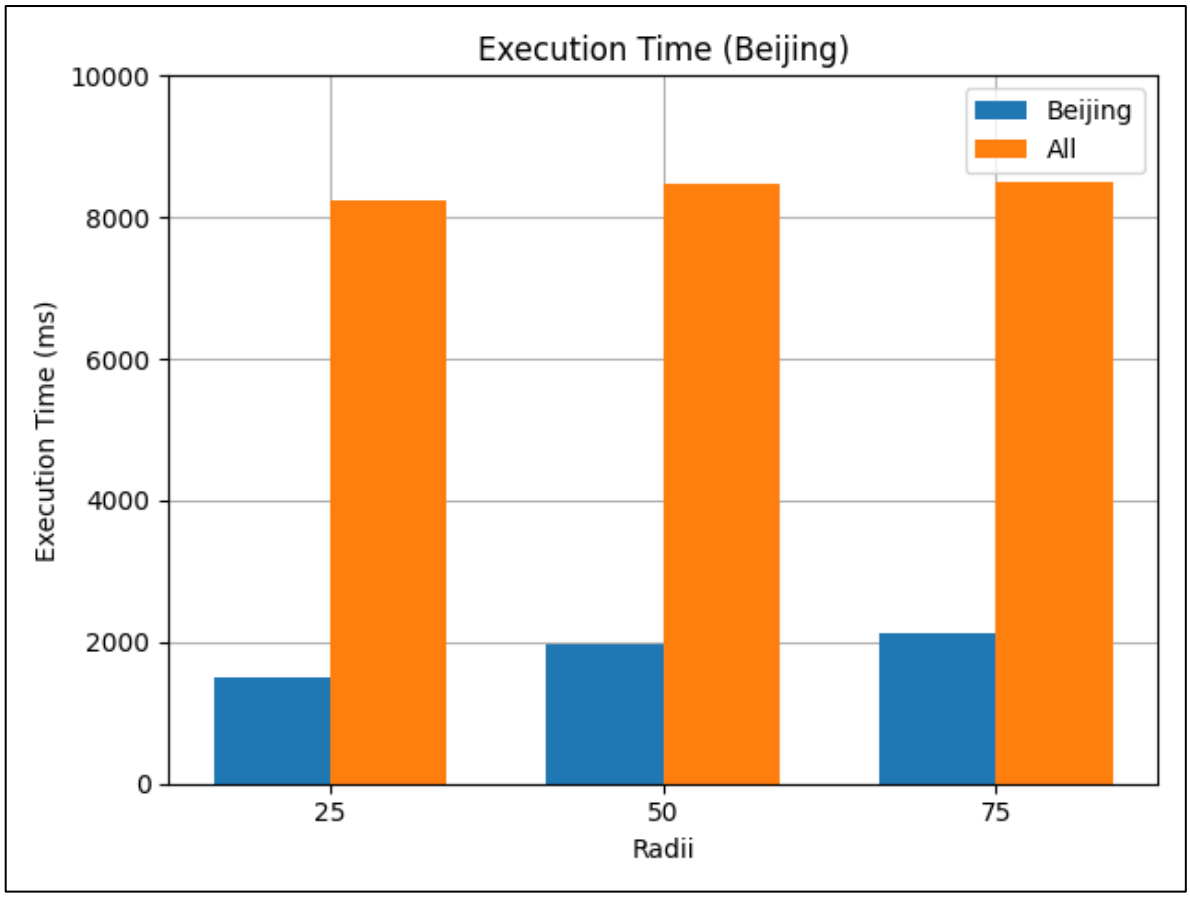


Σχήμα 6. 5 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Oldenburg και All

Χρόνος εκτέλεσης (ms)	Ακτίνα (km)	Αρχείο json
1512.82	25	Beijing
1964.22	50	Beijing
2125.34	75	Beijing
8228.64	25	All
8470.68	50	All
8502.22	75	All

Πίνακας 6. 6 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Beijing και All

Η αντίστοιχη γραφική παράσταση φαίνεται στο Σχήμα 6.6.



Σχήμα 6. 6 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για φόρτωση από αρχεία Beijing και All

Όπως παρατηρούμε από τα Σχήματα 6.3 – 6.6 ο χρόνος εκτέλεσης CPU του αλγόριθμου του EcoCharge για ακτίνες 25 km, 50 km και 75 km και για τις τέσσερις διαδρομές αυξάνεται κατά πολύ όταν οι φορτιστές φορτώνονται από το αρχείο All.json. Αυτό είναι απολύτως φυσιολογικό αφού το αρχείο αυτό περιέχει πληροφορίες για όλους τους σταθμούς φόρτισης που έχουμε στην διάθεσή μας (1601 σταθμοί), ενώ τα υπόλοιπα αρχεία περιέχουν πληροφορίες για πολύ λιγότερους σταθμούς.

6.5 Χρόνος εκτέλεσης CPU και βαθμός βιωσιμότητας για διαφορετικές αποστάσεις

Στο πείραμα αυτό υπολογίζουμε τον χρόνο εκτέλεσης CPU (βλ. Πίνακα 6.7) και τον βαθμό βιωσιμότητας (%) του αλγόριθμου του EcoCharge για Q (απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής) 5 km, 10 km και 15 km για τις τέσσερις διαδρομές. Ο βαθμός βιωσιμότητας είναι ουσιαστικά το κανονικοποιημένο άθροισμα των scores των προτεινόμενων σταθμών φόρτισης (βλ. Πίνακα 6.8). Το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης ήταν 100%. Η ακτίνα, μέσα στην οποία ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων είναι 50 km.

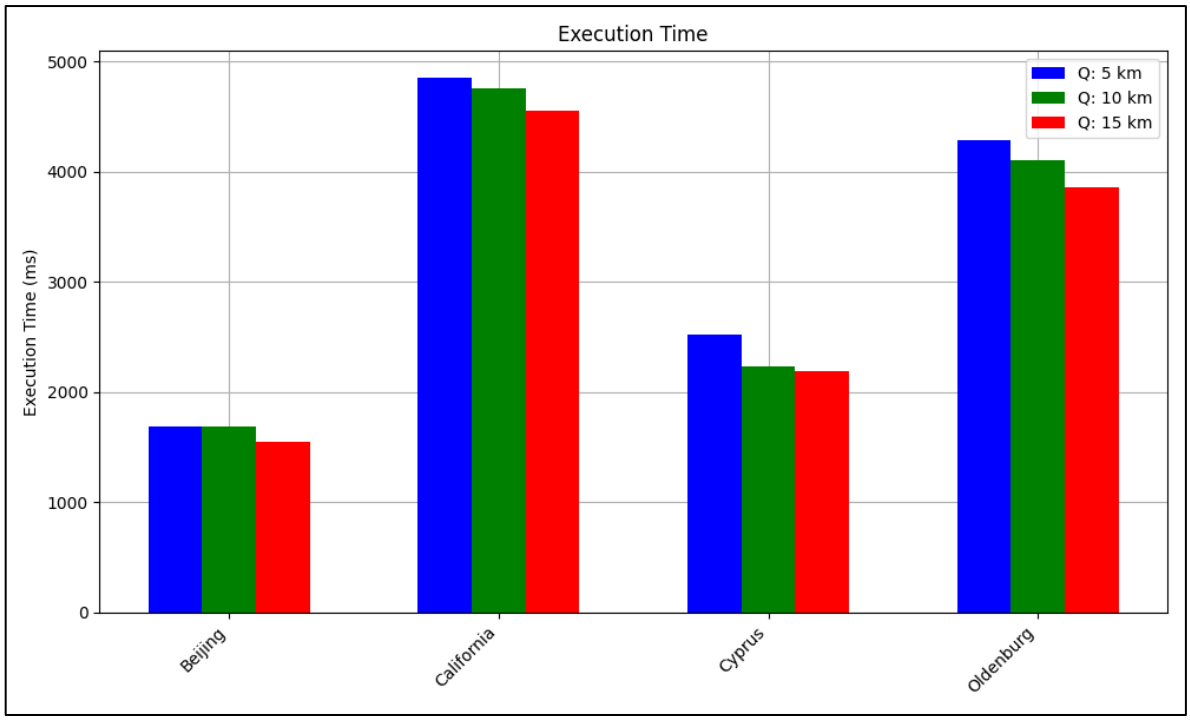
Χρόνος εκτέλεσης (ms)	Q (km)	Πόλη/χώρα
2525.66	5	Cyprus
2229.47	10	Cyprus
2195.53	15	Cyprus
4854.73	5	California
4754.55	10	California
4556.18	15	California
4285.34	5	Oldenburg
4101.9	10	Oldenburg
3856.62	15	Oldenburg
1683.31	5	Beijing
1688.5	10	Beijing
1548.06	15	Beijing

Πίνακας 6. 7 Χρόνος εκτέλεσης αλγόριθμου EcoCharge για διαφορετικά Q

Score	Q (km)	Πόλη/χώρα
17893.36	5	Cyprus
18014.11	10	Cyprus
18014.11	15	Cyprus
14689.16	5	California
14692.08	10	California
14780.89	15	California
18782.19	5	Oldenburg
18782.19	10	Oldenburg
18782.29	15	Oldenburg
10727.42	5	Beijing
10727.42	10	Beijing
10932.64	15	Beijing

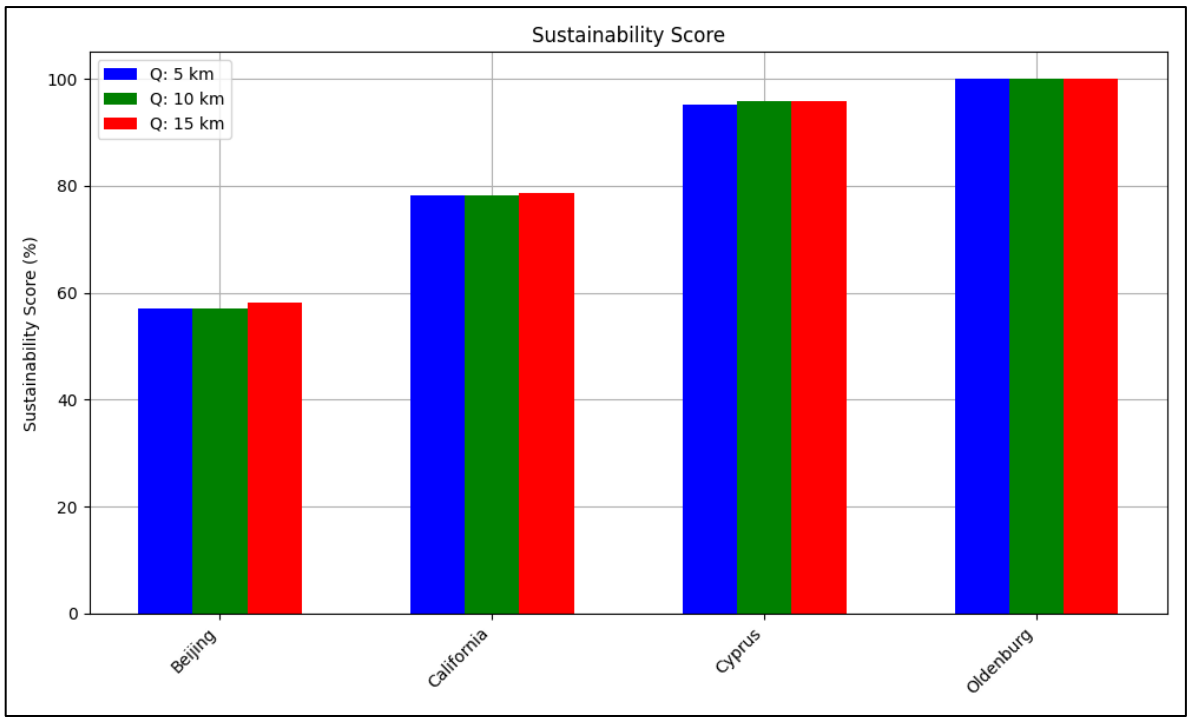
Πίνακας 6. 8 Scores αλγόριθμου EcoCharge για διαφορετικά Q

Η γραφική παράσταση του χρόνου εκτέλεσης του αλγόριθμου EcoCharge για Q 5 km, 10 km και 15 km για τις τέσσερις διαδρομές φαίνεται στο Σχήμα 6.7, ενώ στο Σχήμα 6.8 βλέπουμε την αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας (%) του αλγόριθμου.



