ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Βέλτιστη τοποθέτηση σταθμών φόρτισης για ηλεκτρικά οχήματα σε αστικό περιβάλλον

(Locating charging infrastructures for electric vehicles in urban areas)

Υπό

Νασίκα Στυλιανή

Διπλωματική Εργασία

Βόλος, 2022

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής	Δρ. Σαχαρίδης Γεώργιος
(Επιβλέπων)	Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Λυμπερόπουλος Γεώργιος
	Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Χρυσοχόου Ευαγγελία
	Διδάσκουσα, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον αναπληρωτή καθηγητή Δρ. Σαχαρίδη Γεώργιο για την επίβλεψη του κατά την συγγραφή της παρούσας διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε κατά την διάρκεια των φοιτητικών μου χρόνων. Ιδιαίτερα, θα ήθελα να ευχαριστήσω τους αδερφούς μου, Στέργιο, Απόστολο, Θαλή και τον συνάδελφο Χρήστο Καραγιάννη για όλες τις συζητήσεις και τις συμβουλές.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με το πρόβλημα της χωροθέτησης σταθμών φόρτισης (CSLP- Charging Station Location Problem) και συγκεκριμένα με το CSLP που αφορά αστικές μεταφορές, το οποίο αποτελεί ένα σημαντικό ζήτημα στην σύγχρονη εποχή των μεταφορών, καθώς η μετάβαση στην ηλεκτροκίνηση με στόχο την προστασία του περιβάλλοντος ερευνάται όλο και περισσότερο.

Στόχος της διπλωματικής είναι αρχικά η βιβλιογραφική ανασκόπηση των προσεγγίσεων που έχουν χρησιμοποιηθεί για την περιγραφή του προβλήματος και στην συνέχεια η επίλυση προβλημάτων με χρήση δύο προσεγγίσεων.

Δημιουργήθηκε κώδικας στο περιβάλλον της Python και με την βοήθεια του λογισμικού βελτιστοποίησης CPLEXILOG IBM πραγματοποιήθηκε η ανάλυση επτά προβλημάτων για την εύρεση των κατάλληλων θέσεων σταθμών φόρτισης. Για να βρεθεί λύση σε πεπερασμένο χρονικό ορίζοντα, τέθηκε το όριο των 7200 δευτερολέπτων και σε μία περίπτωση έγινε χρήση αλγόριθμου ομαδοποίησης για την ομαλή επίλυση.

Η εργασία αποτελείται από πέντε κεφάλαια. Τα αρχικά κεφάλαια επικεντρώνονται στην βιβλιογραφική ανασκόπηση, ενώ τα δύο τελευταία στην παρουσίαση των δεδομένων που χρησιμοποιήθηκαν και των αποτελεσμάτων του κώδικα.

Αναλυτικά, το πρώτο κεφάλαιο αφιερώθηκε στην ανάλυση των περιβαλλοντικών λόγων που οδηγούν στην ηλεκτροκίνηση και κατ’ επέκταση στην ανάγκη για σταθμούς φόρτισης, στην συνέχεια στην συσχέτιση των εμπορευματικών μεταφορών με τα ηλεκτρικά οχήματα και τέλος στις παραμέτρους που επηρεάζουν την επιλογή των τοποθεσιών των σταθμών φόρτισης. Το δεύτερο κεφάλαιο αναλύει την προσέγγιση του CSLP ως πρόβλημα χωροθέτησης εγκαταστάσεων. Στο τρίτο κεφάλαιο παρουσιάζεται το μοντέλο δρομολόγησης οχημάτων και χωροθέτησης σταθμών. Στο τέταρτο και πέμπτο κεφάλαιο παρατίθενται τα σύνολα των δεδομένων που χρησιμοποιήθηκαν και τα αναλυτικά αποτελέσματα με τον αντίστοιχο σχολιασμό αυτών.

Πίνακας περιεχομένων

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΑΚΡΩΝΥΜΙΩΝ

EV	Ηλεκτρικό όχημα
ICE	Μηχανή εσωτερικής καύσης
ECV	Ηλεκτρικό εμπορευματικό όχημα
CV	Εμπορευματικό όχημα
LDV	Ελαφρό επαγγελματικό όχημα
MDV	Μεσαίου βάρους επαγγελματικό όχημα
HDV	Υψηλού βάρους επαγγελματικό όχημα
CS	Σταθμός φόρτισης
FLP	Πρόβλημα χωροθέτησης
UFLP	Μη περιορισμένης δυναμικότητας FLP
CFLP	Περιορισμένης δυναμικότητας FLP
OD	Διαδρομή Αφετηρίας-Προορισμού
FBM	Προσέγγιση σύμφωνα με την ροή κίνησης
CSLP	Πρόβλημα χωροθέτησης σταθμών φόρτισης
FCLM	Μοντέλο κάλυψης ροής-κίνησης
FRLM	Μοντέλο χωροθέτησης ανεφοδιασμού ροής
BSOC	Μοντέλο παρακολούθησης της κατάστασης της μπαταρίας
LRP	Πρόβλημα δρομολόγησης οχημάτων και χωροθέτησης

Κεφάλαιο 1ο

1.1 Εισαγωγή

Η ζήτηση για μεταφορά προϊόντων σε αστικά κέντρα έχει αυξηθεί τα τελευταία χρόνια και αναμένεται να αυξηθεί ακόμα περισσότερο [1]. Αντίστοιχα, αναμένεται να αυξηθεί ο αριθμός των οχημάτων που εκτελούν καθημερινά δρομολόγια για διανομές προϊόντων. Σύμφωνα με την Ευρωπαϊκή Επιτροπή, η οδική μεταφορά ευθύνεται για περίπου το 20% των συνολικών εκπομπών διοξειδίου του άνθρακα στην Ευρώπη (15% ελαφρά οχήματα μεταφορών, 5% βαρέα οχήματα μεταφορών) [2]. Έτσι, η Ε.Ε. έχει θέσει αυστηρούς στόχους για την μείωση των εκπομπών ρύπων λόγω μεταφορών. Συγκεκριμένα, προβλέπει μείωση των εκπομπών αερίων του θερμοκηπίου στο επίπεδο του 20% σε σύγκριση με το 2008 για το 2030 και για εκπομπές μεταφορών στο 60% σε σύγκριση με το 1990 για το 2050 [3]. Είναι απαραίτητο, άρα, να βρεθούν ρεαλιστικές και βιώσιμες εναλλακτικές για την μεταφορά προϊόντων που να συνάδουν με τους περιβαλλοντικούς στόχους και πολιτικές που έχουν θεσπιστεί.

Στον τομέα των μεταφορών, σημαντική έρευνα έχει διεξαχθεί πάνω στην χρήση ηλεκτρικών οχημάτων (EV) ως μέτρο αντιμετώπισης της κλιματικής αλλαγής και της ατμοσφαιρικής ρύπανσης των αστικών περιοχών. Η επιστημονική βιβλιογραφία εντοπίζει ότι τα οφέλη της χρήσης των EV προκύπτουν από την μείωση της κλιματικής αλλαγής, των ρύπων και των δεικτών περιβαλλοντικής υγείας στα πυκνοκατοικημένα αστικά κέντρα, όπου ταυτόχρονα η χρήση ηλεκτρικών οχημάτων έχει το μεγαλύτερο αντίκτυπο [3]. Το ποσοστό των ηλεκτρικών οχημάτων στην αγορά δεν είναι τόσο, όσο θα ήταν αναμενόμενο στην μοντέρνα εποχή των μεταφορών. Σταδιακά, η παρουσία τους στους στόλους οχημάτων έχει αρχίζει να υιοθετείτε ως μέτρο έναντι στην κλιματική αλλαγή. Πρόκειται για ένα παγκόσμιο φαινόμενο, το οποίο είναι ακόμα στα αρχικά του στάδια, καθώς είναι απαραίτητο να ξεπεραστούν αρκετά εμπόδια και προκλήσεις προκειμένου να μπορέσουν τα ηλεκτρικά οχήματα να ανταγωνιστούν τα οχήματα με μηχανές εσωτερικής καύσης (Internal combustion engines, ICE).

Παρόλο που τα ηλεκτρικά οχήματα ξεκίνησαν να κερδίζουν έδαφος στην αγορά την τελευταία 10ετια, δεν αποτελούν καινούρια έννοια ή σχέδιο. Στο ξεκίνημα της αυτοκινητοβιομηχανίας τα ηλεκτρικά οχήματα ήταν πιο δημοφιλή απ' ότι τα οχήματα με μηχανές εσωτερικής καύσης. Στην συνέχεια όμως γύρω στο 1940, τα οχήματα ICE ξεπέρασαν τις υπόλοιπες κατηγορίες και εδραιώθηκαν στον χώρο [3]. Πλέον, τόσο η αυτοκινητοβιομηχανία όσο και οι εθνικές κυβερνήσεις έχουν εκφράσει το ενδιαφέρον τους για την υιοθέτηση των ηλεκτρικών οχημάτων για μεταφορές με στόχο την μείωση ρύπων, της κατανάλωσης ενέργειας και ορυκτών πόρων.

Τα EV παρουσιάζουν πολλές δυνατότητες για την αντιμετώπιση της παγκόσμιας κλιματικής αλλαγής. Έτσι, αξίζει να ερευνηθεί η χρήση των EV στις αστικές μεταφορές ως μια εναλλακτική μορφή μετακίνησης. Η περιορισμένη αυτονομία των EV, καθιστά απαραίτητο να υπάρχουν οι κατάλληλες υποδομές φόρτισης κατά μήκος των

διαδρομών. Ένας από τους κυριότερους λόγους που οι επενδυτές παραμένουν επιφυλακτικοί προς την επιστράτευση ηλεκτρικών οχημάτων για μεταφορές κάθε είδους είναι το λεγόμενο άγχος εμβέλειας (range anxiety). Τα ηλεκτρικά αυτοκίνητα λόγω τις περιορισμένης αυτονομίας τους, σε σύγκριση με τα αντίστοιχα ICE οχήματα, απαιτούν πιο συχνές στάσεις για επαναφόρτιση οι οποίες συγχρόνως είναι και μεγαλύτερης διάρκειας. Επίσης, καίριο ρόλο στην αποδοχή των ηλεκτρικών αυτοκινήτων, κατέχουν οι σταθμοί φόρτισης που είναι διαθέσιμοι και λειτουργούν στο οδικό δίκτυο. Συνεπώς, για την χρήση των EV για Ι.Χ. όσο και για εμπορευματικές μεταφορές (electric commercial vehicle, ECV) είναι απαραίτητο να εγκατασταθούν σε κατάλληλες τοποθεσίες σταθμοί φόρτισης, ώστε να είναι δυνατό να μειωθεί το range anxiety των οδηγών αλλά και να δοθεί η δυνατότητα στα οχήματα να εκτελούν τις διαδρομές τους χωρίς περιορισμούς εμβέλειας ενώ ταυτόχρονα θα διατηρείται το χρονοδιάγραμμα των μετακινήσεων.

1.2 Εμπορευματικές μεταφορές και ηλεκτρικά οχήματα

Οι αστικές εμπορευματικές μεταφορές έχουν προσδιοριστεί ως η διαδικασία και το σύστημα της συγκέντρωσης, μεταφοράς και διανομής προϊόντων μέσα στα όρια ενός αστικού κέντρου . Στο πλαίσιο της βιώσιμης ανάπτυξης των αστικών κέντρων, είναι απαραίτητο να εφαρμοστούν μέτρα στον τομέα των μεταφορών, στοχεύοντας στην μείωση των εκπομπών καυσαερίων και θορύβου. Αναλύσεις έχουν δείξει πως η ηλεκτροκίνηση έχει τις προοπτικές και δυνατότητες να είναι μια κατάλληλη εναλλακτική σε σύγκριση με τα συμβατικά οχήματα μεταφορών [4],[5].

Τα επαγγελματικά οχήματα μεταφορών (Commercial Vehicles, CV) κατατάσσονται ευρέως σε τρείς κύριες κατηγορίες με κριτήριο το μικτό βάρος του οχήματος. Οι κατηγορίες αυτές είναι τα ελαφρά επαγγελματικά οχήματα(Light Duty Vehicles, LDV) με μεικτό βάρος μικρότερο από 3,5 τόνους, τα μεσαία επαγγελματικά οχήματα(Medium Duty Vehicles, MDV) με μικτό βάρος από 3,5 έως 12 τόνους και τέλος τα βαρέα επαγγελματικά οχήματα(Heavy Duty Vehicles, HDV) με μικτό βάρος μεγαλύτερο από 12 τόνους [6]. Έχει αποδειχθεί πως η μετάβαση των LDV στην ηλεκτροκίνηση είναι δυνατή, χωρίς να υπάρχουν σημαντικές αλλαγές στην οδική συμπεριφορά τους και την καθημερινή τους χρήση. Οι εξελίξεις στην τεχνολογία των μπαταριών και των κυψελών καυσίμου καθιστούν πλέον εφικτή την χρήση των EV για περεταίρω εφαρμογές. Τα MDV και HDV χρησιμοποιούνται για παραδώσεις, συλλογή απορριμμάτων και άλλες γενικές αποστολές μεγάλων αποστάσεων. Είναι σημαντικό, να εξεταστεί η χρήση ηλεκτρικών οχημάτων αυτών των κατηγοριών για μείωση των ρύπων, ειδικά σε αστικό περιβάλλον.

Στην Ευρώπη, το 75% του πληθυσμού κατοικεί στις πόλεις και στα αστικά κέντρα [7] με αποτέλεσμα, πληθώρα εργασιών και ενεργειών να λαμβάνουν χώρα στο πλαίσιο του αστικού ιστού. Ανάμεσα σε αυτές τις δραστηριότητες, οι εμπορευματικές μεταφορές είναι ένας σημαντικός παράγοντας που συμβάλλει στην ρύπανση της ατμόσφαιρας και στην εκπομπή αερίων του θερμοκηπίου, καθώς οι αστικές μεταφορές είναι απαραίτητες για να προμηθεύονται οι κάτοικοι προϊόντα. Τα ηλεκτρικά οχήματα μπορούν να είναι μια καλή εναλλακτική για τις μεταφορές καθώς δεν εκπέμπουν τοπικά ρύπους και συμβάλλουν σημαντικά στην μείωση της ηχορύπανσης. Η αυτονομία των ECV μπορεί να μην είναι επαρκής στις περιπτώσεις που θα πρέπει να εκτελούνται δρομολόγια μεγάλων αποστάσεων ή ακόμα πολλαπλά κυκλικά δρομολόγια. Έτσι, η βέλτιστη τοποθέτηση σταθμών φόρτισης αποτελεί μια απόφαση μεγάλης σημασίας ώστε να διασφαλιστεί η business as usual (BAU) λειτουργία των επιχειρήσεων κάτω υπό τους νέους περιβαλλοντολογικούς περιορισμούς.

Η εμβέλεια των ECV εξαρτάται από διάφορους παράγοντες, τόσο επιχειρησιακούς όσο και λόγω καιρικών συνθηκών. Αρχικά, τα CV συνήθως λειτουργούν πολλαπλές βάρδιες ημερησίως, χρειάζεται να κάνουν πολλές στάσεις για την εξυπηρέτηση πελατών και καλύπτουν μεγάλες χιλιομετρικές αποστάσεις που μπορεί να ξεπερνάει την αυτονομία της μπαταρίας ενός ECV. Επιπροσθέτως, καθώς τα EV λειτουργούν με

μπαταρίες, πιθανές ακραίες θερμοκρασίες επηρεάζουν την εμβέλεια τους μειώνοντας την χιλιομετρική απόσταση που θα μπορούσαν να καλύψουν με μία πλήρη φόρτιση. Αυτό έχει επιβεβαιωθεί από έρευνα της Αμερικανικής Ένωσης Αυτοκινήτων, κατά την οποία οι χαμηλές θερμοκρασίες περιβάλλοντος μπορούν να μειώσουν σημαντικά την εμβέλεια των ηλεκτρικών οχημάτων [8]. Κατά μέσω όρο η εμβέλεια των EV μπορεί να μειωθεί έως και κατά 41% σε ατμοσφαιρικές θερμοκρασίες κάτω των -6^{o}C καθώς η χρήση του συστήματος θέρμανσης καταναλώνει μεγάλο μέρος της μπαταρίας των οχημάτων. Αντίστοιχα σε θερμοκρασίες άνω των 35 ^{o} C, το εύρος των EV οχημάτων μπορεί να μειωθεί κατά 17% εξαιτίας πάλι της χρήσης του κλιματισμού. Έτσι, είναι ιδιαίτερα σημαντικό να εξεταστούν αυτοί οι παράγοντες όταν γίνεται η μετάβαση από τα συμβατικά οχήματα στα ECV ώστε σταθμοί φόρτισης να τοποθετηθούν σε στρατηγικές θέσεις και να διασφαλιστεί η καθημερινή λειτουργία του στόλου οχημάτων.