Σχήμα 6. 7 Γραφική παράσταση χρόνου εκτέλεσης αλγόριθμου EcoCharge για διαφορετικά Q

Παρατηρώντας το Σχήμα 6.7 βλέπουμε ότι και στις τέσσερις διαδρομές (με εξαίρεση το Beijing Q=10 km) όταν αυξάνεται η απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής μειώνεται ο χρόνος εκτέλεσης. Αυτό είναι απολύτως λογικό αφού στον αλγόριθμο του EcoCharge όταν η απόσταση είναι μεγαλύτερη από το Q γίνονται ξανά οι υπολογισμοί για την εύρεση των κατάλληλων φορτιστών. Επίσης, οι μεγαλύτεροι χρόνοι εμφανίζονται στην California, αφού στην περιοχή αυτή υπάρχουν οι περισσότεροι φορτιστές που διαθέτουμε (907) και οι μικρότεροι χρόνοι στο Beijing, λόγω του ότι εκεί υπάρχουν οι λιγότεροι φορτιστές που διαθέτουμε (175).



Σχήμα 6. 8 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικά Q

Παρατηρώντας το Σχήμα 6.8 βλέπουμε ότι και στις τέσσερις διαδρομές όταν αυξάνεται η απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής αυξάνεται ελάχιστα ή παραμένει ίδιος ο βαθμός βιωσιμότητας. Αυτό είναι λογικό αφού η ακτίνα παραμένει ίδια. Επίσης, οι μεγαλύτεροι βαθμοί βιωσιμότητας εμφανίζονται στο Oldenburg και οι μικρότεροι στο Beijing, λόγω του ότι σε κάποια σημεία κατά μήκος της διαδρομής στο Beijing δεν υπήρχαν καθόλου φορτιστές ή υπήρχαν ένας ή δύο μέσα στην ακτίνα αναζήτησης.

6.6 Βαθμός βιωσιμότητας για διαφορετικά βάρη εκτιμώμενων στοιχείων

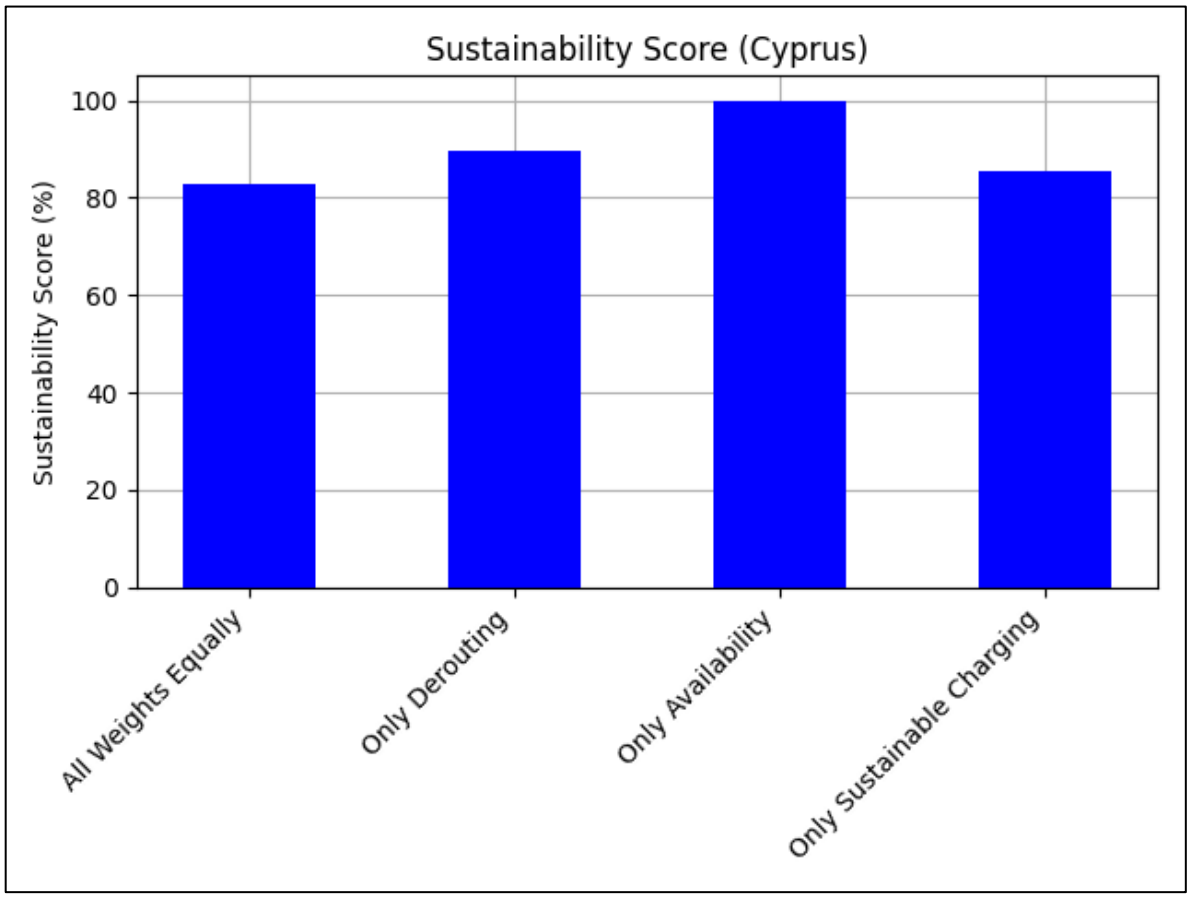
Στο πείραμα αυτό υπολογίζουμε τον βαθμό βιωσιμότητας (%) του αλγόριθμου του EcoCharge για διαφορετικά βάρη των εκτιμώμενων στοιχείων, για τις τέσσερις διαδρομές. Τα βάρη των εκτιμώμενων στοιχείων είναι τα ακόλουθα:

- All Weights Equally: Derouting 33%, Availability 33%, Sustainable Charging 33%
- Only Derouting: Derouting 99%
- Only Availability: Availability 99%
- Only Sustainable Charging: Sustainable Charging 99%

Ο βαθμός βιωσιμότητας είναι ουσιαστικά το κανονικοποιημένο άθροισμα των scores των προτεινόμενων σταθμών φόρτισης. Το Q (απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής) είναι 5 km. Η ακτίνα, μέσα στην οποία ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων είναι 50 km.
Στον Πίνακα 6.9 φαίνονται τα scores του αλγόριθμου του EcoCharge για τα διάφορα βάρη των εκτιμώμενων στοιχείων για την Κύπρο και στο Σχήμα 6.9 η αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας.

Score	Βάρος	Πόλη/χώρα
5904.81	All Weights Equally	Cyprus
6385.51	Only Derouting	Cyprus
7135.42	Only Availability	Cyprus
6097.67	Only Sustainable Charging	Cyprus

Πίνακας 6. 9 Scores αλγόριθμου EcoCharge για διαφορετικά βάρη για την Κύπρο

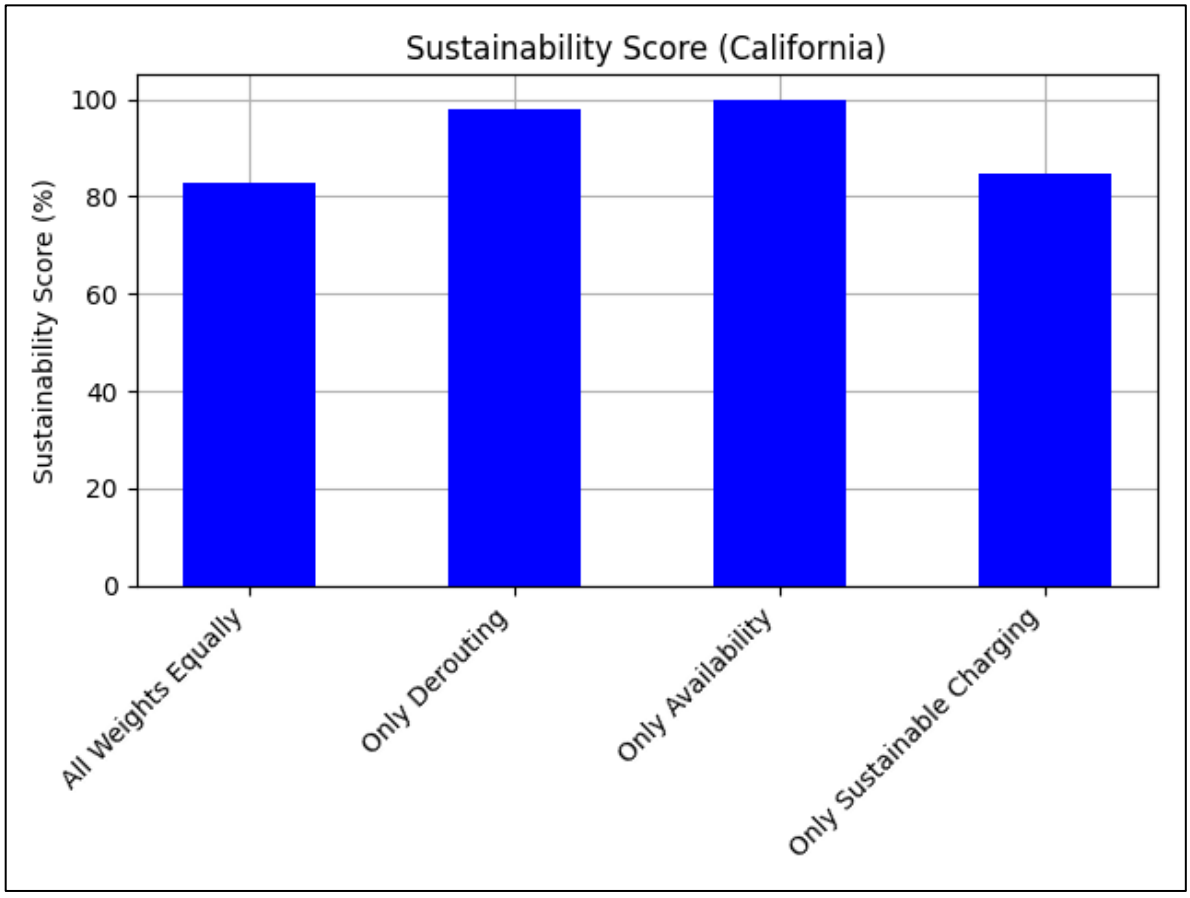


Σχήμα 6. 9 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικά βάρη για την Κύπρο

Στον Πίνακα 6.10 φαίνονται τα scores του αλγόριθμου του EcoCharge για τα διάφορα βάρη των εκτιμώμενων στοιχείων για την California και στο Σχήμα 6.10 η αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας.