Η χρήση των ECV από εταιρείες είναι μια εξέλιξη στον τομέα των logistics που έχει συζητηθεί αρκετά τα τελευταία χρόνια, εξαιτίας των μέτρων που λαμβάνουν οι πολιτείες για την μείωση των ρύπων στα αστικά κέντρα. Σε πολλές μεγαλουπόλεις υπάρχουν ήδη καθορισμένες περιοχές όπου η πρόσβαση ορισμένων ρυπογόνων οχημάτων περιορίζεται ή αποτρέπεται με στόχο τη βελτίωση της ποιότητας του αέρα (Low emission zone LEZ, zero emission zone ZEZ) Τέτοιου είδους πολιτικές ευνοούν την χρήση οχημάτων με εναλλακτικά καύσιμα, υβριδικά και ηλεκτρικά οχήματα.

Χώρες όπως η Αυστρία, η Γερμανία και η Δανία συμμετέχουν σε δράσεις που αφορούν την προώθηση των ηλεκτρικών οχημάτων εμπορικών μεταφορών και είναι από τις πρώτες χώρες που ξεκίνησαν να χρησιμοποιούν ECV [9]. Συγκεκριμένα, στην Αυστρία υπάρχουν αρκετές εταιρίες διανομών και μεταφορών οι οποίες χρησιμοποιούν καθημερινά EV, στοχεύοντας στις CO_2 ουδέτερες διανομές. Στην Γερμανία από τον Ιανουάριο του 2021 περίπου το 1,2% των οχημάτων σε κυκλοφορία είναι πλέον ηλεκτρικά ή υβριδικά, τον Ιούνιο του ίδιου έτους επιτεύχθηκε ο στόχος των ενός εκατομμυρίου ηλεκτρικών αυτοκινήτων εν λειτουργία που είχε τεθεί το 2010, με την πλειοψηφία αυτών να είναι επιβατικά αυτοκίνητα και τον αριθμό των LDV να ξεπερνάει τις 22 χιλιάδες [10].

Τα CV κατά μέσο όρο είναι μεγαλύτερο ποσοστό του χρόνου στο οδικό δίκτυο. Επομένως, το οικολογικό αντίκτυπο που έχουν στην ατμόσφαιρα, είναι σημαντικά υψηλότερο σε σύγκριση με αυτό των I.X.. Σύμφωνα με την έρευνα FREVUE, στην αρχή των πειραμάτων το 39% των χειριστών στόλων οχημάτων μεταφορών πίστευαν ότι τα ηλεκτρικά βαν και φορτηγά είναι μια βιώσιμη εναλλακτική για τα αντίστοιχα ντιζελοκίνητα οχήματα. Μετά το πέρας των δοκιμών, το ποσοστό αυτό ανέβηκε στο 72%. Συνεπώς, είναι εύλογο το συμπέρασμα ότι τα ECV είναι μια βιώσιμη και καλή εναλλακτική για τις μεταφορές προϊόντων ενώ συγχρόνως μειώνονται οι επιβλαβείς ρύποι που απελευθερώνονται στο περιβάλλον.

Ανακεφαλαιώνοντας, τα ECV μπορούν να αποτελέσουν μια εξαιρετική εξέλιξη στις μεταφορές φορτίων στον αστικό ιστό, ενώ σε πολλές χώρες ήδη έχουν υιοθετηθεί ως πρακτική. Σε αυτή την μετάβαση από την βενζινοκίνηση ή πετρελαιοκίνηση στην

ηλεκτροκίνηση, ιδιαίτερα σημαντικό είναι να εξασφαλιστεί η αδιάκοπη λειτουργία των επιχειρήσεων και ταυτόχρονα να λαμβάνονται υπόψιν οι περιορισμοί και οι απαιτήσεις τόσο των επιχειρήσεων όσο και των ίδιων των ECV. Έτσι, η βέλτιστη και στρατηγική λήψη αποφάσεων για την εγκατάσταση σταθμών φόρτισης είναι απαραίτητη.

1.3 Τοποθέτηση σταθμών φόρτισης.

Όπως έχει ήδη αναφερθεί, ο τομέας των μεταφορών ευθύνεται για ένα μεγάλο ποσοστό των εκπομπών CO_2 παγκοσμίως. Έτσι, μέτρα έχουν ληφθεί για να προωθηθεί η χρήση «πράσινων» οχημάτων για να επιτευχθεί μείωση των ρύπων. Η διάδοση των EV επιταχύνεται όλο και περισσότερο τα τελευταία χρόνια, ωστόσο οι αυξημένες τιμές, η περιορισμένη αυτονομία και η ελλιπείς υποδομές φόρτισης παραμένουν εμπόδιο στην εκτενέστερη υιοθέτηση τους.

Ο αριθμός των EV αναμένεται να αυξηθεί παγκοσμίως από 1 εκατομμύριο το 2020 στα 12 εκατομμύρια έως το 2040 [11]. Τα νεότερα μοντέλα EV μπορούν πλέον να διανύσουν σημαντικά περισσότερα χιλιόμετρα με μία πλήρης φόρτιση απ' ότι προηγουμένως, εξακολουθούν όμως να έχουν μικρότερη αυτονομία από αυτή των αντίστοιχων ICE αυτοκινήτων. Συνδυάζοντας τον αυξανόμενο αριθμό EV και την περιορισμένη εμβέλεια τους, προκύπτει μεγάλη ανάγκη για τοποθέτηση σταθμών φόρτισης σε πολλές περιοχές ανά τον κόσμο. Προκειμένου να διασφαλιστεί η σωστή λειτουργία ενός τέτοιου δικτύου, είναι ιδιαίτερα σημαντικό να εκτελείται έρευνα πριν την εγκατάσταση των σταθμών φόρτισης ώστε να γίνει η βέλτιστη τοποθέτηση τους.

Αρχικά, χρειάζεται να επιλεγούν οι κατάλληλες εν δυνάμει θέσεις στο οδικό δίκτυο για να τοποθετηθούν οι σταθμοί φόρτισης (Charging Station, CS). Τοποθεσίες δηλαδή, όπου τα EV θα μπορούν να έχουν άμεση πρόσβαση και χώρο στάθμευσης. Συνήθως, οι CS συναντώνται σε κατοικίες, στο οδικό δίκτυο, σταθμούς στάθμευσης οχημάτων, τοποθεσίες ενδιαφέροντος όπως μουσεία και εμπορικά κέντρα, προ υπάρχοντες σταθμούς ανεφοδιασμού καυσίμων. Μια ακόμα κατηγορία πιθανών τοποθεσιών προκύπτει από την χρήση EV για εμπορευματικές μεταφορές και αφορά την εγκατάσταση CS για ECV. Είναι τοποθεσίες πελατών και τοποθεσίες όπου γίνεται στάθμευση ανάμεσα στα διαλλείματα και τις βάρδιες. Κατά την διάρκεια των δρομολογίων τους, τα ECV μπορούν να φορτίζουν ενώ εξυπηρετούν τους πελάτες ή/και κατά την διάρκεια των διαλλειμάτων των.

Επίσης, να αναφερθεί πως οι CS χωρίζονται σε δύο γενικές κατηγορίες τους ιδιωτικούς και τους δημόσιους ανάλογα με το εάν είναι προσβάσιμοι και από τον γενικό πληθυσμό ή μόνο από την εταιρεία που χρηματοδότησε την εγκατάσταση τους [12].

Κατά το στάδιο μελέτης της εγκατάσταση των σταθμών φόρτισης θα πρέπει να ληφθούν υπόψιν και οικονομικοί περιορισμοί. Τόσο η αγορά του φορτιστή όσο και η εγκατάσταση του, έχουν μεγάλο κόστος. Η αξία του φορτιστή διαφοροποιείται ανάλογα με τον τύπο του φορτιστή. Υπάρχουν τρείς τύποι φορτιστών. Ο πρώτος είναι ο αργός φορτιστής (slow charger) ο οποίος πρακτικά είναι αυτός που χρησιμοποιείται σε σπίτια και σε σταθμούς ή αφετηρίες οχημάτων μεταφορών, όπου τα EV μπορούν να σταθμεύσουν για μεγάλο χρονικό διάστημα για να φορτίσει η μπαταρία τους. Στην συνέχεια είναι ο 2^{ου} επιπέδου που είναι πιο γρήγορος απ' ότι ο 1^{ου} επιπέδου. Τέλος,

υπάρχει και ο φορτιστής 3ου επιπέδου (Fast charger) ο οποίος είναι ο πιο γρήγορος φορτιστής και μέσα σε 20-30 λεπτά έχει φορτιστεί το 80% της μπαταρίας του EV.

Charging Level Summary

Level	ChargeHub Markers	Power (kW)	Approximate Charging Time (Empty Battery)
1		1	200 km (124 miles): +/- **20 hours** 400 km (249 miles): +/- **43 hours**
2		3 to 20, typically 6	200 km (124 miles): +/- **5 hours** 400 km (249 miles): +/- **11 hours**
3 (DCFC)		Typically 50, occasionaly 20	80% of 200 km (124 miles): +/- **30 min** 80% of 400 km (249 miles): +/- **1 hour**

Εικόνα 1 Κατηγορίες και χαρακτηριστικά φορτιστών EV [13].

Ένα ακόμα κριτήριο που πρέπει να εξεταστεί κατά την τοποθέτηση CS είναι η ικανότητα του δικτύου παροχής ηλεκτρικής ενέργειας να ανταπεξέλθει στην ζήτηση για φόρτιση. Κατά την φόρτιση των EV απαιτούνται μεγάλα ποσά ηλεκτρικής ενέργειας και για αυτό θα πρέπει να γίνεται μελέτη κατά την εγκατάσταση των CS για να είναι βέβαιο ότι το δίκτυο μπορεί να παρέχει το ζητούμενο ρεύμα ακόμα ειδικά σε περιόδους μεγάλης ζήτησης χωρίς να διακινδυνεύετε η ομαλή λειτουργία του [6]. Αυτού του είδους η μελέτη είναι πολύ σημαντική και για οικονομικούς λόγους, καθώς για στρατηγικούς λόγους μπορεί να έχει επιλεγεί μια συγκεκριμένη τοποθεσία για να καλύπτει μεγάλο ποσοστό της κίνησης EV, αλλά το προ υπάρχον δίκτυο να μην είναι ικανό να καλύψει την ζήτηση. Με αποτέλεσμα να απαιτείται η αναβάθμιση του για να εγκατασταθεί στην συνέχεια ο CS.

Ο σχεδιασμός για την τοποθέτηση σταθμών φόρτισης έχει στρατηγική επίδραση πάνω στην προώθηση και υιοθέτηση των ηλεκτρικών οχημάτων [9], [14]. Έρευνες έχουν δείξει πως η έλλειψη διαθέσιμων σταθμών φόρτισης στο παρόν οδικό δίκτυο επηρεάζει σε μεγάλο βαθμό την αγορά EV. Ο λόγος είναι η αβεβαιότητα που αισθάνονται οι οδηγοί των EV για το εάν θα μπορέσουν να ολοκληρώσουν την διαδρομή τους ή να φτάσουν στον επόμενο διαθέσιμο σταθμό φόρτισης πριν εξαντληθεί η μπαταρία του αυτοκινήτου (range anxiety). Άρα, με την τοποθέτηση CS σε κατάλληλες τοποθεσίες θα υπάρχουν περισσότερες επιλογές για τους οδηγούς για επαναφόρτιση και περισσότεροι άνθρωποι θα επιλέγουν τα EV.

Σε αυτό το σημείο, χρειάζεται να γίνει ο διαχωρισμός ανάμεσα στις απαιτήσεις για φόρτιση των IX EV και των ECV. Στις περισσότερες περιπτώσεις οι αποστάσεις που διανύουν καθημερινά τα IX EV δεν ξεπερνούν την αυτονομία τους. Άρα, οι ιδιοκτήτες μπορούν να τα φορτίζουν, κυρίως κατά την διάρκεια της νύχτας στην παροχή ηλεκτρικής ενέργειας του σπιτιού τους χωρίς να ανησυχούν για τις μετακινήσεις τους. Η απαίτηση των IX EV για φόρτιση στο οδικό δίκτυο, συνήθως προκύπτει κατά την διάρκεια υπεραστικών ταξιδιών. Από την άλλη, τα ECV διανύουν καθημερινά μεγάλες αποστάσεις και το μεγαλύτερο ποσοστό του χρόνου βρίσκονται στο οδικό δίκτυο εκτελώντας δρομολόγια. Ως αποτέλεσμα, έχουν μεγάλες απαιτήσεις για φόρτιση και χρειάζονται ενδιάμεσες στάσεις για να μπορέσουν να συνεχίσουν τα δρομολόγια τους.

Το πρόβλημα της «Βέλτιστης τοποθέτησης σταθμών φόρτισης» (Charging Station Location Problem – CSLP) αφορά την βέλτιστη τοποθέτηση CS που ικανοποιεί όσο το δυνατόν μεγαλύτερο ποσοστό της ζήτησης για φόρτιση, ενώ ταυτόχρονα τηρούνται οικονομικοί και λειτουργικοί περιορισμοί. Η ζήτηση για φόρτιση θεωρείται ότι έχει ικανοποιηθεί όταν οι οδηγοί των EV μπορούν να ολοκληρώσουν τις διαδρομές τους χωρίς να εξαντληθεί η μπαταρία του οχήματος. Υπάρχει πλήθος επιστημονικών δημοσιεύσεων που αφορούν το CSLP. Οι πιο συνηθισμένες προσεγγίσεις του CSLP ανήκουν στην κατηγορία των προβλημάτων χωροθέτησης (Facility Location Problems-FLP) και αφορούν την τοποθέτηση δημόσιων CS. Υπάρχει και η προσέγγιση του CSLP ως πρόβλημα ταυτόχρονης χωροθέτησης και δρομολόγησης οχημάτων (Location Routing Problem-LRP), που αφορά την βέλτιστη τοποθέτηση CS συγκεκριμένα για την κάλυψη των αναγκών των ECV.

Στα κεφάλαια 2 και 3 θα παρουσιαστούν αναλυτικά οι προσεγγίσεις που έχουν αναφερθεί και θα αναλυθεί σε κάθε κατηγορία το βασικό μαθηματικό μοντέλο. Στην συνέχεια, τα επόμενα κεφάλαια επικεντρώνονται στην παρουσίαση και σύγκριση των αποτελεσμάτων από δύο μορφοποιήσεις. Θα παρουσιαστούν και θα σχολιαστούν παραδείγματα διαφορετικών μεγεθών που έχουν επιλυθεί πρώτα με3.1 Δρομολόγηση οχημάτων και χωροθέτηση σταθμών φόρτισης. και μετέπειτα με το 2.2.1 Μοντέλο κάλυψης ροής-κίνησης (Flow capturing location model-FCLM).

Κεφάλαιο 2ο

2.1 Πρόβλημα βέλτιστης χωροθέτησης εγκαταστάσεων (Facility Location Problems)

Η μελέτη των προβλημάτων βέλτιστης χωροθέτησης εγκαταστάσεων (Facility Location Problems, FLP), είναι κλάδος της επιχειρησιακής έρευνας και ασχολείται με τη βέλτιστη επιλογή τοποθεσιών για εγκαταστάσεις [15]. Ένα FLP επιδιώκει να εντοπίσει τις βέλτιστες τοποθεσίες ανάμεσα στις πιθανές θέσεις, με στόχο την ελαχιστοποίηση του κόστους ανάθεσης ή την μεγιστοποίηση της κάλυψης της ζήτησης. Ταυτόχρονα, θα πρέπει να λαμβάνονται υπόψη περιορισμοί που προκύπτουν από τον προσδιορισμό του εκάστοτε προβλήματος.

Το πρόβλημα εγκατάστασης CS (CSLP) αποτελεί ένα FLP. Η πιο απλή εκδοχή του είναι η επιλογή των θέσεων ακριβώς p εγκαταστάσεων ώστε να ελαχιστοποιείται το συνολικό κόστος ($p-median$). Σε άλλες περιπτώσεις, ο αριθμός των εγκαταστάσεων που θα τοποθετηθούν είναι μια εσωτερική απόφαση, που λαμβάνεται κατά την επίλυση του προβλήματος. Αυτού του είδους τα προβλήματα κατατάσσονται στα προβλήματα μη περιορισμένη δυναμικότητας - Uncapacitated FLP (UFLP). Στα $p-median$ και UFLP ο κάθε πελάτης/ζήτηση ανατίθεται στην εγκατάσταση που ελαχιστοποιεί το κόστος ανάθεσης του. Από τις σημαντικότερες επεκτάσεις του UFLP είναι τα προβλήματα περιορισμένης δυναμικότητας - Capacitated FLP (CFLP), στα οποία εξωτερικοί παράγοντες προσδιορίζουν την μέγιστη ζήτηση που μπορεί να ικανοποιηθεί από κάθε πιθανή θέση εγκατάστασης[16]. Εφαρμογές των παραπάνω προβλημάτων/μοντέλων για την βέλτιστη τοποθέτηση CS έχουν παρατηρηθεί στην ερευνητική κοινότητα. Πιο αναλυτικά, εφαρμογή του $p-median$ προβλήματος παρατηρείται στους [17], του UFLP στους [18] και του CFLP στους [19].

Ακόμα ένα συνηθισμένο μοντέλο για την τοποθέτηση CS είναι το πρόβλημα μέγιστης κάλυψης (maximal covering problem), όπου αναζητείται η μέγιστη κάλυψη της ζήτησης που μπορεί να επιτευχθεί δεδομένης μιας χιλιομετρικής απόστασης εξυπηρέτησης ή χρόνου εξυπηρέτησης όπως και περιορισμένου αριθμού CS. Κατά την επίλυση του προβλήματος μέγιστης κάλυψης, προσδιορίζεται τόσο η μέγιστη τιμή της ζήτησης που μπορεί να καλυφθεί, όσο και οι τοποθεσίες των εγκαταστάσεων που ικανοποιούν αυτή την ζήτηση [20]. Η εφαρμογή του προβλήματος μέγιστης κάλυψης για την τοποθέτηση CS σε αστικό περιβάλλον μπορεί να εντοπιστεί στους [21]. Τα προβλήματα κάλυψης συνόλου (set covering problem), είναι ένα ακόμα μοντέλο που έχει χρησιμοποιηθεί για την μελέτη της βέλτιστης τοποθέτησης CS. Σε αυτά τα προβλήματα, ζητείται να προσδιοριστεί η θέση και το πλήθος των θέσεων έτσι ώστε κάθε σημείο ζήτησης, να καλύπτεται από τουλάχιστον ένα σημείο εξυπηρέτησης. Οι [22] έχουν χρησιμοποιήσει την παραπάνω προσέγγιση για τον προσδιορισμό των θέσεων σταθμών ανεφοδιασμού οχημάτων.

Για τα CSLP εξετάζονται οι περιπτώσεις όπου το μοντέλο περιλαμβάνει και την απόφαση του τύπου του φορτιστή που θα εγκατασταθεί στην επιλεγόμενη θέση. Οι [22] χρησιμοποιούν μόνο ενός τύπου φορτιστών, τους FS καθώς είναι πιο δημοφιλής για καθημερινά ταξίδια. Αντιθέτως, οι [23] και [24] περιλαμβάνουν στα μοντέλα τους την απόφαση για το τί τύπου φορτιστές θα εγκατασταθούν στην κάθε εγκατάσταση.

Τα CSLP μπορούν να ταξινομηθούν ανάλογα με την προσέγγιση που γίνεται για τον σχεδιασμό του προβλήματος. Τα μοντέλα στα οποία η ζήτηση τοποθετείται σε σημεία, για παράδειγμα σημεία που αντιπροσωπεύουν σπίτια ή χώρους εργασίας, ονομάζονται μοντέλα βασισμένα σε σημεία-κόμβους (node base models) [25],[14]. Η προσέγγιση αυτή λαμβάνει υπόψιν την αφετηρία και τον προορισμό των ταξιδιών των οχημάτων για τον προσδιορισμό των πιθανών θέσεων των CS. Συνήθως, η μελέτη με node-base μοντέλα αφορά εγκαταστάσεις slow chargers όπου η φόρτιση θα γίνεται κατά την διάρκεια μερικών ωρών όσο το όχημα βρίσκεται στο σπίτι, στον χώρο εργασίας ή σε κάποιο εμπορικό κέντρο. Χαρακτηριστικά των κόμβων, όπως πλήθος οχημάτων που επισκέπτονται το σημείο και η ζήτηση για στάθμευση οχημάτων, είναι απαραίτητα για τον προσδιορισμό των θέσεων των CS [25]. Η ζήτηση υπολογίζεται ως άθροισμα των οδηγών που επιθυμούν να κάνουν στάση για επαναφόρτιση στα σημεία που αναφέρθηκαν προηγουμένως.

Σε αντίθεση με τα μοντέλα των κόμβων, που εξετάζουν μεμονωμένα και συγκεκριμένα σημεία των διαδρομών ως πιθανές θέσεις τοποθέτησης CS, υπάρχουν μοντέλα που εξετάζουν ολόκληρη την διαδρομή του οχήματος, τα μοντέλα ροής-κίνησης οχημάτων (flow-base models, FBM). Τα flow-based models αποτελούν την πλειοψηφία των μοντελοποιήσεων που χρησιμοποιούνται για CSLP, όπως παρατηρείται και από πρόσφατη έρευνα [14]. Αξίζει, λοιπόν, να γίνει μια εκτενής αναφορά σε αυτή την προσέγγιση καθώς η μέθοδος που χρησιμοποιείται για τον προσδιορισμό της ζήτησης για φόρτιση αντιπροσωπεύει καλύτερα την πραγματικότητα, μιας και εξαρτάται από τις διαδρομές που ακολουθούν τα EV στα ταξίδια τους. Στα υπό-κεφάλαια που ακολουθούν, θα αναλυθεί το μοντέλο ροής-κίνησης οχημάτων, θα παρουσιαστούν οι κατηγορίες αυτού και αντίστοιχα παραδείγματα για την κάθε προσέγγιση.

2.2 Προσέγγιση σύμφωνα με την ροή κίνησης (Flow based models, FBM)

Τα FBM είναι μια σχετικά πρόσφατη προσέγγιση στο πρόβλημα της τοποθέτησης CS. Στα FBM, η ζήτηση αναπαρίσταται ως ένα σύνολο από διαδρομές Αφετηρίας – Προορισμού (Origin-Destination, OD), σε αντίθεση με τα κλασσικά μοντέλα χωροθέτησης εγκαταστάσεων όπου η ζήτηση συγκεντρώνεται σημειακά σε κόμβους του δικτύου. Το σκεπτικό πίσω από αυτού του είδους των μοντελοποιήσεων είναι ότι τα EV όταν διανύουν μεγάλες αποστάσεις, εξαιτίας της περιορισμένης εμβέλειας που προσφέρεται λόγω των μπαταριών τους, χρειάζεται να κάνουν ενδιάμεσες στάσεις κατά μήκος των διαδρομών τους για επαναφόρτιση. Έτσι, η ζήτηση για επαναφόρτιση μπορεί να εκπροσωπηθεί από ένα σύνολο ταξιδιών OD, όπου CS θα πρέπει να τοποθετηθούν σε κατάλληλες θέσεις, ώστε η απόσταση μεταξύ δύο συνεχόμενων σταθμών να μην ξεπερνάει την αυτονομία των EV και οι οδηγοί να μπορούν να ταξιδέψουν στον προορισμό τους χωρίς να εξαντληθεί η μπαταρία του EV [14]

Υπάρχει πληθώρα ερευνών για την βέλτιστη τοποθέτηση CS με χρήση FBM. Οι έρευνες αυτές αφορούν την τοποθέτηση δημόσιων σταθμών φόρτισης, χρησιμοποιώντας δεδομένα που να εκφράζουν την ροή κίνησης οχημάτων στο οδικό δίκτυο. Οπότε, η τοποθέτηση των CS στοχεύει στην εξυπηρέτηση του συνόλου των οχημάτων, τόσο των IX όσο και των οχημάτων μεταφορών(λεωφορεία, Ταξί, ECV). Οπότε δεν λαμβάνονται υπόψιν ιδιαιτερότητες και απαιτήσεις των διαφορετικών διαδρομών και κατηγοριών των EV.

Τα περισσότερα FLM στοχεύουν στην μεγιστοποίηση της κάλυψης της ροής των οχημάτων και αποτελούν προβλήματα μέγιστης κάλυψης. Λόγω συγκεκριμένων οικονομικών κεφαλαίων που διατίθενται σε προγράμματα εγκατάστασης CS, για να ακολουθηθεί ο καθορισμένος προϋπολογισμός, είναι πιο εύκολο να επιλέγονται οι τοποθεσίες για συγκεκριμένο αριθμό p CS ώστε να καλύπτετε το μεγαλύτερο δυνατό ποσοστό της ζήτησης για φόρτιση ($p - median$). Άλλα μοντέλα, προσεγγίζουν το πρόβλημα ελαχιστοποιώντας το κόστος, ενώ εξασφαλίζουν την πλήρη κάλυψη της ζήτησης (set covering problems). Το κόστος μπορεί να αφορά την εγκατάσταση [26] και την λειτουργία των CS [27], την επέκταση του ηλεκτρικού δικτύου [28], επενδύσεις σε μπαταρίες [26] και αποθήκευση ενέργειας [27] καθώς και εκπομπές αερίων του θερμοκηπίου[24]. Τα κόστη μπορεί να εκφράζουν και κάποια χρονική έννοια, όπως ο χρόνος αναμονής [29] ή ο χρόνος έως να φτάσει το EV στον CS [24]. Μια διαφορετική προσέγγιση μπορεί να γίνει εάν αντί για ελαχιστοποίηση του κόστους, εξεταστεί η μεγιστοποίηση του κέρδους που προέρχεται από την χρήση των CS. Τα κέρδη που προέρχονται από την χρήση των σταθμών είναι άμεσα συνδεδεμένα με την βέλτιστη τοποθέτηση των CS. Συνήθως, οι μοντελοποιήσεις με αντικειμενική συνάρτηση την μεγιστοποίηση κέρδους, αφορούν ιδιωτικούς φορείς που επενδύουν στην εγκατάσταση των CS.

2.2.1 Μοντέλο κάλυψης ροής-κίνησης (Flow capturing location model-FCLM)

Το FCLM προτάθηκε από τον M.J. Hodgson [30], λαμβάνει υπόψιν την έννοια του «κανιβαλισμού» της εξυπηρέτησης, πιο αναλυτικά τις περιπτώσεις όπου οι εγκαταστάσεις έχουν τοποθετηθεί τόσο κοντά η μία στην άλλη που η πρώτη καλύπτει μερίδιο της ζήτησης της δεύτερης. Για την συγκεκριμένη μορφοποίηση έχει γίνει η υπόθεση ότι ένας κόμβος μπορεί να καλύψει όλη τη ζήτηση της ροής κίνησης που διέρχεται από την θέση του. Ως μοντέλο δεν λαμβάνει υπόψιν την εμβέλεια των οχημάτων. Η μαθηματική μορφοποίηση του FCLM έχει αρκετές ομοιότητες με τα κλασσικά προβλήματα μέγιστης κάλυψης, με κύρια διαφοροποίηση το σύνολο N_q που περιλαμβάνει τους κόμβους που βρίσκονται πάνω στην διαδρομή q και είναι ικανοί να καλύψουν τη ροή f_q. Το μαθηματικό μοντέλο που περιγράφει το FCLM παρουσιάζεται στην συνέχεια:

$$Max\ Z = \sum_{q \in Q} f_q * y_q \qquad (1)$$

$$s.t. \sum_{k \in N_q} x_k \geq y_q, \qquad \forall q \in Q \qquad (2)$$

$$\sum_{k \in N} x_k = p \qquad (3)$$

$$x_k, y_q \in \{0,1\}, \qquad \forall\ q \in Q, k \in N \qquad (4)$$

Δεδομένα

N	Σύνολο των κόμβων του οδικού δικτύου
Q	Σύνολο των διαδρομών Αφετηρία-Προορισμός των ΕV στο δίκτυο
N_q	Σύνολο κόμβων που ανήκουν στη διαδρομή $q \in Q$
f_q	Ροή ΕV στη διαδρομή q
p	Αριθμός σταθμών φόρτισης που θα εγκατασταθούν

Μεταβλητές απόφασης

x_k $\begin{cases} 1, & \text{εάν στην τοποθεσία } k \text{ τοποθετηθεί σταθμός φόρτισης} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

y_q $\begin{cases} 1, & \text{εάν η ροή κίνησης των EV στο ταξίδι } q \text{ καλύπτεται} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

Η αντικειμενική συνάρτηση (1) μεγιστοποιεί την συνολική ροή EV που μπορεί να καλυφθεί από την εγκατάσταση των CS. Οι περιορισμοί (2) διασφαλίζουν ότι μια διαδρομή q έχει καλυφθεί εάν τουλάχιστον ένας σταθμός έχει τοποθετηθεί κατά το μήκος της. Ο περιορισμός (3) ορίζει τον αριθμό p των σταθμών που θα εγκατασταθούν. Τέλος, οι περιορισμοί (4) δηλώνουν ότι οι μεταβλητές x_k, y_q είναι δυαδικές.