Score	Βάρος	Πόλη/χώρα
4847.42	All Weights Equally	California
5730.83	Only Derouting	California
5844.96	Only Availability	California
4939.73	Only Sustainable Charging	California

Πίνακας 6. 10 Scores αλγόριθμου EcoCharge για διαφορετικά βάρη για την California

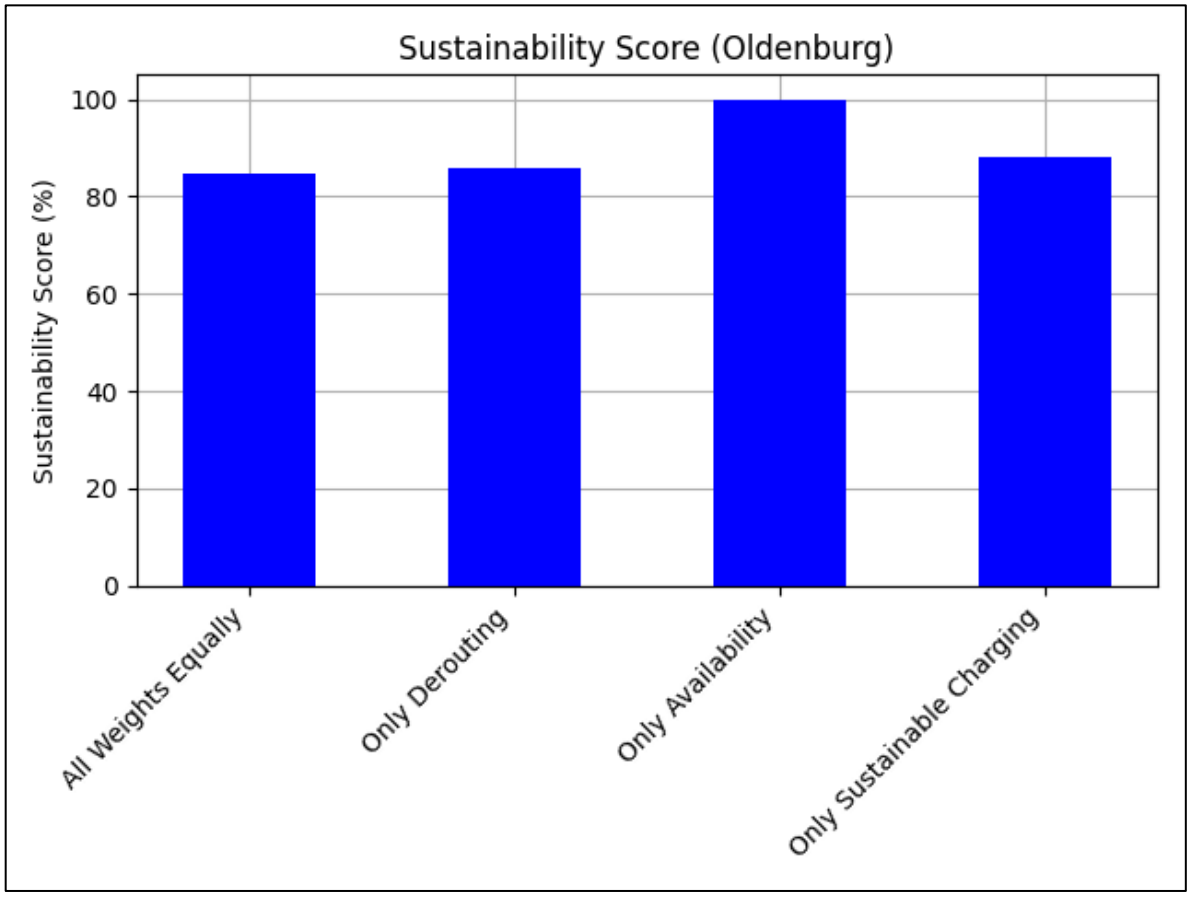


Σχήμα 6. 10 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικά βάρη για την California

Στον Πίνακα 6.11 φαίνονται τα scores του αλγόριθμου του EcoCharge για τα διάφορα βάρη των εκτιμώμενων στοιχείων για το Oldenburg και στο Σχήμα 6.11 η αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας.

Score	Βάρος	Πόλη/χώρα
6198.12	All Weights Equally	Oldenburg
6302.89	Only Derouting	Oldenburg
7328.47	Only Availability	Oldenburg
6443.68	Only Sustainable Charging	Oldenburg

Πίνακας 6. 11 Scores αλγόριθμου EcoCharge για διαφορετικά βάρη για το Oldenburg

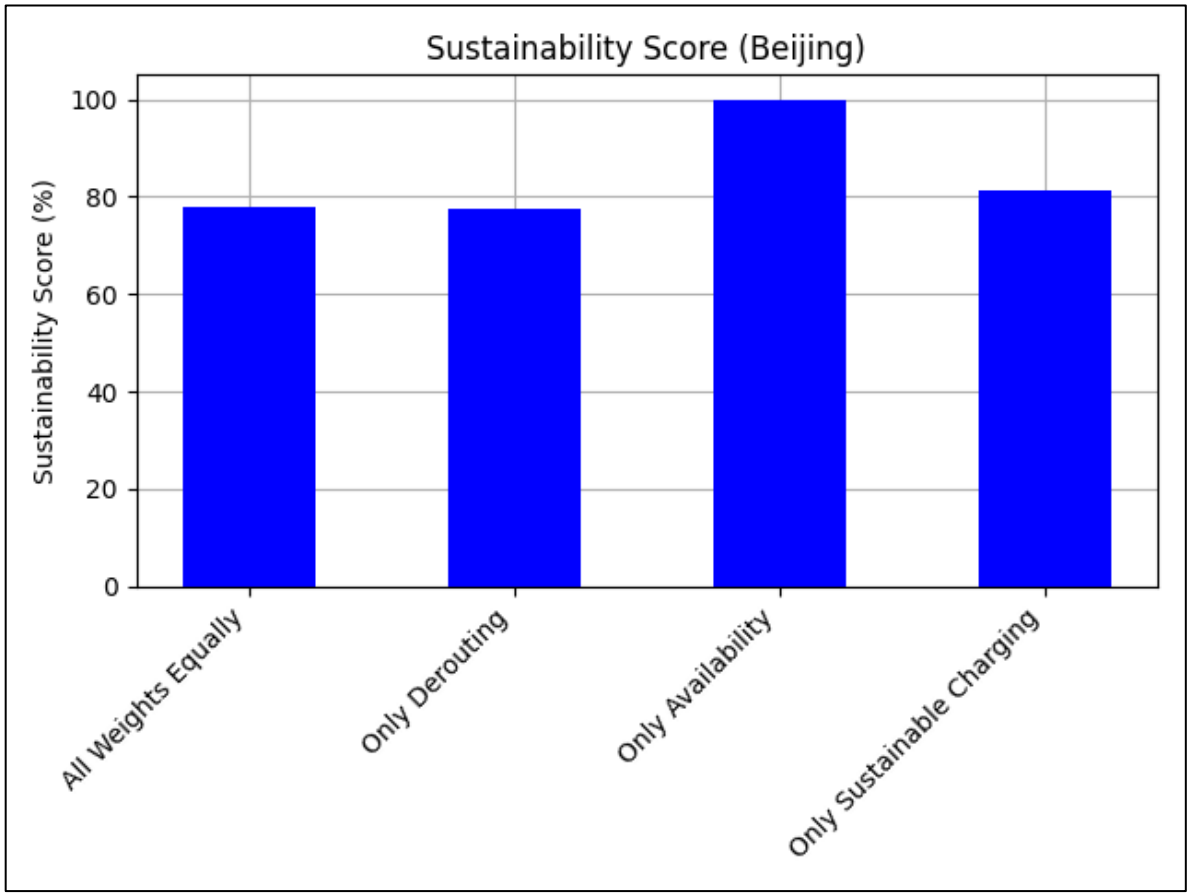


Σχήμα 6. 11 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικά βάρη για το Oldenburg

Στον Πίνακα 6.12 φαίνονται τα scores του αλγόριθμου του EcoCharge για τα διάφορα βάρη των εκτιμώμενων στοιχείων για το Beijing και στο Σχήμα 6.12 η αντίστοιχη γραφική παράσταση του βαθμού βιωσιμότητας.

Score	Βάρος	Πόλη/χώρα
3540.05	All Weights Equally	Beijing
3522.47	Only Derouting	Beijing
4549.55	Only Availability	Beijing
3699.00	Only Sustainable Charging	Beijing

Πίνακας 6. 12 Scores αλγόριθμου EcoCharge για διαφορετικά βάρη για το Beijing



Σχήμα 6. 12 Γραφική παράσταση βαθμού βιωσιμότητας αλγόριθμου EcoCharge για διαφορετικά βάρη για το Beijing

Όπως παρατηρούμε από τα Σχήματα 6.9 – 6.12 ο μεγαλύτερος βαθμός βιωσιμότητας και για τις τέσσερις διαδρομές εμφανίζεται στην περίπτωση που χρησιμοποιούμε μόνο τη διαθεσιμότητα του φορτιστή (Only Availability) και κανένα άλλο εκτιμώμενο στοιχείο. Αυτό όμως είναι σχετικό, αφού οι τιμές της διαθεσιμότητας καθώς και οι τιμές του βιώσιμου επιπέδου φόρτισης είναι στην περίπτωση μας τυχαίες.

6.7 Χρόνος εμφάνισης φορτιστών στον χάρτη

Στο πείραμα αυτό υπολογίζουμε τον χρόνο που χρειάζεται να περάσει για να εμφανιστούν οι φορτιστές στον χάρτη στη συγκεκριμένη πόλη/χώρα που επιλέγουμε. Ο web browser που χρησιμοποιήθηκε είναι ο Firefox. Το κόστος παρέκκλισης από την προγραμματισμένη διαδρομή, η διαθεσιμότητα του φορτιστή και το βιώσιμο επίπεδο φόρτισης είναι 100%. Η ακτίνα, μέσα στην οποία ο αλγόριθμος προτείνει σταθμούς φόρτισης των ηλεκτρικών οχημάτων είναι 50 km. Το Q (απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής) είναι 5 km.
Στον Πίνακα 6.13 φαίνεται ο χρόνος που χρειάζεται να περάσει για να εμφανιστούν οι φορτιστές στον χάρτη στη συγκεκριμένη πόλη/χώρα που επιλέγουμε.