2.2.2 Μοντέλο χωροθέτησης ανεφοδιασμού ροής (Flow refueling location model, FRLM)

Το FRLM είναι μια επέκταση του FCLM με την διαφοροποίηση ότι λαμβάνουν υπόψιν περιορισμούς εμβέλειας των οχημάτων με εναλλακτικά καύσιμα. Συγκεκριμένα για τα EV, η περιορισμένη χωρητικότητα της μπαταρίας και ο ρυθμός κατανάλωσης της ενέργειας περιορίζουν την εμβέλεια τους. Έτσι, ανάλογα με την απόσταση που διανύουν μπορεί να είναι απαραίτητη η επαναφόρτιση τους περισσότερες από μία φορές. Η δυνατότητα για πολλαπλές στάσεις είναι η κύρια διαφοροποίηση των FRLM από τα FCLM. Διαδρομές OD εκπροσωπούν την ζήτηση που πρέπει να καλυφθεί και μια ροή θεωρείται ότι έχει ανεφοδιαστεί εάν επαρκής αριθμός σταθμών έχει κατανεμηθεί καταλλήλως κατά μήκος των διαδρομών. Στο στάδιο της προ επεξεργασίας ορίζεται το σύνολο H το οποίο περιέχει όλους τους δυνατούς συνδυασμούς h των τοποθεσιών που μπορούν να καλύψουν κάθε διαδρομή q. Στην συνέχεια με βάση τα $h \in H$, δημιουργούνται οι παράμετροι b_{qh}, a_{hk}, που εκφράζουν εάν ο συνδυασμός h ανεφοδιάζει την διαδρομή q και εάν η τοποθεσία CS k ανήκει στον συνδυασμό h αντίστοιχα. Η δημιουργία όλων των συνδυασμών h απαιτεί μεγάλη υπολογιστική δύναμη και χρόνο. Το παρακάτω FRLM προτάθηκε από τους [31] και αποτελεί το πρώτο μοντέλο χωροθέτησης με βάση την ροή οχημάτων που λαμβάνει υπόψιν περιορισμούς εμβέλειας των EV.

$$Max\ Z = \sum_{q \in Q} f_q * y_q \tag{5}$$

$$s.t. \sum_{h \in H} b_{qh} * v_h \geq y_q\,, \qquad \forall\ q \in Q \tag{6}$$

$$a_{hk} * x_k \geq v_h, \qquad \forall h \in H, k \in N | a_{hk} = 1 \tag{7}$$

$$\sum_{k \in N} x_k = p \tag{8}$$

$$x_k, y_q, v_h\ \in \{0,1\}, \qquad \forall\ q \in Q, k \in N, h \in H \tag{9}$$

Δεδομένα

N Σύνολο των κόμβων του οδικού δικτύου

Q Σύνολο των διαδρομών Αφετηρία-Προορισμός των ΕV στο δίκτυο

H Σύνολο όλων των συνδυασμών των τοποθεσιών σταθμών φόρτισης, $h \in H$

h Συνδυασμός τοποθεσιών σταθμών φόρτισης που μπορεί να καλύψει μία ή περισσότερες διαδρομές q.

p Αριθμός σταθμών φόρτισης που θα εγκατασταθούν

f_q Ροή ΕV στη διαδρομή q

b_{qh} $\begin{cases} 1, \textit{εαν ο συνδυασμός τοποθεσιών } h \textit{ μπορεί να επαναφορτίσει την διαδρομή } q \\ 0, \quad \textit{σε διαφορετική περίπτωση} \end{cases}$

a_{hk} $\begin{cases} 1, \textit{εαν η τοποθεσία } k \in N \textit{ ανοίκει στον συνδυασμό } h \in H \\ 0, \quad \textit{σε διαφορετική περίπτωση} \end{cases}$

Μεταβλητές απόφασης

x_k $\begin{cases} 1, \quad \textit{εάν στην τοποθεσία } k \textit{ τοποθετηθεί σταθμός φόρτισης} \\ 0, \quad \textit{σε διαφορετική περίπτωση} \end{cases}$

y_q $\begin{cases} 1, \quad \textit{εάν η ροή κίνησης των } EV \textit{ στη διαδρομή } q \textit{ καλύπτεται} \\ 0, \quad \textit{σε διαφορετική περίπτωση} \end{cases}$

v_h $\begin{cases} 1, \textit{εαν έχουν εγκατασταθεί } CS \textit{ σε όλες τις τοποθεσίες του συνδυασμού } h \\ 0, \quad \textit{σε διαφορετική περίπτωση} \end{cases}$

Η αντικειμενική συνάρτηση (5) εκφράζει την μεγιστοποίηση της κάλυψης της ζήτησης. Οι περιορισμοί (6) εκφράζουν ότι μια διαδρομή q έχει καλυφθεί εάν τουλάχιστον ένας συνδυασμός εγκαταστάσεων h, που μπορεί να ικανοποιήσει την διαδρομή q, έχει ανοίξει. Οι περιορισμοί (7) θέτουν την μεταβλητή v_h ίση με μηδέν, εκτός εάν όλοι οι σταθμοί στον συνδυασμό h έχουν ανοίξει. Ο περιορισμός (8) ορίζει ότι το πλήθος των σταθμών που θα ανοίξουν συνολικά είναι p. Τέλος οι περιορισμοί (9) ορίζουν τις δυαδικές μεταβλητές του προβλήματος.

2.2.3 Μοντέλα κάλυψης τόξου (Arc covering model)

Για να ξεπεραστεί η δυσκολία του υπολογισμού όλων των συνδυασμών που απαιτείται από τα FRLM, οι [32] ανέπτυξαν μια καινούρια μορφοποίηση, με βάση την οποία η διαδρομή q θεωρείται ότι έχει καλυφθεί εάν κάθε τόξο που απαρτίζει την διαδρομή q καλύπτεται. Η καινοτομία του μοντέλου είναι το σύνολο K_{ij}^{q}, το οποίο περιέχει τους υποψήφιους κόμβους k που καθιστούν δυνατό να διασχιστεί το τόξο (i,j), εφόσον το όχημα έχει φορτιστεί πλήρως στον κόμβο k. Το επίπεδο της μπαταρίας των οχημάτων εξαρτάται από την χωροθέτηση των σταθμών. Εάν το όχημα ξεκινάει από σημείο όπου υπάρχει σταθμός τότε είναι πλήρως φορτισμένη η μπαταρία του. Εάν, όμως, το σημείο δεν έχει σταθμό τότε συνεχίζει το όχημα την διαδρομή του με το εναπομείναν ποσό της ενέργεια που υπάρχει στην μπαταρία. Η τοποθέτηση των CS πρέπει να είναι κατάλληλη ώστε μια κυκλική διαδρομή να μπορεί να διανύεται επαναλαμβανόμενα.

$$Max\ Z = \sum_{q \in Q} f_q * y_q \tag{10}$$

$$s.t. \sum_{k \in K_{ij}^{q}} x_k \geq y_q, \qquad \forall\ q \in Q, (i,j) \in A_q \tag{11}$$

$$\sum_{k \in N} x_k = p \tag{12}$$

$$x_k, y_q\ \in \{0,1\}, \qquad \forall q \in Q, k \in N \tag{13}$$

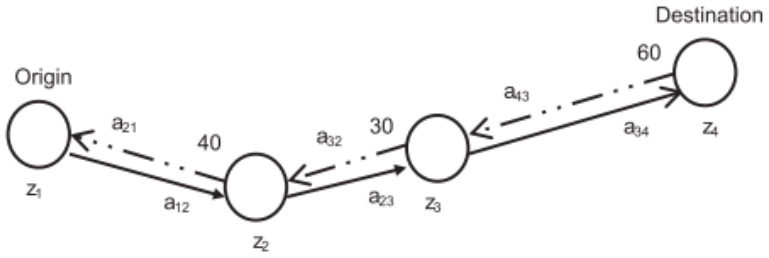
Δεδομένα

N	Σύνολο των κόμβων του οδικού δικτύου
Q	Σύνολο των διαδρομών Αφετηρία-Προορισμός των EV στο δίκτυο
A_q	Σύνολο τόξων κατεύθυνσης (i,j) που ανήκουν στη διαδρομή q
K_{ij}^{q}	Σύνολο υποψήφιων σημείων CS που μπορούν να καλύψουν το τόξο (i,j) διαδρομής q
f_q	Ροή EV στη διαδρομή q
p	Αριθμός σταθμών φόρτισης που θα εγκατασταθούν

Μεταβλητές απόφασης

x_k $\begin{cases} 1, & \varepsilon\acute{\alpha}\nu\ \sigma\tau\eta\nu\ \tau o\pi o\theta\varepsilon\sigma\acute{\iota}\alpha\ k\ \tau o\pi o\theta\varepsilon\tau\eta\theta\varepsilon\acute{\iota}\ \sigma\tau\alpha\theta\mu\acute{o}\varsigma\ \phi\acute{o}\rho\tau\iota\sigma\eta\varsigma \\ 0, & \sigma\varepsilon\ \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\eta}\ \pi\varepsilon\rho\acute{\iota}\pi\tau\omega\sigma\eta \end{cases}$

y_q $\begin{cases} 1, & \varepsilon\acute{\alpha}\nu\ \eta\ \rho o\acute{\eta}\ \kappa\acute{\iota}\nu\eta\sigma\eta\varsigma\ \tau\omega\nu\ EV\ \sigma\tau o\ \tau\alpha\xi\acute{\iota}\delta\iota\ q\ \kappa\alpha\lambda\acute{\upsilon}\pi\tau\varepsilon\tau\alpha\iota \\ 0, & \sigma\varepsilon\ \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\eta}\ \pi\varepsilon\rho\acute{\iota}\pi\tau\omega\sigma\eta \end{cases}$



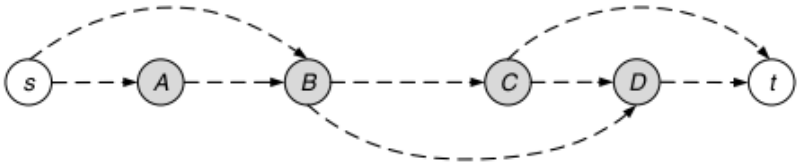
Εικόνα 2 Παράδειγμα OD διαδρομής για όχημα με εμβέλεια 60 μονάδες[32]

Η αντικειμενική συνάρτηση (10) μεγιστοποιεί την συνολική ροή των EV που καλύπτεται. Οι περιορισμοί (11) εξασφαλίζουν ότι μια διαδρομή q καλύπτεται, εάν για κάθε τόξο (i,j) που ανήκει στην q υπάρχει τουλάχιστον ένας σταθμός σε έναν από τους κόμβους του συνόλου $K_{i,j}^{q}$. Ο περιορισμός (12) ορίζει τον αριθμό των σταθμών προς εγκατάσταση και οι περιορισμοί (13) ορίζουν τις μεταβλητές του προβλήματος.

2.2.4 Μοντέλο τμηματοποίησης ή διαχωρισμού σε τμήματα των διαδρομών (Path-segment model)

Τμήμα $[i, j]$ της διαδρομής q ορίζεται ως ένα μέρος της διαδρομής που διανύεται από την αφετηρία προς τον προορισμό του ταξιδιού. Αποτελείται από ένα σύνολο τόξων που διανύονται για να ταξιδέψεις από τον κόμβο i στον κόμβο j και το συνολικό μήκος του τμήματος$[i, j]$ προκύπτει από το αντίστοιχο άθροισμά των μηκών των τόξων που το απαρτίζουν. Για κάθε τμήμα $[i, j]$, το μήκος του θα πρέπει να είναι μικρότερο ή ίσο με την αυτονομία του οχήματος, εκτός εάν το τμήμα ξεκινάει από την αφετηρία ή καταλήγει στον προορισμό της διαδρομής q, όπου σε αυτή την περίπτωση το μήκος θα πρέπει να είναι μικρότερο ή ίσο με το ήμισυ της αυτονομίας του οχήματος. Για κυκλικές διαδρομές που κατά την επιστροφή χρησιμοποιείται η ίδια διαδρομή, δεν χρειάζεται να εξεταστεί ξεχωριστά η διαδρομή της επιστροφής, καθώς εάν το όχημα μπορεί να φτάσει στον πρώτο σταθμό με μισή μπαταρία, τότε μπορεί να φορτίσει στον ίδιο κόμβο στην επιστροφή και να επιστρέψει στην αφετηρία χωρίς να έχει εξαντληθεί η μπαταρία του. Στην περίπτωση όμως, όπου σε κυκλική διαδρομή, η επιστροφή γίνεται μέσω διαφορετικής διαδρομής, τότε η διαδρομή της επιστροφής θα πρέπει να εξεταστεί ξεχωριστά, ως νέα διαδρομή q. Οι διαδρομές q που εξετάζονται αποτελούν τα συντομότερα μονοπάτια που ακολουθούνται από την αφετηρία προς τον τελικό προορισμό.

Κατά την προετοιμασία του μοντέλου δημιουργείται ένα αναπτυγμένο δίκτυο $\widehat{N}_q$. Για τον σχηματισμό αυτού του δικτύου, προστίθενται στο αρχικό σύνολο των κόμβων της διαδρομής q ένας κόμβος «πηγή» s, πριν την αφετηρία της διαδρομής q και ένας κόμβος «καταβόθρα» t, μετά τον κόμβο προορισμού. Στους κόμβους s και t ανατίθενται οι τιμές 1 και -1 αντίστοιχα, ώστε να εκφράσουν την εικονική παροχή και ζήτηση. Στους υπόλοιπους ενδιάμεσους κόμβους ανατίθεται η τιμή 0 . Στην συνέχεια, δημιουργούνται τμήματα με αρχή το $[s, i]$ $\forall\ i\ \in \widehat{N}_q$ για τα οποία ισχύει $d[O_q, i] \leq \frac{R}{2}$ και αντίστοιχα για το t δημιουργούνται τμήματα $[i, t]$ $\forall\ i\ \in \widehat{N}_q$ για τα οποία ισχύει $d[i, D_q] \leq \frac{R}{2}$. Τέλος, προστίθενται και τα τμήματα $[i, j], i, j\ \in \widehat{N}_q$ όπου το i προηγείται του j και η μεταξύ τους απόσταση ικανοποιεί την σχέση $d[i, j] \leq R$. Το παρόν μοντέλο παρουσιάστηκε από τους [33].



Εικόνα 3 Παράδειγμα ανεπτυγμένου δικτύου $\widehat{N}_q$ διαδρομής[33]

$$Max\ Z = \sum_{q \in Q} f_q * (1 - y_{st}^q) \quad (14)$$

$$s.t. \sum_{j|(i,j)\in \hat{A}_q} y_{ij}^q - \sum_{j|(i,j)\in \hat{A}_q} y_{ji}^q = \begin{cases} 1, & i = s \\ -1, & i = t, \\ 0, & i \neq s,t \end{cases} \quad \forall q \in Q, i \in \hat{N}_q \quad (15)$$

$$\sum_{j|(i,j)\in \hat{A}_q} y_{ij}^q \leq x_i, \quad \forall i \in N, q \in Q \quad (16)$$

$$\sum_{k \in N} x_k = p \quad (17)$$

$$y_{ij}^q \geq 0, \quad \forall\ q \in Q, (i,j) \in \hat{A}_q \quad (18)$$

$$x_k \in \{0,1\}, \quad \forall k \in N \quad (19)$$

Δεδομένα

$\hat{A}_q$	Σύνολο των τόξων διαδρομής q του ανεπτυγμένου δικτύου
N	Σύνολο των κόμβων του οδικού δικτύου
$\hat{N}_q$	Σύνολο των κόμβων διαδρομής q του ανεπτυγμένου δικτύου
Q	Σύνολο των διαδρομών Αφετηρία-Προορισμός των EV στο δίκτυο
Q_i	Υποσύνολο του Q που περιέχει τις διαδρομές που διέρχονται από τον κόμβο i
f_q	Ροή EV στη διαδρομή q
p	Αριθμός σταθμών φόρτισης που θα εγκατασταθούν

Μεταβλητές απόφασης

y_{ij}^q	Ροή κίνησης στο τόξο (i, j) που ανήκει στη ανεπτυγμένη διαδρομή q
x_k	$\begin{cases} 1, & εάν\ στην\ τοποθεσία\ k\ τοποθετηθεί\ σταθμός\ φόρτισης \\ 0, & σε\ διαφορετική\ περίπτωση \end{cases}$

Η αντικειμενική συνάρτηση (14) μεγιστοποιεί την ροή που καλύπτεται από τους σταθμούς που ανοίγουν. Οι εξισώσεις (15) είναι οι περιορισμοί ισοζυγίου ροής. Οι περιορισμοί (16) εξασφαλίζουν ότι ένας σταθμός θα εγκατασταθεί σε οποιονδήποτε θέση από την οποία διέρχεται κάποιο τμήμα διαδρομής. Ο περιορισμός (17) περιορίζει το πλήθος των σταθμών που θα ανοίξουν. Τέλος, οι περιορισμοί (18),(19) ορίζουν τις μεταβλητές του προβλήματος.

2.2.5 Μοντέλο παρακολούθησης της κατάστασης της μπαταρίας (Battery state of charge tracking model, BSOC)

Το BSOC μοντέλο εστιάζει στην παρακολούθηση της κατάστασης της μπαταρίας του EV για την βέλτιστη τοποθέτηση των σταθμών. Βασίζεται στην λογική της δρομολόγησης οχημάτων για την κατάστρωση των σχέσεων της μπαταρίας. Για την μορφοποίηση του προβλήματος, χρησιμοποιείται ο πίνακας των αποστάσεων των συντομότερων μονοπατιών για τις OD διαδρομές. Γίνεται, η υπόθεση ότι όλα τα οχήματα είναι του ίδιου τύπου, με σταθερή εμβέλεια και η κατανάλωση εξαρτάται γραμμικά από την απόσταση που διανύεται. Το ίδιο ισχύει και για τον ρυθμό επαναφόρτισης. Επίσης, οι σταθμοί θεωρείται ότι έχουν μη περιορισμένη χωρητικότητα και ότι τα οχήματα ξεκινούν από την αφετηρία τους με πλήρως φορτισμένη μπαταρία. Η αντικειμενική συνάρτηση αφορά την ελαχιστοποίηση του κόστους εγκατάσταση των σταθμών.

Δεδομένων των OD διαδρομών, των κόμβων και των τόξων δημιουργείται ο πίνακας αποστάσεων των συντομότερων μονοπατιών και ο προσδιορισμός των τόξων που διανύει κάθε όχημα. Στο μοντέλο απαιτείται να ικανοποιηθεί η απαίτηση για φόρτιση (set covering), οι σταθμοί έχουν μη περιορισμένη δυναμικότητα (Uncapacitated) και εξετάζεται η χρήση μόνο ενός τύπου CS (fast chargers).

Η εμβέλεια των οχημάτων είναι κρίσιμη μεταβλητή για τον προσδιορισμό του πλήθους των κατάλληλων τοποθεσιών των CS. Επίσης, η βελτιστοποίηση γίνεται με γνώμονα το ελάχιστο κόστος εγκατάστασης, καθώς το κόστος της εγκατάστασης ανά τοποθεσία επηρεάζει την επιλογή των τοποθεσιών. Αυτό σημαίνει ότι διαφορετική λύση θα δοθεί εάν όλες οι τοποθεσίες έχουν ίδιο ή διαφορετικό κόστος.

$$Min\ Z = \sum_{i \in N} c_i * x_i \tag{20}$$

$$s.t. B_{im} \geq 0, \qquad \forall i \in N, m \in M \tag{21}$$

$$B_{im} = B_{jm} + R_{jm} - d(j,i) * \delta_{jim}\ , \qquad \forall (j,i) \in A, m \in M \tag{22}$$

$$R_{im} \leq R - B_{im}, \qquad \forall i \in N, m \in M \tag{23}$$

$$R_{im} = y_{im} * R - A_{im}, \qquad \forall i \in N, m \in M \tag{24}$$

$$\sum_{m \in M} y_{im} \leq L * x_i, \qquad \forall i \in N \tag{25}$$

$$x_i \in \{0,1\}, \qquad i \in N \tag{26}$$

$$y_{im} \in \{0,1\}, \qquad i \in N, m \in M \tag{27}$$

$$A_{im}, B_{im}, R_{im} \geq 0, \qquad \forall i \in N, m \in M \tag{28}$$

Δεδομένα

N	Σύνολο των κόμβων του οδικού δικτύου
M	Σύνολο οχημάτων
A	Σύνολο των τόξων (i,j) του δικτύου, $i \in N, j \in N$
c_i	Κόστος εγκατάστασης CS στον κόμβο i
$d(i,j)$	Μήκος τόξου (i,j) $\in A$
δ_{jim}	$\begin{cases} 1, \text{εάν το τόξο } (i,j) \text{ είναι σε διαδρομή που διανύει το όχημα } m \in M \\ 0, \text{σε άλλη περίπτωση} \end{cases}$
R	Εμβέλεια EV (km ή αντίστοιχη μέγιστη φόρτιση/μπαταρία)
L	Μεγάλος αριθμός

Μεταβλητές απόφασης

x_i	$\begin{cases} 1, & \text{εάν στην τοποθεσία } i \text{ τοποθετηθεί σταθμός φόρτισης} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$
y_{im}	$\begin{cases} 1, & \text{εαν το όχημα } m \text{ φορτίζει στον κόμβο } i \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$
A_{im}	Συντελεστής προσαρμογής επαναφόρτισης του EV m στον κόμβο i
B_{im}	Το εναπομείναν ποσό ενέργειας που έχει η μπαταρία του EV m στον κόμβο i
R_{im}	Ποσό επαναφόρτισης του EV m στον κόμβο i

Η αντικειμενική συνάρτηση (20) ελαχιστοποιεί το συνολικό κόστος εγκατάστασης των σταθμών. Οι περιορισμοί (21) εξασφαλίζει ότι όταν το όχημα m βρίσκεται στο σημείο i , το ποσό της μπαταρίας B_{im} θα είναι μη αρνητικό. Το ποσό της επαναφόρτισης στο προηγούμενο σημείο θα πρέπει να είναι ίσο ή μεγαλύτερο από την ενέργεια που καταναλώθηκε κατά την διαδρομή από το προηγούμενο σημείο j στο i. Οι περιορισμοί (22) ορίζουν το ποσό B_{im} σε σχέση με το εναπομείναν ποσό ενέργειας στο j B_{jm}, το ποσό επαναφόρτισης του EV m στο j R_{jm} και την ενέργεια που καταναλώνεται κατά την διαδρομή από το j προς το i. Οι περιορισμοί (23) ορίζουν ότι το ποσό επαναφόρτισης στο σημείο i θα πρέπει να είναι ίσο ή μικρότερο από το μέγιστο ποσό επαναφόρτισης της μπαταρίας ελαττωμένο κατά το εναπομείναν ποσό ενέργειας που έχει η μπαταρία του EV m στον κόμβο i, B_{im}. Οι περιορισμοί (24) ορίζουν ότι το ποσό επαναφόρτισης του EV m στον κόμβο i, R_{im} θα πρέπει να ισούται με το μέγιστο ποσό επαναφόρτισης της μπαταρίας, εάν το όχημα φορτίζει στον σταθμό i, μειωμένο κατά μια μεταβλητή προσαρμογής A_{im}, που χρησιμοποιείται για να υπολογίζεται το ποσό της επαναφόρτισης. Τέλος, οι περιορισμοί (25) συνδέουν τις μεταβλητές χωροθέτησης με της μεταβλητές επαναφόρτισης, σταθμός θα εγκατασταθεί εφόσον οχήματα φορτίζουν στο εν λόγω σημείο. Οι περιορισμοί (26),(27) και (28) ορίζουν τις μεταβλητές του προβλήματος.

2.3 Προσέγγιση με την ζήτηση να εκφράζεται σε σημεία (Node base models)

Μια διαφορετική προσέγγιση που έχει χρησιμοποιηθεί για το πρόβλημα της βέλτιστης χωροθέτησης σταθμών φόρτισης, προκύπτει όταν η ζήτηση για επαναφόρτιση θεωρείται ότι συγκεντρώνεται σε συγκεκριμένα σημεία (nodes) και δεν εξαρτάται από τις διαδρομές που ακολουθούν τα οχήματα. Συνήθως η προσέγγιση στοχεύει την ελαχιστοποίηση του συνολικού κόστους ενώ ικανοποιείται όλη η ζήτηση του δικτύου.

2.3.1 Βασικό μαθηματικό μοντέλο

Η μορφοποίηση που ακολουθεί αποτελεί ένα παράδειγμα προβλήματος μεικτού ακέραιου προγραμματισμού για την προσέγγιση του προβλήματος της χωροθέτησης σταθμών φόρτισης με την ζήτηση να εκφράζεται συγκεντρωτικά σε σημεία[34]. Το πρόβλημα μορφοποιείται έτσι ώστε να ελαχιστοποιείται ο αριθμός σταθμών φόρτισης που απαιτούνται(29), ενώ ικανοποιείται η ζήτηση σε κάθε σημείο(30).

$$Min \sum_{i \in S_c} x_i \qquad (29)$$

$$\sum_{i \in S_c : C_{i,j}=1} x_i \geq 1, \qquad \forall j \in S_p \qquad (30)$$

$$x_i \in \{0,1\}, \qquad \forall i \in S_c \qquad (31)$$

Δεδομένα

$C_{i,j}$ $\begin{cases} 1, & \text{εάν το σημείο } i \text{ μπορεί να καλύψει το σημείο } j \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

S_c Σύνολο προκαθορισμένων σημείων πιθανών θέσεων σταθμών φόρτισης

S_p Σύνολο σημείων από τα οποία γεννάται η ζήτηση

Μεταβλητές *απόφασης*

x_i $\begin{cases} 1, & \text{εάν στην τοποθεσία } i \text{ τοποθετηθεί σταθμός φόρτισης} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

Κεφάλαιο 3ο

3.1 Δρομολόγηση οχημάτων και χωροθέτηση σταθμών φόρτισης.

Το πρόβλημα της Δρομολόγησης Οχημάτων και Χωροθέτησης σταθμών φόρτισης (Location Routing Problem – LRP) ασχολείται με τον προσδιορισμό τόσο των τοποθεσιών εγκαταστάσεων, όσο και των διαδρομών που θα ακολουθήσουν τα οχήματα τα οποία εξυπηρετούν τους πελάτες. Τα προβλήματα LRP είναι ένα σύνολο προβλημάτων που αφορούν τον σχεδιασμό της χωροθέτησης εγκαταστάσεων, λαμβάνοντας υπόψη την σχεδίαση δρομολογίων οχημάτων [35]. Στην δομή του προβλήματος μπορεί να συμπεριλαμβάνονται περιορισμοί που αφορούν την χωρητικότητα των εγκαταστάσεων ή/και των οχημάτων, χρονικοί περιορισμοί ή περιορισμοί εμβέλειας κοκ. Τα LRP είναι εμφανώς συνδεδεμένα τόσο με το κλασσικό πρόβλημα της δρομολόγησης οχημάτων, όσο και με την χωροθέτηση εγκαταστάσεων. Αποτελεί πρόβλημα NP-Hard, καθώς ενσωματώνει δυο NP-Hard προβλήματα.

Οι θέσεις των σταθμών φόρτισης επηρεάζουν τα δρομολόγια των οχημάτων, όπως και αντίστροφα η χωροθέτηση και ο βαθμός αξιοποίησης των εγκαταστάσεων σταθμών φόρτισης εξαρτώνται από τα δρομολόγια των οχημάτων. Πολύ συχνά δεν επιλέγεται να εξεταστούν ταυτόχρονα η απόφαση της τοποθεσίας μιας νέας εγκατάστασης και η δρομολόγηση οχημάτων που διέρχονται από αυτήν. Αυτό συμβαίνει γιατί πολλοί ακαδημαϊκοί και επαγγελματίες υποστηρίζουν ότι η πρώτη απόφαση είναι στρατηγική ενώ τα δρομολόγια μπορεί να επανεξεταστούν μέσα σε σύντομο χρονικό διάστημα, και έτσι δεν είναι αποφάσεις που μπορούν να ληφθούν στον ίδιο χρονικό ορίζοντα. Αντιθέτως, υπάρχει και η άποψη που υποστηρίζει ότι πολλά πλεονεκτήματα, τόσο στρατηγικά όσο και οικονομικά, μπορούν να προκύψουν από την ταυτόχρονη λήψη αυτών των αποφάσεων. Τα πλεονεκτήματα αυτά θα αναλυθούν στην ενότητα «3.2 Πλεονεκτήματα LRP».

3.2 Πλεονεκτήματα LRP

Η LRP μοντελοποίηση παρουσιάζει αρκετά στρατηγικά και οικονομικά πλεονεκτήματα με αποτέλεσμα να αποτελεί έναν ενδιαφέρον κλάδο έρευνας. Αρχικά, με την δρομολόγηση και χωροθέτηση σταθμών φόρτισης μπορεί να επιτευχθεί σημαντική μείωση στις χιλιομετρικές αποστάσεις που διανύουν τα οχήματα. Καθώς, μπορούν να ελαχιστοποιηθούν παρεκκλίσεις για εύρεση σταθμού φόρτισης, λαμβάνοντας υπόψιν τοποθεσίες που λειτουργούν συμπληρωματικά στην δρομολόγηση των οχημάτων, όπως περιοχές διαλλειμάτων των οδηγών και τοποθεσίες πελατών θέσεις CS οι οποίες συμπεριλαμβάνονται στην δρομολόγηση των οχημάτων. Ακόμα, λόγω χρήσης ιδιωτικών τοποθεσιών για την εγκατάσταση σταθμών φόρτισης, αποφεύγεται η περίπτωση να μην υπάρχει διαθέσιμος σταθμός εξαιτίας της χρήσης του από τον γενικό πληθυσμό, έτσι δεν σπαταλάτε χρόνος για αναμονή. Ταυτόχρονα, ελέγχεται η κατάσταση του σταθμού ώστε να μην υπάρχει φθορά από εξωτερικούς παράγοντες.

Στην συνέχεια, αξίζει να γίνει αναφορά στα οικονομικά πλεονεκτήματα που προκύπτουν από την χρήση του LRP μοντέλου. Το καθημερινό πρόγραμμα λειτουργίας δεν θα επηρεάζεται από την διαθεσιμότητα δημόσιων σταθμών φόρτισης. Επίσης, έχει αναφερθεί πως υπάρχουν οικονομικά οφέλη ακόμα και σε περιπτώσεις όπου οι διαδρομές που έχουν χαραχθεί προηγουμένως αλλάξουν στο μέλλον [35].Ακόμα, το κόστος φόρτισης δεν θα εξαρτάται από το κόστος σε δημόσιους σταθμούς, το οποίο συνήθως μεταβάλλεται ανάλογα την χρονική περίοδο μέσα στην ημέρα που θα λαμβάνει χώρα η φόρτιση, όσο και από την τιμή της ηλεκτρικής ενέργειας. Τέλος, οικονομικό όφελος μπορεί να προκύπτει από τις τεχνολογίες Vehicle to Grid (V2G), όπου τα ECV μπορούν να χρησιμοποιηθούν σε περιπτώσεις όπως όταν η τάση του δικτύου παρουσιάζει μεγάλες διακυμάνσεις και οι τιμές των βοηθητικών οι υπηρεσίες είναι υψηλές.

3.3 Μαθηματικό μοντέλο

Το μαθηματικό μοντέλο που θα παρουσιαστεί στην συνέχεια είναι εμπνευσμένο από τους [36], αποτελεί το πρώτο στο οποίο επιτρέπεται μερική φόρτιση, φόρτιση σε πιθανές τοποθεσίες φόρτισης και σε θέσεις πελατών. Χρησιμοποιείται η έννοια των εικονικών σημείων για να μπορούν να χρησιμοποιηθούν πολλαπλές φορές οι σταθμοί φόρτισης. Συγκεκριμένα, για κάθε πιθανή θέση σταθμού φόρτισης, δημιουργούνται τόσοι επιπλέων εικονικοί σταθμοί όσοι και οι πελάτες.