Πόλη/χώρα	Χρόνος (ms)	Αριθμός φορτιστών
Beijing	31	175
Oldenburg	34	203
Cyprus	38	223
California	245	907
All	286	1601

Πίνακας 6. 13 Χρόνος εμφάνισης φορτιστών στον χάρτη

Όπως παρατηρούμε από τον Πίνακα 6.13 όταν αυξάνεται ο αριθμός των φορτιστών, αυξάνεται και ο χρόνος εμφάνισης των φορτιστών στο χάρτη. Αυτό είναι φυσιολογικό αφού όταν ο χρήστης τσεκάρει το checkbox για "Show All Chargers" για μια συγκεκριμένη πόλη/χώρα, τότε ανοίγει το αντίστοιχο αρχείο json που περιέχει πληροφορίες για τους

φορτιστές και φορτώνονται τα δεδομένα του. Ακολούθως, εμφανίζονται οι φορτιστές στο χάρτη. Επομένως, όσο μεγαλύτερο είναι το αρχείο, τόσο περισσότερος χρόνος χρειάζεται για να φορτωθεί.

Κεφάλαιο 7

Συμπεράσματα

Σε αυτό το κεφάλαιο παρουσιάζονται τα συμπεράσματα της διπλωματικής αυτής εργασίας και η μελλοντική εργασία που μπορεί να γίνει για την περαιτέρω ανάπτυξη του EcoCharge.

7.1 Συμπεράσματα

Η διπλωματική αυτή εργασία ασχολείται με την υλοποίηση και την αποτίμηση ενός γεωγραφικού συστήματος για την ηλιακή φόρτιση των ηλεκτρικών οχημάτων.

Τα τελευταία χρόνια η αγορά ηλεκτρικών οχημάτων έχει τεράστια αύξηση, ενώ υπάρχει και αυξανόμενο ενδιαφέρον για την ενσωμάτωση των ανανεώσιμων πηγών ενέργειας στις υποδομές φόρτισης των ηλεκτρικών οχημάτων (φωτοβολταϊκά πάνελ, ανεμογεννήτριες). Η μετάβαση στη χρήση ηλεκτρικών οχημάτων αποτελεί ένα σημαντικό βήμα προς την προστασία του περιβάλλοντος και τη μείωση των εκπομπών αερίων του θερμοκηπίου. Αν η ενέργεια όμως με την οποία φορτίζονται τα οχήματα αυτά προέρχεται από μη ανανεώσιμη ενέργεια, δηλαδή ενέργεια που προέρχεται από την καύση ορυκτών καυσίμων, όπως το πετρέλαιο και το μαζούτ, μπορεί να αναιρεθούν τα περιβαλλοντικά οφέλη της ηλεκτροκίνησης. Η αποτελεσματική χρήση ηλεκτρικών οχημάτων για την ελάχιστη επίδραση στο περιβάλλον απαιτεί την ανάπτυξη και την ευρεία χρήση ανανεώσιμων πηγών

ενέργειας, όπως η ηλιακή και η αιολική ενέργεια, για τη φόρτισή τους. Μόνο με αυτόν τον τρόπο μπορούμε να εξασφαλίσουμε τη μακροπρόθεσμη βιωσιμότητα της ηλεκτροκίνησης και τα οφέλη της για το περιβάλλον.

Το EcoCharge είναι ένα καινοτόμο σύστημα που υπολογίζει και παρουσιάζει σε ένα χάρτη τους κορυφαία καταταγμένους, με βάση τον αλγόριθμο του EcoCharge, σταθμούς φόρτισης ηλεκτρικών οχημάτων σε κάθε σημείο του χάρτη. Το πρόβλημα μοντελοποιήθηκε ως ένα συνεχές ερώτημα k πλησιέστερων γειτόνων που ανακτά τους k πλησιέστερους γείτονες κάθε σημείου λαμβάνοντας υπόψη τα εκτιμώμενα στοιχεία (κόστος παρέκκλισης από προγραμματισμένη διαδρομή, διαθεσιμότητα φορτιστή, βιώσιμο επίπεδο φόρτισης). Το EcoCharge μπορεί να χρησιμοποιηθεί με τρεις διαφορετικούς τρόπους: στο ενσωματωμένο λειτουργικό σύστημα ενός οχήματος, οι υπολογισμοί πραγματοποιούνται κεντρικά σε ένα server και η διαχείριση της λειτουργικότητας γίνεται από μια συσκευή άκρου (π.χ., έξυπνο τηλέφωνο). Σκοπός του EcoCharge είναι να συνεισφέρει σημαντικά στην αγορά της περιβαλλοντικά συνειδητής φόρτισης των ηλεκτρικών οχημάτων.

Ο EcoCharge client υποστηρίζεται από έναν κεντρικό server, ο οποίος αλληλοεπιδρά με εξωτερικά APIs για την ανάκτηση δεδομένων πρόβλεψης καιρού σε πραγματικό χρόνο, λεπτομερών πληροφοριών οδικού δικτύου και κυκλοφοριακής κίνησης, της διαθεσιμότητας των φορτιστών και μιας περιεκτικής λίστας με όλους τους διαθέσιμους φορτιστές ηλεκτρικών οχημάτων με βάση την τοποθεσία του χρήστη.

Υπάρχουν αρκετές υπηρεσίες που ασχολούνται με την φόρτιση και την διαδρομή των ηλεκτρικών οχημάτων, οι οποίες επικεντρώνονται στο να βοηθούν τους χρήστες να εντοπίζουν τα σημεία φόρτισης των οχημάτων τους, ενώ αρκετές χρησιμοποιούν ανανεώσιμες πηγές ενέργειας. Αξίζει να σημειωθεί ότι η IONITY χρησιμοποιεί 100% ανανεώσιμη ενέργεια. Η βελτιστοποίηση της φόρτισης των ηλεκτρικών οχημάτων χρησιμοποιώντας ανανεώσιμες πηγές ενέργειας, αποτελεί έναν σημαντικό τομέα έρευνας.

Για την υλοποίηση του συστήματός μας χρησιμοποιήθηκαν για το backend: Python 3, Flask, CORS, GeoPy, ενώ για το frontend: HTML, CSS, JavaScript, Leaflet, Leaflet Routing Machine, Leaflet.Locate, Leaflet.markercluster, Bootstrap, JQuery, Mapbox, χάρτες από OpenStreetMap, OpenTopoMap και ESRI.

Ο χρήστης επιλέγει την πόλη/χώρα, την αρχή και το τέλος της διαδρομής, την βαρύτητα των εκτιμώμενων στοιχείων, την ακτίνα και τον χάρτη και βλέπει την προτεινόμενη διαδρομή στο χάρτη. Ο χρήστης έχει ακόμη την δυνατότητα με right-click σε ένα σημείο στο χάρτη ή με την μετακίνηση ενός ταξί να βλέπει τους κορυφαία καταταγμένους σταθμούς φόρτισης ηλεκτρικών οχημάτων που βρίσκονται κοντά στο σημείο αυτό και πληροφορίες σχετικά με τους σταθμούς αυτούς. Επίσης, μπορεί να δει το demo με την πορεία ενός ταξί από την αρχή στο τέλος της διαδρομής, όπου εμφανίζονται σε κάθε σημείο κατά μήκος της διαδρομής οι κορυφαία καταταγμένοι σταθμοί φόρτισης.

Με βάση τα πειράματά μας συμπεραίνουμε ότι όταν αυξάνεται η ακτίνα, μέσα στην οποία ο αλγόριθμος του EcoCharge θα προτείνει σταθμούς φόρτισης, αυξάνεται και ο χρόνος εκτέλεσης του αλγόριθμου και ο βαθμός βιωσιμότητας. Επίσης, όταν αυξάνεται η απόσταση μεταξύ διαδοχικών σημείων κατά μήκος της διαδρομής μειώνεται ο χρόνος εκτέλεσης του αλγόριθμου και αυξάνεται ελάχιστα ή παραμένει ίδιος ο βαθμός βιωσιμότητας. Ακόμη, όταν οι φορτιστές φορτώνονται από το αρχείο που περιέχει όλους τους φορτιστές αυξάνεται κατά πολύ ο χρόνος εκτέλεσης του αλγόριθμου σε σχέση με το αρχείο που περιέχει τους φορτιστές κάθε πόλης/χώρας. Τέλος, όταν αυξάνεται ο αριθμός των διαθέσιμων φορτιστών, αυξάνεται και ο χρόνος εμφάνισής τους στο χάρτη.

7.2 Μελλοντική εργασία

Στο μέλλον, σχεδιάζεται η ενσωμάτωση του EcoCharge με τεχνολογίες έξυπνων δικτύων και η εκμετάλλευση των τιμών της ηλεκτρικής ενέργειας τις ώρες μη υψηλής ζήτησης και των υπηρεσιών σταθεροποίησης του δικτύου. Επιπλέον, σχεδιάζεται η παρακολούθηση της κυκλοφοριακής συμφόρησης για την ανακατεύθυνση των οδηγών προς εναλλακτικούς σταθμούς φόρτισης ηλεκτρικών οχημάτων.

Βιβλιογραφία

[1] European Commission. EU Mission: Climate-Neutral and Smart Cities. Retrieved from https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe/eu-missions-horizon-europe/climate-neutral-and-smart-cities_en

[2] 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

[3] 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.

[4] Statista Research Department. (2022, July 14). U.S. electric vehicle energy charging demand between 2020 and 2035. Statista. https://www.statista.com/statistics/952069/electric-vehicle-charging-energy-demand/

[5] 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

[6] 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

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

[8] 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.

[9] 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

[10] "A Framework for Continuous kNN Ranking of EV Chargers with Estimated Components", Soteris Constantinou, Constantinos Costa, Andreas Konstantinidis, Mohamed F. Mokbel, Demetrios Zeinalipour-Yazti, "The 40th IEEE International Conference on Data Engineering" (ICDE '24), IEEE Computer Society, ISBN:, Pages: 13 pages, May 13 - May 17, 2024, Utrecht, Netherlands, 2024.

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

[12] A Better Routeplanner. Retrieved from https://abetterrouteplanner.com/

[13] PlugShare. Retrieved from https://www.plugshare.com/

[14] IONITY. Retrieved from https://ionity.eu/

[15] Tesla Supercharger. Retrieved from https://www.tesla.com/supercharger

[16] Shell Recharge. Retrieved from https://shellrecharge.com/

[17] EnBW mobility+. Retrieved from https://www.enbw.com/company/topics/e-mobility/

[18] 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

[19] 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

[20] 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

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

[22] 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.

[23] 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.

[24] K. Raptopoulou, A. Papadopoulos, and Y. Manolopoulos, “Fast nearestneighbor query processing in moving-object databases.” Geoinformatica 7, 2003, p. 113–137.

[25] 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. Ludaescher and N. Mamoulis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 549–557.

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

[27] 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.

[28] 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.

[29] 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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

[35] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[36] E. W. Dijkstra, “A note on two problems in connection with graphs,” Numer. Math., vol. 1, no. 1, p. 269–271, dec 1959. [Online]. Available: https://doi.org/10.1007/BF01386390

[37] B. Shen, Y. Zhao, G. Li, W. Zheng, Y. Qin, B. Yuan, and Y. Rao, “V-tree: Efficient knn search on moving objects with road-network constraints,” in 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 609–620.

[38] L. Li, M. Zhang, W. Hua, and X. Zhou, “Fast query decomposition for batch shortest path processing in road networks,” in 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 1189–1200.

[39] B. Yao, Z. Chen, X. Gao, S. Shang, S. Ma, and M. Guo, “Flexible aggregate nearest neighbor queries in road networks,” in 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018, pp. 761–772.

[40] Z. Li, L. Chen, and Y. Wang, “G*-tree: An efficient spatial index on road networks,” in 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 268–279.

[41] Z. Liu, Z. Gong, J. Li, and K. Wu, “Mobility-aware dynamic taxi ridesharing,” in 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 961–972.

[42] T. T. Portela, V. Bogorny, A. Bernasconi, and C. Renso, “Automatise: Multiple aspect trajectory data mining tool library,” in 2022 23rd IEEE International Conference on Mobile Data Management (MDM), 2022, pp. 282–285.

[43] Y. Sun, N. Nielsen, X. Xie, T. B. Pedersen, U. Simonsen, H. Lu, and M. Ainciburu, “Urbangen: Generating combined in- and outdoor trajectories,” in 2022 23rd IEEE International Conference on Mobile Data Management (MDM), 2022, pp. 270–273.

[44] A. Sabour, J. Yao, A. Salama, C. Hampshire, E. Al-Masri, M. Ali, H. Govind, M. Tan, V. Agrawal, E. Maresov, and R. Prakash, “Maps vision: A computer vision-based system for detecting discrepancies in map textual labels,” in 2022 23rd IEEE International Conference on Mobile Data Management (MDM), 2022, pp. 298–301.

[45] Qingfeng Zhu, Sai Ji, and Qi Liu. 2021. Privacy-Preserving Smart Road Pricing System in Smart Cities. In 2021 IEEE Conference on Dependable and Secure Computing (DSC). 1–2. https://doi.org/10.1109/DSC49826.2021.9346274

[46] Yingying Yao, Xiaolin Chang, Lin Li, Jiqiang Liu, and Hong Wang. 2022. Dual Privacy-Preserving Lightweight Navigation System for Vehicular Networks. IEEE Access 10 (2022), 121120–121132. https://doi.org/10.1109/ACCESS.2022.3222302

[47] Dimitrios Tomaras, Michail Tsenos, and Vana Kalogeraki. 2022. A Framework for Supporting Privacy Preservation Functions in a Mobile Cloud Environment. In 2022 23rd IEEE International Conference on Mobile Data Management (MDM). 286–289. https://doi.org/10.1109/MDM55031.2022.00061

[48] “National centers for environmental information - global forecast system (gfs),” https://www.ncei.noaa.gov/products/weather-climatemodels/global-forecast, 2023.

[49] “European Centre for medium-range weather forecasts - advancing global NWP through international collaboration,” https://www.ecmwf.int/, 2023.

[50] OpenWeatherMap. Retrieved from https://openweathermap.org/

[51] "EcoCharge: A Framework for Sustainable Electric Vehicles Charging", Soteris Constantinou, Dimitris Papazachariou, Constantinos Costa, Andreas Konstantinidis, Mohamed F. Mokbel, Demetrios Zeinalipour-Yazti, "The 25th IEEE International Conference on Mobile Data Management" (MDM '24), IEEE Computer Society, ISBN:, Pages: 4 pages, June 24 - June 27, 2024, Brussels, Belgium, 2024.

[52] Python 3. Retrieved from https://www.python.org/

[53] Flask. Retrieved from https://flask.palletsprojects.com/en/3.0.x/

[54] GeoPy. Retrieved from https://geopy.readthedocs.io/en/stable/

[55] Leaflet. Retrieved from https://leafletjs.com/

[56] Leaflet Routing Machine. Retrieved from https://www.liedman.net/leaflet-routing-machine/

[57] Leaflet.Locate. Retrieved from https://github.com/domoritz/leaflet-locatecontrol

[58] Leaflet.markercluster. Retrieved from https://github.com/Leaflet/Leaflet.markercluster

[59] Bootstrap. Retrieved from https://getbootstrap.com/

[60] jQuery. Retrieved from https://jquery.com/

[61] Mapbox. Retrieved from https://www.mapbox.com/

[62] OpenStreetMap. Retrieved from https://www.openstreetmap.org/

[63] OpenTopoMap. Retrieved from https://opentopomap.org/

[64] Environmental Systems Research Institute. Retrieved from https://www.esri.com/

Παράρτημα Α

ecocharge_demo.py

```
from flask import Flask, jsonify, request
from flask_cors import CORS
from geopy.distance import geodesic
import json

# Create a Flask application instance named 'app'
app = Flask(__name__)

# Enable Cross-Origin Resource Sharing (CORS) for specific routes
CORS(app, resources={r"/chargers": {"origins": "*"},
                     r"/nearby_chargers": {"origins": "*"},
                     r"/nodes": {"origins": "*"}})

# Open the file "solarDataALL.txt" in read mode
fSolarData = open(r'solarDataALL.txt', "r")
# Read the contents of the "solarDataALL.txt" file and store them in
fSolarData
fSolarData = fSolarData.readlines()

# Open the file "nodes.txt" in read mode
fNodes = open(r'nodes.txt', "r")
# Read the contents of the "nodes.txt" file and store them in fNodes
fNodes = fNodes.readlines()



# Function to get the ranking chargers
def get_ranking_chargers(lat, lng, derouting_cost, charger_availability,
                         sustainable_charging_level, coordinatesEV,
radius):
    previous_center = [0, 0]  # Starting point
    all_ranking_chargers = []
```

```
    for node in fNodes:
        ranking_chargers = []
        dataOfNode = node.split(' ')
        center = [dataOfNode[2], dataOfNode[1]]

        if geodesic(previous_center, (lat, lng)).km > 5:
            filtered_coordinatesEV = []
            farthest_point = 0
            greenestCharger = 0
            for coord in coordinatesEV:
                charger_distance = geodesic((lat, lng), [coord[0],
coord[1]]).km
                if charger_distance <= radius:
                    filtered_coordinatesEV.append(coord)
                    if charger_distance > farthest_point:
                        farthest_point = charger_distance
                    if coord[4] > greenestCharger:
                        greenestCharger = coord[4]
            farthest_point = farthest_point * 2  # Going and returning
back
            previous_center = center
        else:
            print('skip')

        for coord in filtered_coordinatesEV:

            distance = geodesic((lat, lng), [coord[0], coord[1]]).km

            if geodesic((lat, lng), [coord[0], coord[1]]).km <= radius:
                travelDistanceCO2Cost_togo = geodesic((lat, lng),
[coord[0], coord[1]]).km
                minsToArrive = travelDistanceCO2Cost_togo
                travelDistanceCO2Cost_backSame = geodesic([coord[0],
coord[1]], (lat, lng)).km

                try:
```

```
                    travelDistanceCO2Cost_backNext = (
                        geodesic([coord[0], coord[1]],
[fNodes[int(dataOfNode[0]) + 1].split(' ')[2],

fNodes[int(dataOfNode[0]) + 1].split(' ')[1]]).km)
                except:
                    travelDistanceCO2Cost_backNext = farthest_point / 2
                    # The try catch is only to trigger the very last node
element

                try:
                    travelDistanceCO2Cost_backPrevious = (
                        geodesic([coord[0], coord[1]],
[fNodes[int(dataOfNode[0]) - 1].split(' ')[2],

fNodes[int(dataOfNode[0]) - 1].split(' ')[1]]).km)
                except:
                    travelDistanceCO2Cost_backPrevious = farthest_point /
2
                    # The try catch is only to trigger the previous node
element if not exist

                travelDistanceCO2Cost = (travelDistanceCO2Cost_togo +

min(travelDistanceCO2Cost_backSame, travelDistanceCO2Cost_backNext,

travelDistanceCO2Cost_backPrevious))
                travelDistanceCO2Cost = 100 - ((travelDistanceCO2Cost *
100) / farthest_point)

                chargingIntersection = (coord[3] + coord[4]) / 2
                chargingCO2Cost = (chargingIntersection * 100) /
greenestCharger

                # Consider Estimated Time of Arrival (ETA)
                if minsToArrive > 45:
                    chargingCO2Cost = chargingCO2Cost / 4
```

```
                elif minsToArrive > 30:
                    chargingCO2Cost = chargingCO2Cost / 3
                elif minsToArrive > 15:
                    chargingCO2Cost = chargingCO2Cost / 2

                calculatedMetric = coord[:]  # Clone the coord list

                availability = (coord[5] + coord[6]) / 2

                ecocharge_score = (((travelDistanceCO2Cost - 2) *
derouting_cost)
                                   + (availability *
charger_availability)
                                   + (chargingCO2Cost *
sustainable_charging_level))

                calculatedMetric.append(ecocharge_score)

                calculatedMetric.append(minsToArrive)
                calculatedMetric.append(availability)
                calculatedMetric.append(chargingCO2Cost)
                calculatedMetric.append((travelDistanceCO2Cost - 2))
                calculatedMetric.append(distance)

                ranking_chargers.append(calculatedMetric)

        ranking_chargers.sort(key=lambda x: x[8], reverse=True)  # Sort
by ecocharge score in descending order
        all_ranking_chargers.append(ranking_chargers[0:5])  # 5 nearby
chargers

        return all_ranking_chargers


# Define a route for handling POST requests to '/nodes'
@app.route('/nodes', methods=['POST'])
def write_nodes():
```

```
    # Extract JSON data from the request
    data = request.get_json()

    if 'nodesList' in data:
        # Extract nodes list from the JSON data
        nodes_list = data['nodesList']

        # Write nodes to a text file
        write_to_file(nodes_list)

        return jsonify({'message': 'Data received and saved to file
successfully'})

    return jsonify({'error': 'Invalid request'})


def write_to_file(data):
    # Open the file in write mode
    with open('nodes.txt', 'w') as file:
        # Write each item in the data list to the file
        for item in data:
            file.write(f"{item}\n")


# Define a route for handling POST requests to '/chargers'
@app.route('/chargers', methods=['POST'])
def get_chargers():
    # Retrieve JSON data from the request
    data = request.get_json()

    # Extract the 'selected_value' from the JSON data
    selected_value = data.get('selected_value')

    # Open the JSON file corresponding to the selected value
    f = open(r'' + selected_value + '.json')
    # Load the JSON data from the file
    chargers = json.load(f)
```

```
    # Close the file after loading the data
    f.close()

    return jsonify(chargers)


# Define a route for handling POST requests to '/nearby_chargers'
@app.route('/nearby_chargers', methods=['POST'])
def get_nearby_chargers():
    # Retrieve JSON data from the request
    data = request.get_json()

    lat = data['lat']
    lng = data['lng']
    derouting_cost = data['derouting_cost']
    charger_availability = data['charger_availability']
    sustainable_charging_level = data['sustainable_charging_level']
    selected_value = data['selected_value']
    radius = data['radius']

    # Initialize empty list to store coordinates of EV chargers
    coordinatesEV = []
    # Initialize count variable
    count = 0

    # Open the JSON file corresponding to the selected value
    f = open(r'' + selected_value + '.json')
    # Load the JSON data from the file
    chargers = json.load(f)

    for i in chargers:
        chargerKW = i['stations'][0]['outlets'][0]['kilowatts']
        if chargerKW is None:  # Pythonic way of checking for a null
value
            chargerKW = 7.2

        energyKWHmin = float(fSolarData[count].split(',')[2])
```

```
        energyKWHmax = float(fSolarData[count].split(',')[3])

        coordinatesEV.append([i['latitude'],
                              i['longitude'],
                              chargerKW,
                              energyKWHmin,
                              energyKWHmax,
                              i['availabilityMIN'],
                              i['availabilityMAX'],
                              i['name']])

        count += 1

    # Close the file
    f.close()

    nearby_chargers = (
        get_ranking_chargers(
            lat, lng, derouting_cost, charger_availability,
sustainable_charging_level, coordinatesEV, radius
        )
    )

    return jsonify({
        "latitude": lat,
        "longitude": lng,
        "derouting_cost": derouting_cost,
        "charger_availability": charger_availability,
        "sustainable_charging_level": sustainable_charging_level,
        "selected_value": selected_value,
        "radius": radius,
        "nearby_chargers": nearby_chargers
    })


# Check if this script is being executed as the main program
if __name__ == '__main__':
```

```
# Run the Flask application in debug mode
app.run(debug=True)
```