C	Σύνολο των κόμβων των πελατών
R	Σύνολο των πιθανών κόμβων θέσεων σταθμών φόρτισης
S_k	Σύνολο των εικονικών σταθμών του κόμβου $k \in (C \cup R)$
S	Σύνολο όλων των εικονικών κόμβων $(\cup_{k\in(C\cup R)} S_k)$
V	Σύνολο όλων των κόμβων, εκτός της αφετηρίας $(C \cup R \cup S)$
V_0	Σύνολο όλων των κόμβων και της αφετηρίας $(C \cup R \cup S \cup \{0\})$
V_{n+1}	Σύνολο όλων των κόμβων και του κόμβου της αφετηρίας που αντιστοιχεί για την επιστροφή των οχημάτων $(C \cup R \cup S \cup \{n+1\})$
C_0	Σύνολο των κόμβων των πελατών και του κόμβου της αφετηρίας $(C \cup \{0\})$
$V\backslash C$	Σύνολο όλων των κόμβων, εκτός των κόμβων των πελατών
$V\backslash S$	Σύνολο όλων των κόμβων, εκτός των εικονικών κόμβων
e_i	Πρώτη χρονική στιγμή που είναι διαθέσιμος ο κόμβος i
l_i	Τελευταία χρονική στιγμή που είναι διαθέσιμος ο κόμβος i
s_i	Χρόνος εξυπηρέτησης του κόμβου i
p_i	Ζήτηση στον κόμβο i
t_{ij}	Χρονική διάρκεια μετάβασης από τον κόμβο i στον κόμβο j
d_{ij}	Απόσταση ανάμεσα στον κόμβο i και στον κόμβο j
r	Ρυθμός επαναφόρτισης
o	Ρυθμός κατανάλωσης ενέργειας
Q	Χωρητικότητα μπαταρίας
F	Χωρητικότητα φορτίου οχήματος
O^{UB}	Μέγιστος αριθμός φορτιστών
N^{UB}	Μέγιστος αριθμός διαθέσιμων οχημάτων
x_{ij}	$\begin{cases} 1, & \text{εάν το τόξο } (i,j) \text{ διασχίζεται} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$
y_i	$\begin{cases} 1, & \text{εάν τοποθετείται σταθμός στον κόμβο } i \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$
τ_i	Χρόνος άφιξης στον κόμβο i
q_i	Ποσό ενέργειας στη μπαταρία στον κόμβο i
w_i	Ποσό ενέργεια που ανακτάται στον κόμβο i
f_i	Ποσότητα φορτίου κατά την άφιξη στον κόμβο i

$$Minimize \sum_{i\in V_0}\sum_{j\in V_{n+1}} D_{ij} * x_{ij} \quad (32)$$

$$\sum_{j\in V_{n+1}, i\neq j} x_{ij} = 1, \qquad \forall\, i \in C \quad (33)$$

$$\sum_{j\in V_{n+1}, i\neq j} x_{ij} \leq 1, \qquad \forall\, i \in \{V\backslash C\} \quad (34)$$

$$\sum_{i\in V_{n+1}, i\neq j} x_{ji} - \sum_{i\in V_0, i\neq j} x_{ij} = 0, \qquad \forall\, j \in V \quad (35)$$

$$\tau_j \geq \tau_i + \left(t_{ij} + s_i\right) * x_{ij} - l_0 * \left(1 - x_{ij}\right), \quad \forall\, i \in C_0, \forall\, j \in V_{n+1}, i \neq j \quad (36)$$

$$\tau_j \geq \tau_i + t_i * x_{ij} + r * w_i - (l_0 + r * Q) * \left(1 - x_{ij}\right), \qquad \forall i \in V, \quad \forall j \in V_{n+1}, i \neq j \quad (37)$$

$$e_i \leq \tau_i \leq l_i, \qquad \forall i \in V_{0,n+1} \quad (38)$$

$$w_i \leq Q * y_i, \qquad \forall i \in V \quad (39)$$

$$y_i \leq w_i, \qquad \forall i \in S \quad (40)$$

$$O^{LB} \leq \sum_{i\in\{V\backslash S\}} y_i \leq O^{UB} \quad (41)$$

$$y_i \geq y_j, \qquad \forall i \in \{V\backslash S\}, \forall j \in S_i \quad (42)$$

$$0 \leq f_j \leq f_i - p_i * x_{ij} + F * \left(1 - x_{ij}\right), \qquad \forall i \in V_0, \forall j \in V_{n+1}, i \neq j \quad (43)$$

$$0 \leq f_0 \leq F \quad (44)$$

$$q_j \leq q_0 - o * D_{0j} * x_{ij} + Q * \left(1 - x_{ij}\right), \qquad \forall j \in V_{n+1} \quad (45)$$

$$0 \leq q_j \leq q_i + w_i - o * D_{ij} * x_{ij} + Q * \left(1 - x_{ij}\right), \quad \forall i \in V, \forall j \in V_{n+1}, \; i \neq j \quad (46)$$

$$0 \leq q_0 \leq Q \quad (47)$$

$$0 \leq q_i + w_i \leq Q \quad (48)$$

$$N^{LB} \leq \sum_{j\in V_{n+1}} x_{0j} \leq N^{UB} \quad (49)$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in V_0, \forall j \in V_{n+1} \quad (50)$$

$$y_i \in \{0,1\}, \qquad \forall i \in V \quad (51)$$

Η αντικειμενική συνάρτηση (76) στοχεύει στην εύρεση της ελάχιστης διαδρομής. Οι περιορισμοί (33) εκφράζει ότι κάθε πελάτης θα πρέπει να εξυπηρετηθεί ακριβώς μια φορά. Αντίστοιχα, οι περιορισμοί (34) αφορούν τους κόμβους των σταθμών φόρτισης, εικονικούς και πραγματικούς, αλλά είναι πιο χαλαροί. Οι περιορισμοί (35) εξασφαλίζουν την διατήρηση της ροής της διαδρομής των οχημάτων. Οι εξισώσεις (36) εκφράζουν τους περιορισμούς για τον χρόνο άφιξης για τους κόμβους των πελατών, ενώ για τους κόμβους των σταθμών φόρτισης χρησιμοποιούνται οι περιορισμοί (37). Τα χρονικά παράθυρα ορίζονται μέσω των περιορισμών (38).

Οι περιορισμοί για την τοποθέτηση των σταθμών φόρτισης περιγράφονται από τις εξισώσεις (39) έως και (42). Αρχικά, ορίζεται ότι ένα σταθμός πρέπει να τοποθετηθεί σε έναν κόμβο εφόσον φόρτιση λαμβάνει τόπο σε αυτόν τον κόμβο (39) και αντίστροφα, σταθμός φόρτισης δεν μπορεί να τοποθετηθεί εάν δεν υπάρχει φόρτιση (40). Οι σταθμοί φόρτισης που θα τοποθετηθούν θα πρέπει να είναι αρκετοί για να καλύψουν τις ανάγκες της δρομολόγησης και να μην ξεπερνάνε το άνω όριο που έχει οριστεί (41). Η τιμή του O^{LB}θα υπολογιστεί επιλύοντας το πρόβλημα που παρουσιάζεται στην παράγραφο «3.4.3 Εύρεση ελάχιστου αριθμού σταθμών φόρτισης». Τέλος, οι αποφάσεις για τους εικονικούς σταθμούς φόρτισης θα πρέπει να αντικατοπτρίζονται στους πραγματικούς αντίστοιχους κόμβους (42).

Οι επόμενοι περιορισμοί αφορούν το φορτίο του οχήματος. Το ποσό του διαθέσιμου φορτίου στον κόμβο j θα πρέπει να είναι μικρότερο ή ίσο από το ποσό του φορτίου στον κόμβο i, μειωμένο κατά το ποσό της ζήτησης του κόμβου i εφόσον το τόξο (i, j) διασχίζεται (43). Το φορτίο στην αφετηρία ορίζεται από τον περιορισμό (44). Η απαίτηση να μην υπάρχουν υπό-διαδρομές στα δρομολόγια των οχημάτων, εξασφαλίζεται από τους περιορισμούς (43),(44).

Οι περιορισμοί (45) έως και (48) αφορούν τον προσδιορισμό της ενεργειακής κατάστασης των οχημάτων. Οι περιορισμοί (45) περιορίζουν το ποσό της ενέργειας για τα τόξα που ξεκινούν από την αφετηρία. Το επίπεδο της ενέργειας στον κόμβο j συνδέεται με το αντίστοιχο επίπεδο στον κόμβο i και την πιθανή φόρτιση στον κόμβο i μέσω των περιορισμών (46). Στο ποσό της ενέργειας στον κόμβο i προστίθεται το ποσό της πιθανής επαναφόρτισης και αφαιρείται το ποσό της ενέργειας που αντιστοιχεί στο τόξο (i, j). Οι περιορισμοί (48) διασφαλίζουν ότι το άθροισμα του ποσού ενέργειας με το οποίο φτάνει το όχημα στον κόμβο και του ποσού της ενέργειας που θα ανακτήσει από τον σταθμό φόρτισης είναι μικρότερο ή ίσο της χωρητικότητας της μπαταρίας του οχήματος.

Τέλος, περιορίζεται ο αριθμός των οχημάτων προς δρομολόγηση (49), με το κάτω όριο να προσδιορίζεται από την επίλυση του προβλήματος που περιγράφεται στην παράγραφο «3.4.2 Εύρεση ελάχιστου αριθμού οχημάτων». Οι περιορισμοί (50), (51) ορίζουν τις δυαδικές μεταβλητές του προβλήματος.

3.4 Προ επεξεργασία δεδομένων για την μείωση του μεγέθους του προβλήματος

Το πρόβλημα της ταυτόχρονης δρομολόγησης οχημάτων και χωροθέτησης σταθμών φόρτισης ανήκει στην κατηγορία των NP Hard προβλημάτων όπως έχει αναφερθεί προηγουμένως, καθώς όσο αυξάνεται το πλήθος των δεδομένων του, τόσο αυξάνεται και η δυσκολία για την επίλυση του. Η υπολογιστική δυσκολία του προβλήματος αυξάνεται ιδιαίτερα λόγω της χρήση των εικονικών σημείων. Έτσι, εκτελώντας τα παρακάτω βήματα περιορίζεται, έως ένα βαθμό, η υπολογιστική πολυπλοκότητα του προβλήματος.

3.4.1 Εφικτά και Ανέφικτα τόξα

Από τα πιο σημαντικά προβλήματα που παρουσιάζονται κατά την επίλυση των προβλημάτων της κατηγορίας NP - hard είναι ο μεγάλος αριθμός μεταβλητών και περιορισμών. Για να μειωθεί το πλήθος των τόξων που χρησιμοποιούνται κατά την δημιουργία των περιορισμών είναι σημαντικό να βρεθούν τα τόξα που είναι ανέφικτο να χρησιμοποιηθούν με βάση τους λειτουργικούς περιορισμούς που αφορούν τους πελάτες και τα οχήματα. Τα ανέφικτα τόξα εντοπίζονται με βάση τα χρονικά παράθυρα και την ζήτηση των πελατών χρησιμοποιώντας τις παρακάτω σχέσεις.

$$i \in V_0, j \in V_{n+1} \quad \wedge \quad e_i + s_i + t_{ij} > l_j \tag{52}$$

$$i \in V_0, j \in V \quad \wedge \quad e_i + s_i + t_{ij} + s_j + t_{j,n+1} > l_{n+1} \tag{53}$$

$$(i,j) \in C \quad \wedge \quad p_i + p_j > F \tag{54}$$

Περαιτέρω ανέφικτα τόξα προκύπτουν από την χωρητικότητα της μπαταρίας των οχημάτων και την απόσταση μεταξύ των κόμβων.

$$(i,j) \in V_{0,n+1} \quad \wedge \quad O * d_{ij} > Q \tag{55}$$

Τέλος αφαιρούνται τα τόξα που δημιουργούνται ανάμεσα στους πραγματικούς κόμβους και στους αντίστοιχους εικονικούς τους κόμβους.

$$i \in \{C \cup R\} \quad \wedge \quad j \in S_i \tag{56}$$

3.4.2 Εύρεση ελάχιστου αριθμού οχημάτων

Για την εύρεση ενός κατώτερου ορίου για το πλήθος των οχημάτων που είναι απαραίτητο να χρησιμοποιηθούν, επιλύεται το παρακάτω πρόβλημα τύπου bin-packing αξιοποιώντας την ζήτηση των πελατών p_i και την χωρητικότητα F των οχημάτων.

Μεταβλητές απόφασης

z_i $\begin{cases} 1, & \text{εάν το όχημα } i \text{ θα χρησιμοποιηθεί} \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

h_{ij} $\begin{cases} 1, & \text{εάν το όχημα } i \text{ θα εξυπηρετήσει τον πελάτη } j \\ 0, & \text{σε διαφορετική περίπτωση} \end{cases}$

Δεδομένα

Z Πλήθος πιθανών οχημάτων $Z = \{1, \dots, n\},\ n = |C|$

$$Minimize\ N^{LB} = \sum_{i \in Z} z_i \tag{57}$$

$$s.t. \sum_{j \in C} h_{ij} * p_j \leq F * z_i, \qquad \forall i \in Z \tag{58}$$

$$\sum_{i \in Z} h_{ij} = 1, \qquad \forall j \in C \tag{59}$$

$$z_i \in \{0,1\}, \qquad \forall i \in Z \tag{60}$$

$$h_{ij} \in \{0,1\}, \qquad \forall i \in Z, j \in C \tag{61}$$

Η αντικειμενική συνάρτηση (57) δίνει τον ελάχιστο αριθμό οχημάτων που χρειάζονται για την δρομολόγηση των οχημάτων. Η εξίσωση (58) εκφράζει τον περιορισμό για την συνολική χωρητικότητα του κάθε οχήματος που θα χρησιμοποιηθεί, δηλαδή ότι το σύνολο των ζητήσεων των κόμβων που εξυπηρετούνται από το όχημα i θα πρέπει να είναι μικρότερο ή ίσο από την χωρητικότητα φορτίου του οχήματος. Η εξίσωση (59) εκφράζει την απαίτηση να εξυπηρετηθεί ο κάθε κόμβος πελάτη ακριβώς μια φορά. Οι εξισώσεις (60), (61) δηλώνουν ότι οι μεταβλητές είναι δυαδικές.

3.4.3 Εύρεση ελάχιστου αριθμού σταθμών φόρτισης

Για να γίνει η εκτίμηση του κατώτατου ορίου για το πλήθος των σταθμών φόρτισης που είναι απαραίτητοι για την δρομολόγηση, θα επιλυθεί το παρακάτω πρόβλημα, αφού πρώτα προσδιοριστούν οι κόμβοι πελατών που μπορούν να εξυπηρετηθούν μόνο εάν το όχημα επαναφορτιστεί κατά την διαδρομή. Ο προσδιορισμός γίνεται με βάση την απόσταση των κόμβων από την αφετηρία σύμφωνα με την εξίσωση (62)

$$j \in C \quad \wedge \quad O * 2 * d_{0j} > Q \tag{62}$$

Οι κόμβοι που προκύπτουν από τον περιορισμό (62) συνθέτουν το σύνολο J που χρησιμοποιείται για την επίλυση του προβλήματος. Το παρακάτω πρόβλημα δεν λαμβάνει υπόψιν τους περιορισμούς για τα χρονικά παράθυρα, οπότε στην συνέχεια να είναι απαραίτητος μεγαλύτερος αριθμός σταθμών φόρτισης.

Μεταβλητές απόφασης

a_i $\begin{cases} 1, & εαν\ σταθμός\ φόρτισης\ τοποθετείται\ στον\ κόμβο\ i \\ 0, & σε\ διαφορετική\ περίπτωση \end{cases}$

b_{ij} $\begin{cases} 1, & εαν\ ο\ κόμβος\ j\ καλύπτεται\ απο\ τον\ σταθμό\ στον\ κόμβο\ i \\ 0, & σε\ διαφορετική\ περίπτωση \end{cases}$

k_{ij} $\begin{cases} 1, & εαν\ το\ τόξο\ (i,j)\ συνδέει\ δύο\ σταθμούς\ φόρτισης \\ 0, & σε\ διαφορετική\ περίπτωση \end{cases}$

Δεδομένα

J Σύνολό των κόμβων των πελατών που μπορούν να εξυπηρετηθούν μόνο εάν το όχημα επαναφορτιστεί κατά την διαδρομή

$$O^{LB} = minimize \sum_{i \in \{C \cup R\}} a_i \tag{63}$$

$$s.t. \sum_{i \in \{C_0 \cup R\}} b_{ij} \geq 1, \quad \forall j \in J \tag{64}$$

$$b_{ij} \leq a_i, \quad \forall i \in \{C_0 \cup R\}, \forall j \in J \tag{65}$$

$$b_{ij} * O * d_{ij} \leq Q * a_i, \quad \forall i \in \{C_0 \cup R\}, \forall j \in J \tag{66}$$

$$k_{ij} * O * d_{ij} \leq Q, \quad i \in \{C_0 \cup R\}, \forall j \in J \tag{67}$$

$$2 * k_{ij} \leq a_i + a_j, \quad \forall i, j \in \{C_0 \cup R\} \tag{68}$$

$$a_0 = 1 \tag{69}$$

$$a_i \leq \sum_{j \in \{C_0 \cup R\}} k_{ij}, \quad \forall i \in \{C_0 \cup R\} \tag{70}$$

$$a_i, b_{ij}, k_{ij} \in \{0,1\}, \quad \forall i, j \in \{C_0 \cup R\} \tag{71}$$

Η αντικειμενική συνάρτηση (63) αφορά την ελαχιστοποίηση του πλήθους των σταθμών φόρτισης για την κάλυψη των πελατών που ανήκουν στο σύνολο J και απαιτείται επαναφόρτιση ενδιάμεσα στην δρομολόγηση για την εξυπηρέτηση τους. Οι περιορισμοί (64) δηλώνουν κάθε πελάτης θα πρέπει να καλύπτεται από τουλάχιστον ένα σταθμό φόρτισης. Οι περιορισμοί (65) ορίζουν ότι ένας σταθμός τοποθετείται, εφόσον καλύπτει κάποιο πελάτη. Οι περιορισμοί (66) εκφράζουν την ικανότητα να καλυφθεί ένας πελάτης από ένα σταθμό εφόσον η μεταξύ τους απόσταση είναι μικρότερη από την αυτονομία του οχήματος. Δυο κόμβοι στους οποίους έχουν τοποθετηθεί σταθμοί φόρτισης μπορούν να συνδέονται εφόσον η ενέργεια που καταναλώνεται κατά το τόξο που τους ενώνει δεν ξεπερνάει την αυτονομία του οχήματος (67). Οι περιορισμοί (68) αναγκάζουν το k_{ij} να πάρει την τιμή 0 εάν κάποιος από τους κόμβους i, j δεν είναι κόμβος με σταθμό φόρτισης. Ο κόμβος της αφετηρίας αποτελεί και σταθμό φόρτισης (69). Τέλος, εξασφαλίζεται ότι κάθε πελάτης μπορεί να εξυπηρετηθεί λαμβάνοντας υπόψιν τους περιορισμούς της μπαταρίας Q του οχήματος (70). Η σχέση (71) ορίζει τις δυαδικές μεταβλητές του προβλήματος.