ecocharge_demo.html

```
<!DOCTYPE html>
<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">

<!-- Google signin client id -->
<meta name="google-signin-client_id"
content="314654602450-
f3dlg26nrl18j2t0g7s6r9as2399kqgk.apps.googleusercontent.com">

<title>EcoCharge Demo</title>

<!-- Leaflet CSS -->
<link rel="stylesheet"
href="https://unpkg.com/leaflet/dist/leaflet.css"/>

<!-- Leaflet Routing Machine CSS -->
<link rel="stylesheet" href="https://unpkg.com/leaflet-routing-
machine/dist/leaflet-routing-machine.css"/>

<!-- Leaflet Locate CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol/dist/L.Control.L
ocate.min.css"/>

<!-- Leaflet Marker Clustering CSS -->
<link rel="stylesheet"
href="https://unpkg.com/leaflet.markercluster/dist/MarkerCluster.css"/>
```

```
<link rel="stylesheet"
href="https://unpkg.com/leaflet.markercluster/dist/MarkerCluster.Default.
css"/>

<!-- Bootstrap 5.3.3 CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min
.css" rel="stylesheet"
integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

<style>
body {
padding: 0;
margin: 0;
height: 100%;
width: 100vw;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

html {
height: 100%;
width: 100vw;
}

#map {
position: absolute;
height: 100%;
width: 100vw;
top: 0;
bottom: 0;
right: 0;
left: 0;
}

/* Remove the flag in Leaflet control attribution */
```

```
.leaflet-control-attribution svg {
width: 0
}

/* Nearby chargers */
.custom-marker {
background-color: #008000;
color: #fff;
border-radius: 50%;
width: 30px;
height: 30px;
line-height: 30px;
text-align: center;
font-weight: bold;
}

/* Google Sign-In button */
.g-signin2 {
z-index: 9999;
}
</style>

</head>

<body>

<!-- Leaflet Javascript -->
<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

<!-- Leaflet Routing Machine Javascript -->
<script src="https://unpkg.com/leaflet-routing-machine/dist/leaflet-
routing-machine.js"></script>

<!-- Leaflet Locate Javascript -->
<script
src="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol/dist/L.Control.Lo
cate.min.js" charset="utf-8"></script>
```

```
<!-- Leaflet Marker Clustering Javascript -->
<script
src="https://unpkg.com/leaflet.markercluster/dist/leaflet.markercluster.j
s"></script>

<!-- Bootstrap 5.3.3 Javascript -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundl
e.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>

<!-- JQuery 3.7.1 Javascript -->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>

<!-- Google Platform Library -->
<script src="https://apis.google.com/js/platform.js" async
defer></script>

<!-- Map -->
<div id="map"></div>

<!--Left panel -->
<div class="row" id="panel" style="position: absolute; top: 5px; z-index:
9999; width: 20%; left: 5px;
background-color:#F1F1F1; border-radius: 8px;">
<button type="button" class="btn btn-primary" data-bs-toggle="collapse"
data-bs-target="#demo">Route Configurations</button>

<div class="col-1"></div>

<div id="demo" class="col-7 collapse">
<br>

<span>Select City/Country:</span>
```

```
<br>
<select id="chargersLocation" class="form-select" aria-label="Default
select example">
<option selected value="California">California</option>
<option value="Oldenburg">Oldenburg</option>
<option value="Cyprus">Cyprus</option>
<option value="Beijing">Beijing</option>
<option value="All">All</option>
</select>
<hr>

<label for="origin">Select Origin:</label><br>
<input type="text" class="form-control" id="origin" placeholder="Enter
origin address"
list="origin-suggestions">
<br>
<label for="destination">Select Destination:</label><br>
<input type="text" class="form-control" id="destination"
placeholder="Enter destination address"
list="destination-suggestions">
<hr>

<!-- Datalists for autocomplete suggestions -->
<datalist id="origin-suggestions"></datalist>
<datalist id="destination-suggestions"></datalist>

<span style="">Show All Chargers</span>
<label for="checkbox"><input type="checkbox" id="checkbox"
unchecked></label>
<hr>

<button type="button" class="btn btn-primary" onclick="showRoute()">Show
Route</button>
<br><br>
<button type="button" class="btn btn-warning"
onclick="resetMapView()">Reset View</button>
<br><br>
```

```
<button type="button" class="btn btn-success" onclick="startDemo()">Start
Demo</button>
<br><br>
<button type="button" class="btn btn-success"
onclick="togglePauseResume()">Pause/Resume Demo</button>
<br><br>
</div>

<div class="col-1"></div>

</div>

<!--Right panel -->
<div class="row" id="panel2" style="position: absolute; top: 5px; z-
index: 9999; width: 20%; right: 5px;
background-color:#F1F1F1; border-radius: 8px;">
<button type="button" class="btn btn-primary" data-bs-toggle="collapse"
data-bs-target="#demo2">User Login/Preferences</button>

<div class="col-1"></div>

<div id="demo2" class="col-7 collapse">
<br>

<!-- Google Sign-In button -->
<div class="g-signin2" data-onsuccess="onSignIn"></div>
<hr>

<span>Estimated Components:</span>
<br>
<label for="ec1">Derouting: </label>
<span id="range-value1"></span>
<span> %</span>
<input type="range" id="ec1" name="ec1" min="0" max="100" value="100"
class="range-slider"
onchange="updateSlider(this.value, 'ec1')"><br>
```

```
<label for="ec2">Availability: </label>
<span id="range-value2"></span>
<span> %</span>
<input type="range" id="ec2" name="ec2" min="0" max="100" value="100"
class="range-slider"
onchange="updateSlider(this.value, 'ec2')"><br>

<label for="ec3">Sustainable Charging: </label>
<span id="range-value3"></span>
<span> %</span>
<input type="range" id="ec3" name="ec3" min="0" max="100" value="100"
class="range-slider"
onchange="updateSlider(this.value, 'ec3')"><br>
<hr>

<label for="radius">Radius: </label>
<span id="range-value4"></span>
<span> Km</span>
<input type="range" id="radius" name="radius" min="0" max="100"
value="50" class="range-slider"
onchange="updateSlider(this.value, 'radius')"><br>
<br>

</div>

<div class="col-1"></div>

</div>

<script>
// Map initialization and configuration
// Disable zoomControl (topleft by default)
var map = L.map('map', {
zoomControl: false
}).setView([40, -105], 5);

// Add map scale
```

```
L.control.scale({
position: "bottomright",
maxWidth: 100,
metric: true,
imperial: true
}).addTo(map);

// Tile layers
// https://leaflet-extras.github.io/leaflet-providers/preview/
var OpenStreetMap =
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
{
attribution: '&copy; ' +
'<a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);

var OpenTopoMap =
L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png',
{
maxZoom: 17,
attribution: 'Map data: &copy; ' +
'<a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors, ' +
'<a href="http://viewfinderpanoramas.org">SRTM</a> | Map style: &copy; '
+
'<a href="https://opentopomap.org">OpenTopoMap</a> ' +
'(<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-
SA</a>)'
});

var Esri_WorldStreetMap =
L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_S
treet_Map/MapServer/tile/{z}/{y}/{x}',
{
attribution: 'Tiles &copy; Esri &mdash; Source: Esri, DeLorme, NAVTEQ,
USGS, Intermap, iPC, NRCAN, ' +
```

```
'Esri Japan, METI, Esri China (Hong Kong), Esri (Thailand), TomTom, 2012'
});

var Esri_WorldTopoMap =
L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_T
opo_Map/MapServer/tile/{z}/{y}/{x}',
{
attribution: 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom,
Intermap, iPC, ' +
'USGS, FAO, NPS, NRCAN, GeoBase, Kadaster NL, Ordnance Survey, Esri
Japan, METI, ' +
'Esri China (Hong Kong), and the GIS User Community'
});

var baseMaps = {
"OpenStreetMap": OpenStreetMap,
"OpenTopoMap": OpenTopoMap,
"Esri_WorldStreetMap": Esri_WorldStreetMap,
"Esri_WorldTopoMap": Esri_WorldTopoMap
};

var layersControlOptions = {
collapsed: true,
position: 'bottomright'
};

var layersControl = L.control.layers(baseMaps, null,
layersControlOptions);
layersControl.addTo(map);

// Add zoom control
L.control.zoom({
position: 'bottomright'
}).addTo(map);

// Show my location
L.control
```

```
.locate({
position: "bottomright",
showCompass: "true",
strings: {
title: "Show my location"
}
})
.addTo(map);

// Mapbox access token
const mapboxAccessToken =
'pk.eyJ1IjoiZGltaXRyaXMxOTc3IiwiYSI6ImNscmNqYW1tMjB2bTgyam13YmxqeXZjMjYif
Q.uy-DqB8KY3gD2zZxaG29Ew';



// Origin
var originInput = document.getElementById('origin');

// Destination
var destinationInput = document.getElementById('destination');

// Add an event listener to the 'input' event of the origin element
originInput.addEventListener('input', function () {
updateAutocomplete(originInput.value, 'origin');
});

// Add an event listener to the 'input' event of the destination element
destinationInput.addEventListener('input', function () {
updateAutocomplete(destinationInput.value, 'destination');
});

// Selected value = City/Country
var selected_value = document.getElementById("chargersLocation").value;
console.log(selected_value);

// Radius
var radius = parseInt(document.getElementById("radius").value);
```

```
console.log(radius);

// Estimated Component: Derouting
var slider1 = document.getElementById("ec1");
var output1 = document.getElementById("range-value1");
output1.innerHTML = slider1.value;

slider1.oninput = function () {
output1.innerHTML = this.value;
}

// Estimated Component: Availability
var slider2 = document.getElementById("ec2");
var output2 = document.getElementById("range-value2");
output2.innerHTML = slider2.value;

slider2.oninput = function () {
output2.innerHTML = this.value;
}

// Estimated Component: Sustainable Charging
var slider3 = document.getElementById("ec3");
var output3 = document.getElementById("range-value3");
output3.innerHTML = slider3.value;

slider3.oninput = function () {
output3.innerHTML = this.value;
}

// Radius
var slider4 = document.getElementById("radius");
var output4 = document.getElementById("range-value4");
output4.innerHTML = slider4.value;

slider4.oninput = function () {
output4.innerHTML = this.value;
}
```

```
// Add an event listener to the 'change' event of the chargersLocation
element
document.getElementById("chargersLocation").addEventListener('change',
function () {
var selectedValue = this.value;
if (selectedValue === "All") {
// Set default view when selected value is "All"
map.setView([40, -105], 5);
} else {
// Call the geocode function with the selected value
geocode(selectedValue, function (coordinates) {
if (coordinates) {
// Set the view of the map to the coordinates obtained from geocoding
map.setView([coordinates[1], coordinates[0]], 8);
} else {
console.error('Unable to geocode the selected name.');