Κεφάλαιο 4ο

4.1 Περιγραφή προβλήματος

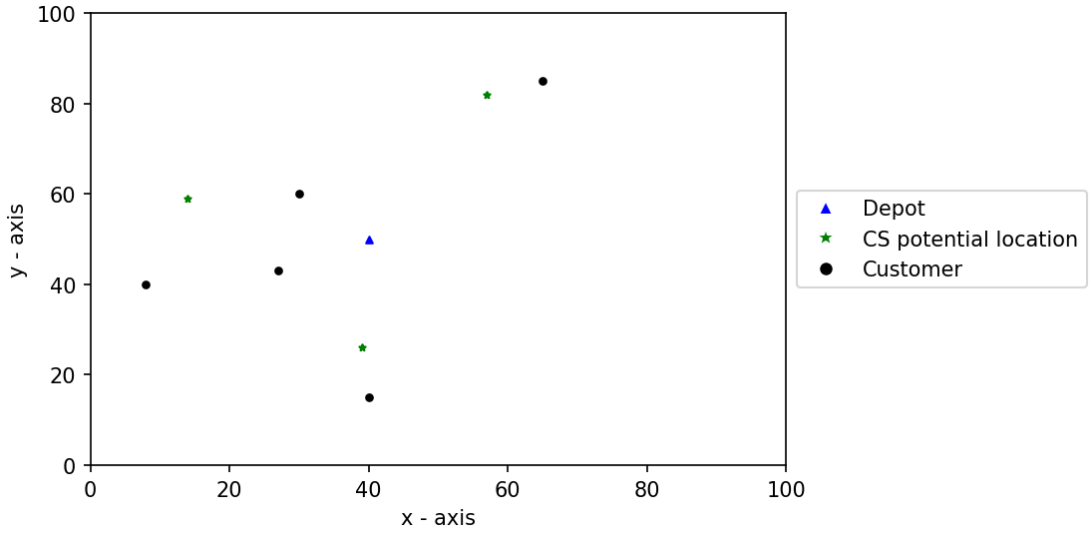
Στη συγκεκριμένη διπλωματική εργασία τα επόμενα Κεφάλαια έχουν αφιερωθεί στη παρουσίαση των συνόλων των δεδομένων και των αποτελεσμάτων. Στην συνέχεια γίνονται σχόλια και παρατηρήσεις πάνω στα αποτελέσματα των δύο μεθόδων επίλυσης του προβλήματος της CSLP για ECV. Η πρώτη μέθοδος που επιλέχθηκε είναι η Κεφάλαιο 3ο
3.1 Δρομολόγηση οχημάτων και χωροθέτηση σταθμών φόρτισης., ενώ η δεύτερη είναι το 2.2.1 Μοντέλο κάλυψης ροής-κίνησης (Flow capturing location model-FCLM). Τα αντίστοιχα μαθηματικά μοντέλα έχουν αναλυθεί στα ομώνυμα Κεφάλαια.

Χρησιμοποιήθηκαν συνολικά επτά (7) διαφορετικά σύνολα δεδομένων των 5, 10, 15 και 100 σημείων πελατών, των αντίστοιχων επιπρόσθετων πιθανών σημείων σταθμών φόρτισης και της αφετηρίας των οχημάτων. Τα εν λόγω δεδομένα αντλήθηκαν από τους [37].

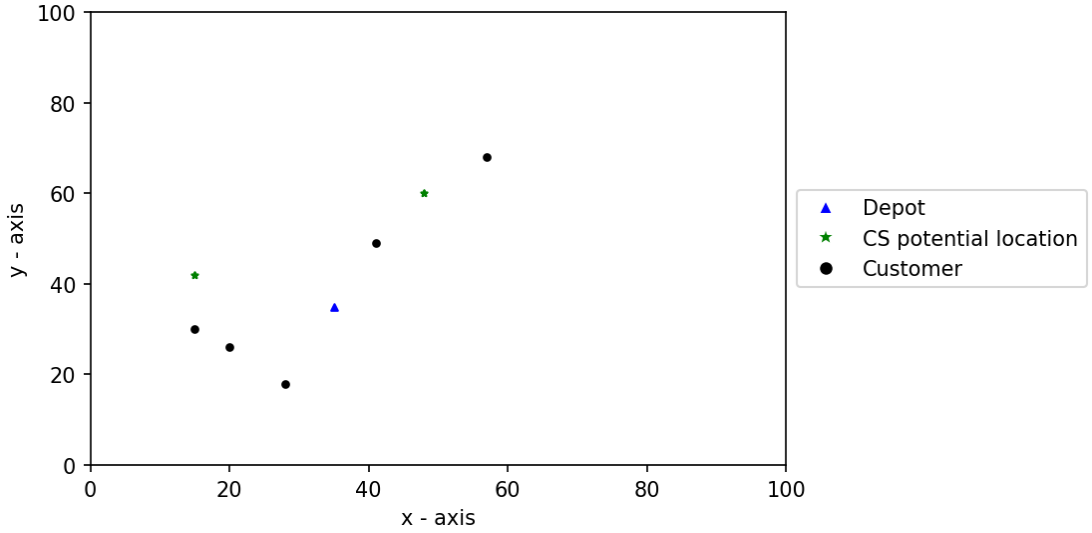
Η επίλυση των μοντέλων πραγματοποιήθηκε με την χρήση του λογισμικού CPLEXILOGIBM στο περιβάλλον της Python. Ο υπολογιστής που χρησιμοποιήθηκε είχε 16 Gb RAM, επεξεργαστή AMD R7 4800H με λογισμικό Windows 11. Το χρονικό όριο που τέθηκε για την επίλυση των συνόλων δεδομένων είναι τα 7200 δευτερόλεπτα. Ο κώδικας που αναπτύχθηκε κατά την συγγραφή της διπλωματικής μπορεί να βρεθεί στο Παράρτημα.

4.2 Παρουσίαση δεδομένων

4.2.1 Σύνολα 5 πελατών

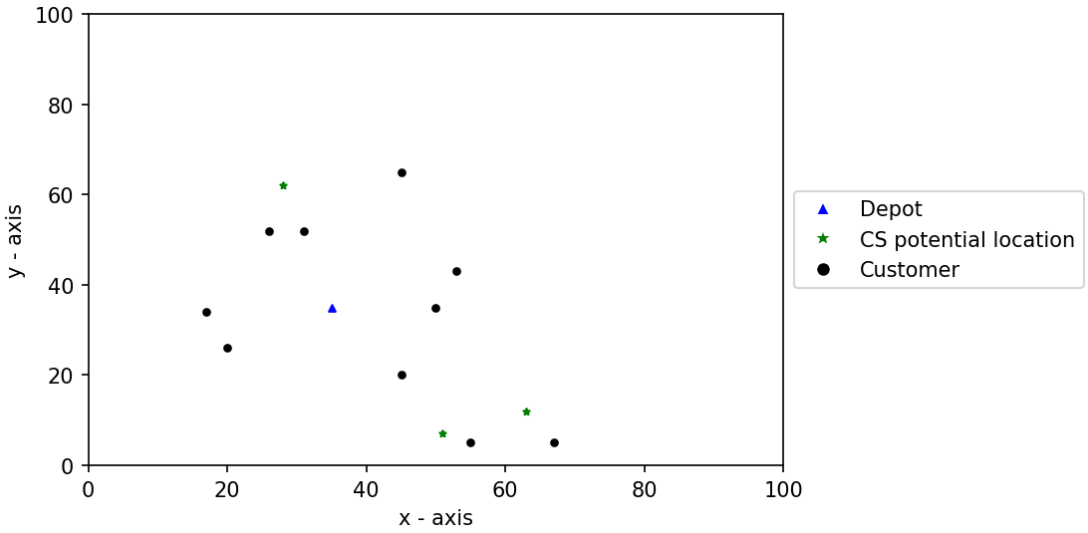


Εικόνα 4 Γραφική αναπαράσταση συνόλου rc105C5

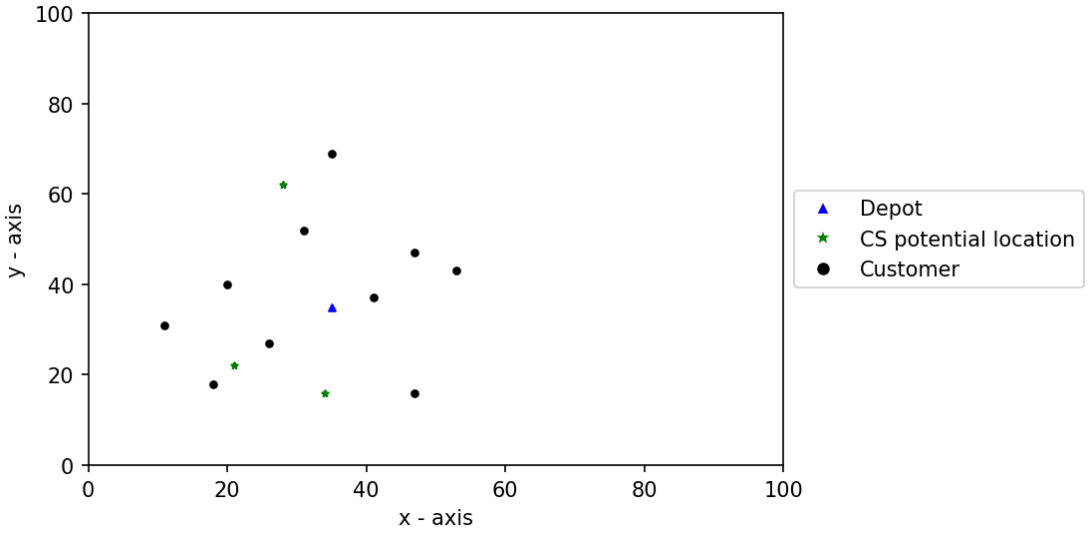


Εικόνα 5 Γραφική αναπαράσταση συνόλου r104C5

4.2.2 Σύνολα10 πελατών

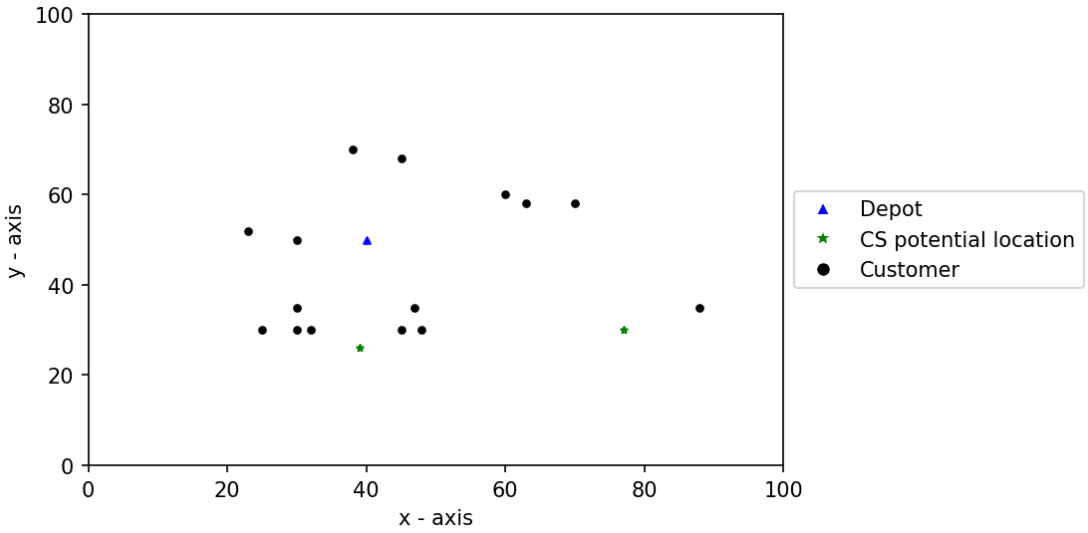


Εικόνα 6 Γραφική αναπαράσταση συνόλου r102C10

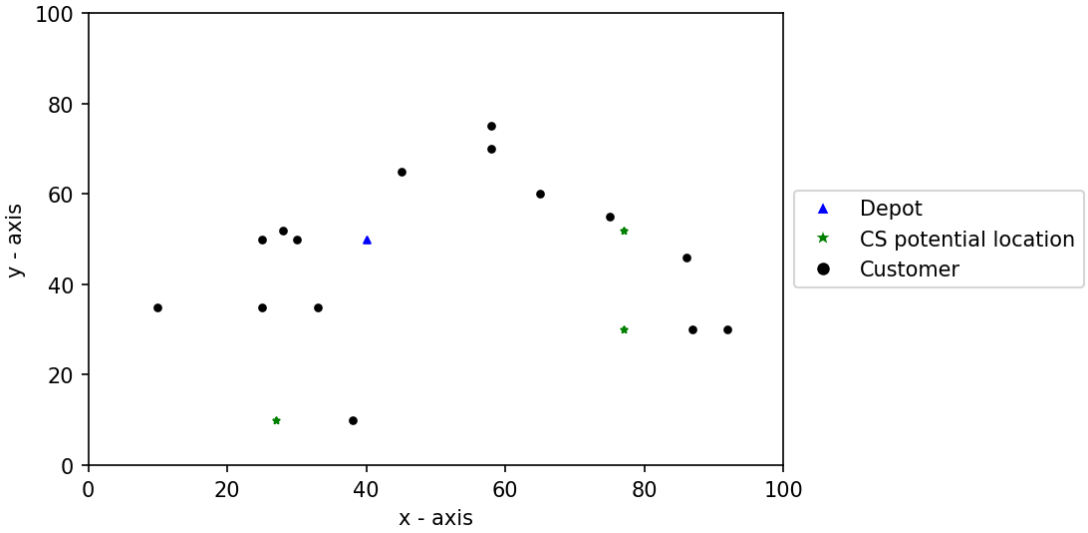


Εικόνα 7 Γραφική αναπαράσταση συνόλου r201C10

4.2.3 Σύνολα 15 πελατών

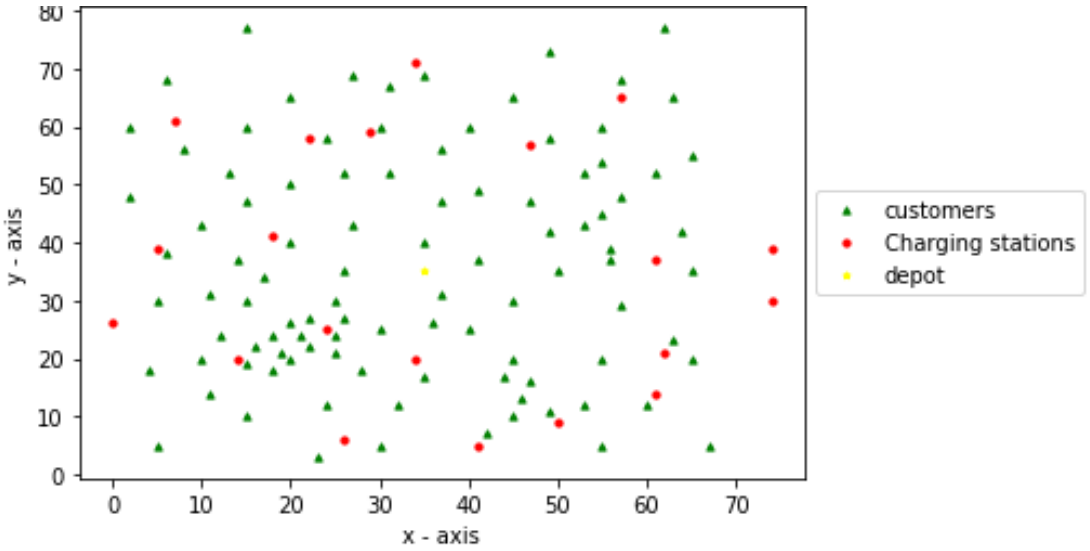


Εικόνα 8 Γραφική αναπαράσταση συνόλου c106C15



Εικόνα 9 Γραφική αναπαράσταση συνόλου c208C15

4.2.4 Σύνολο 100 πελατών



Εικόνα 10 Γραφική αναπαράσταση συνόλου r201C100

4.2.4.1 Ομαδοποίηση 100 σημείων

Κατά την επίλυση του προβλήματος των 100 σημείων, για να βρεθεί λύση σε πεπερασμένο χρονικό ορίζοντα, ήταν απαραίτητη η ομαδοποίηση τους σε μικρότερες υπό-ομάδες καθώς το χρονικό όριο των 7200 δευτερολέπτων που έχει οριστεί δεν ήταν αρκετό για να βρεθεί λύση. Χρησιμοποιήθηκε το παρακάτω τροποποιημένο μαθηματικό μοντέλο που προτάθηκε από τους [38] για την εύρεση των υποομάδων.