}
});
}
});

// Add an event listener to the 'change' event of the checkbox element
document.getElementById('checkbox').addEventListener('change', function
() {
if (this.checked) {
// Checkbox is checked, show all chargers
showAllChargers();
} else {
// Checkbox is unchecked
clearChargers();
}
});

// Make current location as origin address
map.on('locationfound', function (e) {
```

```
// Update the value of the origin input field with the user's current
location coordinates
const currentLocation = `${e.latlng.lat}, ${e.latlng.lng}`;

// Call reverse geocoding to get the address of the current location
reverseGeocode(e.latlng.lat, e.latlng.lng, function (address) {
if (address) {
originInput.value = address;
} else {
originInput.value = currentLocation;
}
});
});

// Handle errors when getting the user's location
map.on('locationerror', function (e) {
// Log the error message to the console
console.log(e.message);
});

// Event listener to the 'contextmenu' event of the map to open a custom
popup on right-click
map.on('contextmenu', function (event) {
var lat = event.latlng.lat;
var lng = event.latlng.lng;

// Show the custom popup
showCustomPopup(lat, lng);
});

// Taxi icon
const taxiIcon = L.icon({
iconUrl: 'taxi.png',
iconSize: [70, 70],
iconAnchor: [16, 32],
popupAnchor: [0, -32]
});
```

```
// Green marker icon
const greenMarkerIcon = L.icon({
iconUrl: 'green-marker.png',
iconSize: [25, 41],
iconAnchor: [12, 41],
popupAnchor: [1, -34]
});

// Global variables
var origLng, origLat, destLng, destLat; // Longitude and latitude of the
origin and destination
var routeControl = null; // Control route
var routeShown = false; // Track whether the route has been shown on the
map
var markersList = []; // Store origin, route waypoints and destination
var interval, interval1; // Interval variables for periodic tasks
var routeCoordinates = [], routeCoordinates1 = []; // Store coordinates
of the route waypoints
var distanceAlongRoute = 0; // Keep track of distance along the route
var increment = 100; // Increment value for distance along the route
var timeOfInterval = 2000; // Time interval for intervals in milliseconds
var lastAlongRouteLocation = false; // Indicate the last location along
the route
var isPaused = false; // Indicate whether the demo is paused

// Function to show the route on the map
function showRoute() {
var selected_value = document.getElementById("chargersLocation").value;
// console.log(selected_value);

var radius = parseInt(document.getElementById("radius").value);
// console.log(radius);

var distanceAlongRoute1 = 0;
routeCoordinates1 = [];
```

```
// If the route has already been shown, clear it and reset the flag
if (routeShown) {
map.removeControl(routeControl);
routeShown = false;
}

// Remove only markers having the class 'custom-marker' or TaxiIcon
map.eachLayer(function (layer) {
if ((layer instanceof L.Marker && layer.options.icon.options.className
=== 'custom-marker') ||
(layer instanceof L.Marker && layer.options.icon === taxiIcon)) {
map.removeLayer(layer);
}
});

// Get the coordinates for origin
geocode(originInput.value, function (originCoords) {
if (!originCoords) {
alert('Unable to find coordinates for the origin address.');
return;
}

// Get the coordinates for destination
geocode(destinationInput.value, function (destinationCoords) {
if (!destinationCoords) {
alert('Unable to find coordinates for the destination address.');
return;
}

// TaxiMarker
var taxiMarker = L.marker([originCoords[1], originCoords[0]], {
icon: taxiIcon,
draggable: true // Make the marker draggable
}).addTo(map);

// Event listener to update coordinates when marker is dragged
taxiMarker.on('dragend', function (event) {
```

```
var marker = event.target;
var position = marker.getLatLng();
console.log('New Marker Position:', position);
// Call fetchNearbyChargers function with updated coordinates
fetchNearbyChargers(
position.lat,
position.lng,
slider1.value / 100,
slider2.value / 100,
slider3.value / 100,
selected_value,
radius);
});

// Show an alert on the map instructing the user to move the cab
taxiMarker.bindPopup("Drag the cab to find nearby
chargers.").openPopup();

// Create the routing control
routeControl = L.Routing.control({
waypoints: [
L.latLng(originCoords[1], originCoords[0]),
L.latLng(destinationCoords[1], destinationCoords[0])
],
lineOptions: {
styles: [{color: '#1E90FF', weight: 5}] // Change color and width of the
route line
},
zIndex: 9999,
createMarker: function (i, waypoint, n) {
var isOrigin = i === 0,
isDestination = i === n - 1;

if (isOrigin) {
// Create blue marker for the origin
return L.marker(waypoint.latLng, {
icon: L.icon({
```

```
iconUrl: 'blue-marker.png',
iconSize: [25, 41],
iconAnchor: [12, 41],
popupAnchor: [1, -34]
})
});
} else if (isDestination) {
// Create flag marker for the destination
return L.marker(waypoint.latLng, {
icon: L.icon({
iconUrl: 'flag.jpg',
iconSize: [25, 41],
iconAnchor: [12, 41],
popupAnchor: [1, -34]
})
});
}
// Return null for other waypoints
return null;
},
position: 'bottomright'
}).addTo(map);

// Get the route's bounds
var routeBounds = routeControl.getWaypoints().reduce(function (bounds,
waypoint) {
return bounds.extend(waypoint.latLng);
}, L.latLngBounds());

// Adjust the map's view to fit the route bounds with a specific zoom
level
map.setView(routeBounds.getCenter(), 10);

origLng = originCoords[0];
console.log('Origin longitude:', origLng);

origLat = originCoords[1];
```

```
console.log('Origin latitude:', origLat);

destLng = destinationCoords[0];
console.log('Destination longitude:', destLng);

destLat = destinationCoords[1];
console.log('Destination latitude:', destLat);

// Set the flag to indicate that the route has been shown
routeShown = true;
});
});

clearInterval(interval1);

// Set up an interval
interval1 = setInterval(function () {
WriteNodes(distanceAlongRoute1);
distanceAlongRoute1 += increment;

// Check if the end of the route is reached
if (distanceAlongRoute1 >= routeControl._routes[0].coordinates.length) {
clearInterval(interval1); // Stop the interval when the end is reached
}

}, 1); // Every 1 millisecond
}

// Function to geocode an address and get its coordinates
function geocode(address, callback) {
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${address}.json?
access_token=${mapboxAccessToken}`)
.then(response => response.json())
.then(data => {
if (data.features && data.features.length > 0) {
const coordinates = data.features[0].geometry.coordinates;
callback(coordinates);
```

```
} else {
callback(null);
}
})
.catch(error => {
console.error('Error fetching Mapbox Geocoding data:', error);
callback(null);
});
}

// Function to update autocomplete suggestions based on input
function updateAutocomplete(query, inputType) {
const suggestionsElement = document.getElementById(`${inputType}-
suggestions`);

// Clear previous suggestions
suggestionsElement.innerHTML = '';

// Fetch Mapbox Geocoding API for suggestions
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${query}.json?ac
cess_token=${mapboxAccessToken}`)
.then(response => response.json())
.then(data => {
const suggestions = data.features.map(feature => feature.place_name);

// Update suggestions list
suggestions.forEach(suggestion => {
const option = document.createElement('option');
option.value = suggestion;
suggestionsElement.appendChild(option);
});
})
.catch(error => console.error('Error fetching Mapbox Geocoding data:',
error));
}

// Function to write a text file with the nodes
```

```
function WriteNodes(distance) {
// Get the coordinates along the route based on the distance
var coordinates = routeControl._routes[0].coordinates[distance];

// Add coordinates to the routeCoordinates1 array
routeCoordinates1.push(coordinates);

if (routeCoordinates1.length === 0) {
console.log('No coordinates to send.');
return;
}

console.log('All coordinates along the route:', routeCoordinates1);
var nodesList = []

for (var i = 1; i < routeCoordinates1.length + 1; i++) {
nodesList[i] = i + " " + routeCoordinates1[i - 1].lng + " " +
routeCoordinates1[i - 1].lat;
}

nodesList[0] = 0 + " " + origLng + " " + origLat
nodesList[routeCoordinates1.length + 1] = (routeCoordinates1.length + 1)
+ " " + destLng + " " + destLat

console.log('Nodes:', nodesList);
sendNodes(nodesList);
}

// Function to send the nodes list
function sendNodes(nodesList) {
fetch('http://localhost:5000/nodes', {
method: 'POST',
headers: {
'Content-Type': 'application/json',
},
body: JSON.stringify({nodesList: nodesList}),
})
```

```
.then(response => response.json())
.then(data => {
console.log('Data:', data);
})
.catch(error => console.error('Error sending coordinates to server:',
error));
}

// Function to fetch nearby chargers
function fetchNearbyChargers(lat, lng, derouting_cost,
charger_availability, sustainable_charging_level,
selected_value, radius) {

fetch('http://localhost:5000/nearby_chargers', {
method: 'POST',
headers: {
'Content-Type': 'application/json',
},
body: JSON.stringify({
lat: lat,
lng: lng,
derouting_cost: derouting_cost,
charger_availability: charger_availability,
sustainable_charging_level: sustainable_charging_level,
selected_value: selected_value,
radius: radius
}),
})
.then(response => response.json())
.then(data => {
console.log('Data:', data);

// Remove only markers having the class 'custom-marker'
map.eachLayer(function (layer) {
if (layer instanceof L.Marker && layer.options.icon.options.className ===
'custom-marker') {
map.removeLayer(layer);
```

```
}
});

// Add markers for the nearby chargers
addMarkers(data.nearby_chargers);
})
.catch(error => console.error('Error sending coordinates to server:',
error));
}


// Function to add markers to the map
function addMarkers(chargersData) {
console.log('Chargers Data:', chargersData);
chargersData.forEach((chargersGroup, outerIndex) => {
chargersGroup.forEach((charger, innerIndex) => {
const latitude = charger[0];
const longitude = charger[1];
const chargerKW = charger[2];
const energyKWHmin = charger[3];
const energyKWHmax = charger[4];
const availabilityMIN = charger[5];
const availabilityMAX = charger[6];
const name = charger[7];
const ecoChargeScore = charger[8];
const minsToArrive = charger[9];
const availability = charger[10];
const chargingLevel = charger[11];
const derouting = charger[12];
const distance = charger[13];

const marker = L.marker([latitude, longitude], {
icon: L.divIcon({
className: 'custom-marker',
iconSize: [30, 30],
html: `<div>${outerIndex * 5 + innerIndex + 1}</div>`, // Show an index
for each charger
```

```
}),
}).addTo(map);

// Add a popup with charger information
marker.bindPopup(`
Name: ${name}<br>
<hr>
Latitude: ${latitude.toFixed(4)}<br>
Longitude: ${longitude.toFixed(4)}<br>
<hr>
Kilowatts: ${chargerKW}<br>
<hr>
EcoCharge Score: ${ecoChargeScore.toFixed(2)}<br>
<hr>
Availability: ${availability}<br>
Charging Level: ${chargingLevel.toFixed(2)}<br>
Derouting: ${derouting.toFixed(2)}<br>
<hr>
Distance: ${distance.toFixed(2)} km<br>`)
});
});
}

// Function to start the demo
function startDemo() {
var selected_value = document.getElementById("chargersLocation").value;
console.log(selected_value);

var radius = parseInt(document.getElementById("radius").value);
console.log(radius);

distanceAlongRoute = 0;
routeCoordinates = [];

// Check if the route is shown
if (!routeShown) {
alert('Please show the route first.');
```

```
return;
}

// Check if the route control exists and remove it
if (routeControl) {
map.removeControl(routeControl);
routeShown = false; // Reset the flag
}

// Remove all markers
map.eachLayer(function (layer) {
if (layer instanceof L.Marker) {
map.removeLayer(layer);
}
});

showChargersNearbyOrigin();

clearInterval(interval);

// Set up an interval
interval = setInterval(function () {
// Check if the demo is not paused
if (!isPaused) {
// Fetch coordinates along the route
showChargersNearbyWaypointsAndDestination(distanceAlongRoute);

// Increment the distance
distanceAlongRoute += increment;

// Check if the end of the route is reached
if (distanceAlongRoute >= routeControl._routes[0].coordinates.length) {
clearInterval(interval); // Stop the interval when the end is reached
}
}
}, timeOfInterval);
}
```

```
// Function to pause or resume the demo
function togglePauseResume() {
// Check if the route is shown
if (!routeShown) {
alert('Please show the route first.');
return;
}

isPaused = !isPaused;

if (isPaused) {
clearInterval(interval);
} else {
clearInterval(interval);

// Set up an interval
interval = setInterval(function () {
if (!isPaused) {
showChargersNearbyWaypointsAndDestination(distanceAlongRoute);
distanceAlongRoute += increment;

// Check if the end of the route is reached
if (distanceAlongRoute >= routeControl._routes[0].coordinates.length) {
clearInterval(interval); // Stop the interval when the end is reached
}
}
}, timeOfInterval);
}
}

// Function to show chargers nearby route waypoints and destination
function showChargersNearbyWaypointsAndDestination(distance) {
var selected_value = document.getElementById("chargersLocation").value;
// console.log(selected_value);

var radius = parseInt(document.getElementById("radius").value);
```

```
// console.log(radius);

// Last along-route location
if (distance === ((routeCoordinates1.length - 1) * increment)) {
lastAlongRouteLocation = true;
console.log('Last along-route location');
}

// Get the coordinates along the route based on the distance
const coordinates = routeControl._routes[0].coordinates[distance];

// Add coordinates to the routeCoordinates array
routeCoordinates.push(coordinates);

if (routeCoordinates.length === 0) {
console.log('No coordinates to send.');
return;
}

console.log('All coordinates along the route:', routeCoordinates);

// Remove only route waypoints including origin and destination
map.eachLayer(function (layer) {
if (layer instanceof L.Marker && markersList.includes(layer)) {
map.removeLayer(layer);
}
});

// Iterate over routeCoordinates and add the last location as a marker
const lastLocation = routeCoordinates[routeCoordinates.length - 1];
console.log('Last location:', lastLocation);

const marker = L.marker(lastLocation, {icon: taxiIcon}).addTo(map);

// Add the last marker to the markers list
markersList.push(marker);
```

```
console.log('Estimated Components:', slider1.value / 100, slider2.value /
100, slider3.value / 100);

fetchNearbyChargers(
lastLocation.lat,
lastLocation.lng,
slider1.value / 100,
slider2.value / 100,
slider3.value / 100,
selected_value,
radius
);

// Last along-route location
if (lastAlongRouteLocation) {
const marker1 = L.marker([destLat, destLng], {icon:
taxiIcon}).addTo(map);
// Add the destination to the markers list
markersList.push(marker1);

fetchNearbyChargers(
destLat,
destLng,
slider1.value / 100,
slider2.value / 100,
slider3.value / 100,
selected_value,
radius);

console.log('Chargers around destination');
lastAlongRouteLocation = false;
marker.remove();
}
}

// Function to show all chargers
function showAllChargers() {
```

```
var selected_value = document.getElementById("chargersLocation").value;

fetch('http://localhost:5000/chargers', {
method: 'POST',
headers: {
'Content-Type': 'application/json'
},
body: JSON.stringify({selected_value: selected_value}),
})
.then(response => response.json())
.then(chargers => {
var markerCluster = L.markerClusterGroup(); // Create a marker cluster
group

chargers.forEach(charger => {
var marker = L.marker([charger.latitude, charger.longitude], {icon:
greenMarkerIcon})
.bindPopup(charger.name);
markerCluster.addLayer(marker); // Add marker to the cluster group
});

map.addLayer(markerCluster); // Add the marker cluster group to the map
})
.catch((error) => console.error('Error:', error));
}

// Function to clear all chargers
function clearChargers() {
// Iterate through all layers on the map
map.eachLayer(function (layer) {
// Check if the layer is a marker cluster group
if (layer instanceof L.MarkerClusterGroup) {
layer.eachLayer(function (marker) {
// Check if the marker's icon matches greenMarkerIcon
if (marker.options.icon.options.iconUrl ===
greenMarkerIcon.options.iconUrl) {
layer.removeLayer(marker);
```

```
}
});
}
});
}

// Function to show chargers nearby origin
function showChargersNearbyOrigin() {
var selected_value = document.getElementById("chargersLocation").value;
console.log(selected_value);

// Get the coordinates for origin
geocode(originInput.value, function (originCoords) {
if (!originCoords) {
alert('Unable to find coordinates for the origin address.');
return;
}

// Get the coordinates for destination
geocode(destinationInput.value, function (destinationCoords) {
if (!destinationCoords) {
alert('Unable to find coordinates for the destination address.');
return;
}

// Create the routing control
routeControl = L.Routing.control({
waypoints: [
L.latLng(originCoords[1], originCoords[0]),
L.latLng(destinationCoords[1], destinationCoords[0])
],
lineOptions: {
styles: [{color: '#1E90FF', weight: 5}] // Change color and width of the
route line
},
zIndex: 9999,
createMarker: function (i, waypoint, n) {
```

```
var isOrigin = i === 0,
isDestination = i === n - 1;

if (isOrigin) {
// Create blue marker for the origin
return L.marker(waypoint.latLng, {
icon: L.icon({
iconUrl: 'blue-marker.png',
iconSize: [25, 41],
iconAnchor: [12, 41],
popupAnchor: [1, -34]
})
});
} else if (isDestination) {
// Create flag marker for the destination
return L.marker(waypoint.latLng, {
icon: L.icon({
iconUrl: 'flag.jpg',
iconSize: [25, 41],
iconAnchor: [12, 41],
popupAnchor: [1, -34]
})
});
}
// Return null for other waypoints
return null;
},
position: 'bottomright'
}).addTo(map);

// Get the route's bounds
var routeBounds = routeControl.getWaypoints().reduce(function (bounds,
waypoint) {
return bounds.extend(waypoint.latLng);
}, L.latLngBounds());
```

```
// Adjust the map's view to fit the route bounds with a specific zoom
level
map.setView(routeBounds.getCenter(), 10);

// Set the flag to indicate that the route has been shown
routeShown = true;

const marker = L.marker([originCoords[1], originCoords[0]], {icon:
taxiIcon}).addTo(map);

// Add the origin to the markers list
markersList.push(marker);

console.log('Estimated Components:', slider1.value / 100, slider2.value /
100, slider3.value / 100);

fetchNearbyChargers(
originCoords[1],
originCoords[0],
slider1.value / 100,
slider2.value / 100,
slider3.value / 100,
selected_value,
radius
);
console.log('Chargers around origin');
});
});
}

// Function to convert coordinates to address using reverse geocoding
function reverseGeocode(lat, lng, callback) {
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${lng},${lat}.js
on?access_token=${mapboxAccessToken}`)
.then(response => response.json())
.then(data => {
if (data.features && data.features.length > 0) {
```

```
const address = data.features[0].place_name;
callback(address);
} else {
callback(null);
}
})
.catch(error => {
console.error('Error fetching Mapbox Geocoding data:', error);
callback(null);
});
}

// Function to set map view to origin coordinates
function resetMapView() {
// Check if the route is shown
if (!routeShown) {
alert('Please show the route first.');
return;
}

map.setView([origLat, origLng], 8);
}

// Function to handle slider value changes
function updateSlider(value, id) {
console.log(`Slider value for ${id}: ${value}`);
}

// Function to open a custom popup on right-click and show the nearby
chargers
function showCustomPopup(lat, lng) {
var selected_value = document.getElementById("chargersLocation").value;
// console.log(selected_value);

var radius = parseInt(document.getElementById("radius").value);
// console.log(radius);
```

```
// Create a custom popup
var popupContent = `<p>Latitude: ${lat.toFixed(4)}</p><p>Longitude:
${lng.toFixed(4)}</p>
<p><a href="#" id="showNearbyChargers">Show nearby chargers</a></p>`;
var popup = L.popup()
.setLatLng([lat, lng])
.setContent(popupContent)
.openOn(map);

// Attach a click event listener to the "Show nearby chargers" link
document.getElementById("showNearbyChargers").addEventListener('click',
function (event) {
event.preventDefault(); // Prevent the default link behavior

fetchNearbyChargers(
lat,
lng,
slider1.value / 100,
slider2.value / 100,
slider3.value / 100,
selected_value,
radius
);
// Close the popup after clicking the link
map.closePopup(popup);
});
}

// Function triggered when a user signs in using Google Sign-In
function onSignIn(googleUser) {
// Get the basic profile information of the signed-in user
var profile = googleUser.getBasicProfile();
console.log('ID: ' + profile.getId()); // Do not send to your backend!
Use an ID token instead.
console.log('Name: ' + profile.getName());
console.log('Image URL: ' + profile.getImageUrl());
```

```
console.log('Email: ' + profile.getEmail()); // This is null if the
'email' scope is not present.
}


// Function to sign out the user
function signOut() {
var auth2 = gapi.auth2.getAuthInstance();
auth2.signOut().then(function () {
console.log('User signed out.');
});
}

</script>
</body>
</html>
```