$$Minimize \sum_{k\in K}\sum_{i\in n} d_{ik} * Y_{ik} \quad (72)$$

$$s.t. \sum_{i\in n} D_i * Y_{ik} \leq F, \qquad \forall\, k \in K \quad (73)$$

$$\sum_{k\in K} Y_{ik} = \begin{cases} K, & i = 0 \\ 1, & i \in n \end{cases} \quad (74)$$

$$\sum_{i\in n} Y_{ik} \leq NUMC, \qquad \forall k \in K \quad (75)$$

$$Y_{ik} \in \{0,1\}, \qquad \forall i \in n, k \in K \quad (76)$$

Δεδομένα

d_{ik}	Κόστος προσθήκης του σημείου i στην διαδρομή από την αφετηρία στο σημείο k και μετά επιστροφή
D_i	Ζήτηση στον κόμβο i
F	Χωρητικότητα φορτίου οχήματος
K	Πλήθος υποομάδων που θα δημιουργηθούν
n	Πλήθος πελατών που είναι να εξυπηρετηθούν
$NUMC$	Μέγιστος αριθμός πελατών που θα ανατεθούν στην υποομάδα

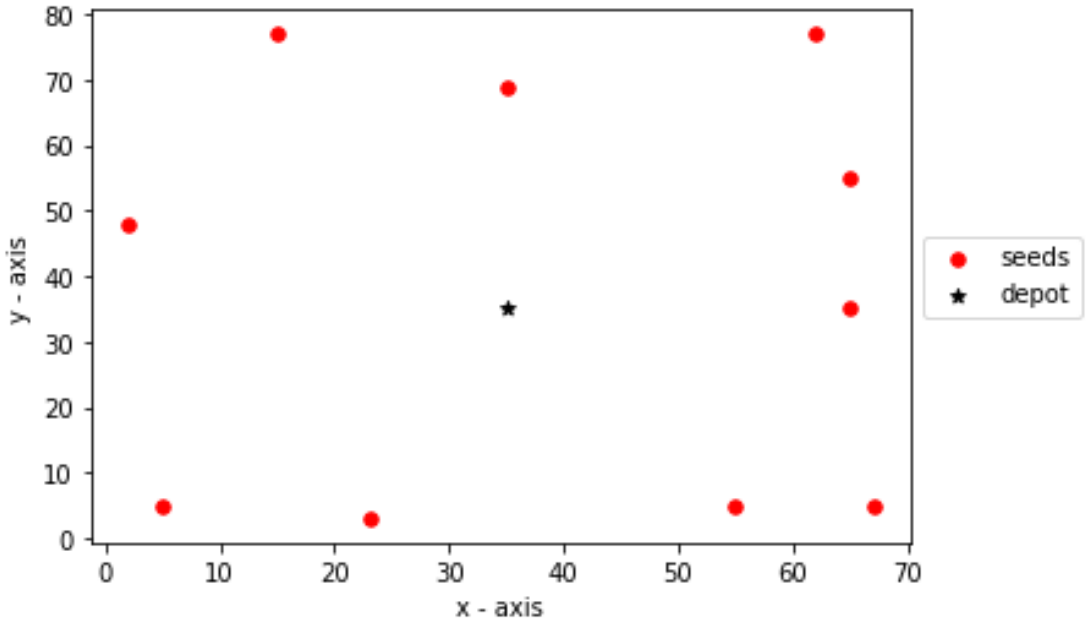
Μεταβλητές απόφασης

$$Y_{ik} \quad \begin{cases} 1, & εαν\ ο\ κομβος\ i\ ανατίθεται\ στην\ ομάδα\ k \\ 0, & σε\ διαφορετική\ περίπτωση \end{cases}$$

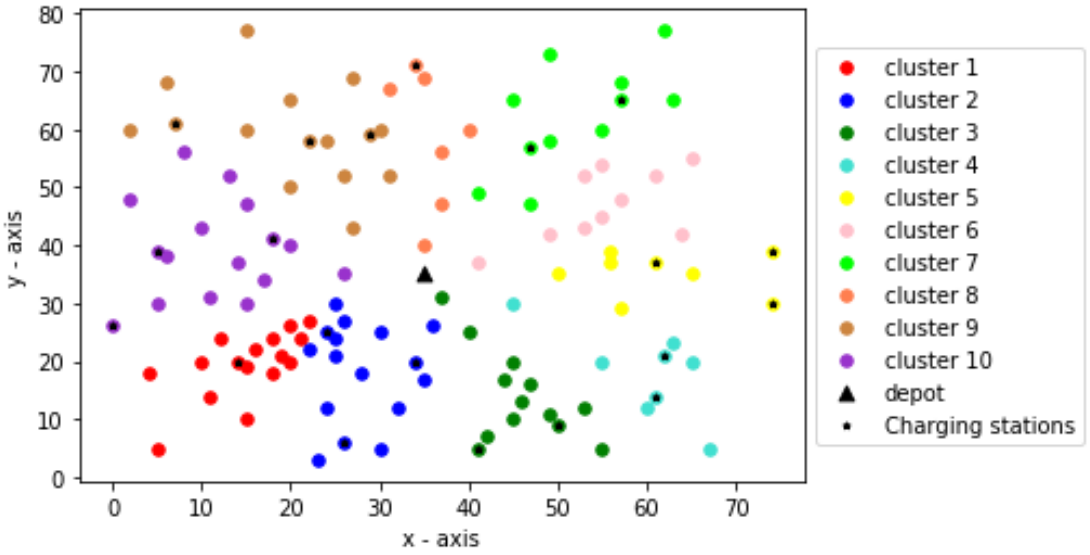
Ο συντελεστής d_{ik} υπολογίζεται με βάση τον παρακάτω τύπο:

$$d_{ik} = \min\left(c_{0i} + c_{ii_k} + c_{i_k i}, c_{0i_k} + c_{i_k i} + c_{i0}\right) - \left(c_{0i_k} + c_{i_k 0}\right) \quad (77)$$

Η αντικειμενική συνάρτηση (72) εκφράζει την ελαχιστοποίηση του κόστους προσθήκης των σημείων στην κυκλική διαδρομή από την αφετηρία προς τα σημεία «σπόρους». Η εξίσωση (73) εκφράζει ότι η ζήτηση που θα ανατεθεί σε κάθε υποομάδα θα πρέπει να μην ξεπερνάει την χωρητικότητα του οχήματος. Η εξίσωση (74) αντίστοιχα εκφράζει ότι κάθε σημείο θα πρέπει να αντιστοιχηθεί σε μία μόνο συστάδα, ενώ το σημείο την αφετηρίας θα ανήκει σε όλες. Τέλος, η εξίσωση (75) περιορίζει το πλήθος των σημείων που θα ανατεθούν σε κάθε συστάδα. Η τελευταία εξίσωση προστέθηκε εξαιτίας της δυσκολίας επίλυσης προβλημάτων με μεγάλο πλήθος σημείων.



Εικόνα 11 Γραφική αναπαράσταση σημείων «σπόρων» που χρησιμοποιήθηκαν για τον αλγόριθμο ομαδοποίησης για το σύνολο r201C100



Εικόνα 12 Γραφική αναπαράσταση των Ομάδων που δημιουργήθηκαν για το σύνολο r201C2100

Τελικό αποτέλεσμα ομαδοποίησης:

1η:[14, 16, 37, 38, 44, 59, 61, 85, 86, 91, 93, 96, 98, 99, 100]
2η:[2, 6, 13, 15, 42, 43, 57, 58, 87, 92, 94, 95, 97]
3η:[21, 22, 23, 40, 41, 53, 56, 72, 73, 74, 75]
4η:[4, 25, 26, 39, 55, 67]
5η:[12, 24, 54, 68, 80]
6η:[3, 28, 29, 33, 34, 76, 77, 78, 79, 81]
7η:[1, 9, 20, 35, 50, 51, 65, 66, 71]
8η:[27, 30, 32, 69, 70, 90]
9η:[7, 10, 11, 19, 31, 36, 49, 52, 62, 63, 64, 88]
10η:[5, 8, 17, 18, 45, 46, 47, 48, 60, 82, 83, 84, 89]

4.2.5 Δεδομένα Οχημάτων και σταθμών

Σύνολο	**Χωρητικότητα μπαταρίας**	**Χωρητικότητα φορτίου**	**Συντελεστής κατανάλωσης**	**Συντελεστής επαναφόρτισης**	**Πλήθος διαθέσιμων οχημάτων**	**Πλήθος διαθέσιμων CS**
rc105C5	60,63	1000	1	0,49	2	3
r104C5	60,63	200	1	0,49	2	3
r201C10	60,63	1000	1	0,49	3	4
r102C10	60,63	200	1	0,49	1	4
c106c15	77.75	200	1	3.47	3	2
c208c15	77.75	700	1	3.47	3	3
r201c100	77.75	700	1	3.47	15	21

Κεφάλαιο 5ο

5.1 Παρουσίαση αποτελεσμάτων

Στην συνέχεια παρουσιάζονται τα αποτελέσματα του κώδικα για κάθε Σύνολο σημείων. Πρώτα παρατίθενται τα αποτελέσματα της επίλυσης με το LRP μοντέλο και στην συνέχεια με το FCM. Οι διαδρομές που χρησιμοποιήθηκαν για το FCM προήλθαν από το αντίστοιχο VRP από το Παράρτημα. Τα αποτελέσματα αποτελούνται από τον χρόνο επίλυσης, το Optimality Gap, την τιμή της αντικειμενικής συνάρτησης και τις διαδρομές των οχημάτων.

5.1.1 Σύνολα 5 πελατών

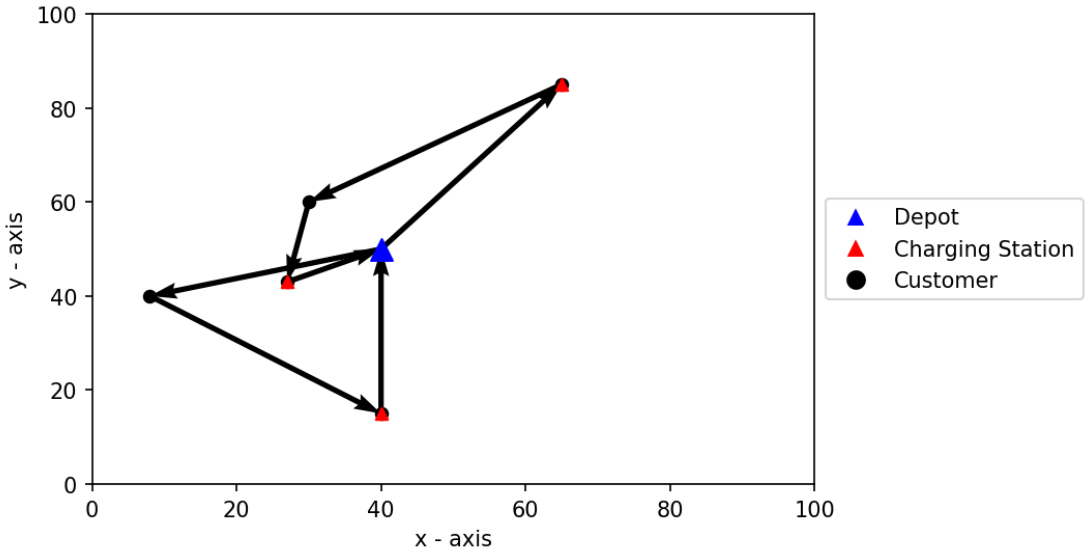
RC105C5

LRP

time = 24.812 s.
gap = 0.00869463%
Objective = 227.18

ROUTE 1
Route Order: 0-->1-->2-->0
Time Line: 0-->76.0-->146.0-->return at depot
battery line 77.75-->44.22-->3.61-->0

ROUTE 2
Route Order: 0-->5-->3-->4-->0
Time Line: 0-->59.0-->112.01-->139.27-->return at depot
battery line 77.75-->34.74-->17.26-->0-->0



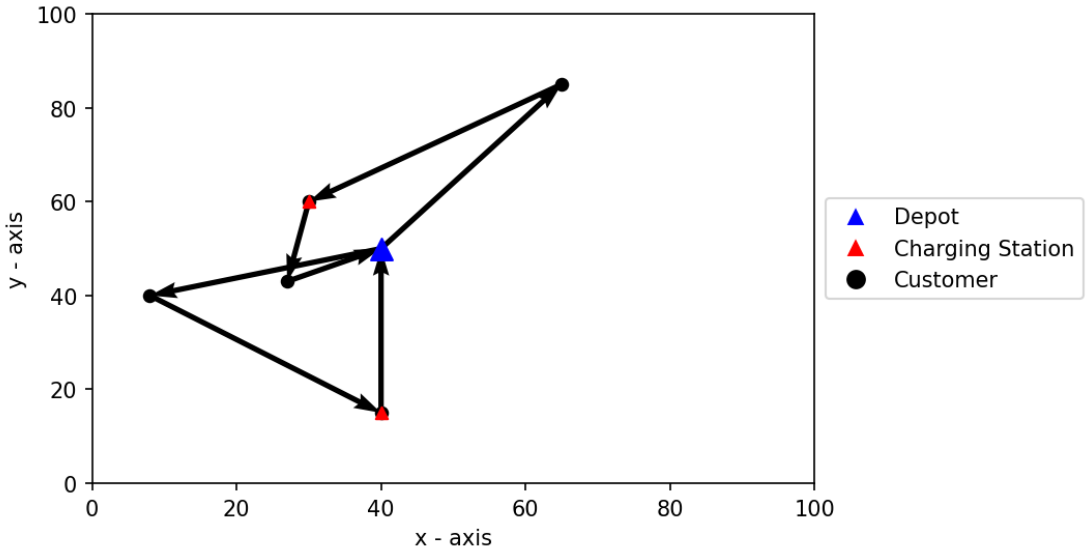
Εικόνα 13 Γραφική αναπαράσταση λύσης LRP συνόλου rc105C5

FCM

time = 0.015 s.
gap = 0%
Objective = 71.68

ROUTE 1
Route Order: 0-->1-->2-->0
Time Line: 0-->95.39-->146.0-->return at depot

ROUTE 2
Route Order: 0-->5-->3-->4-->0
Time Line: 0-->59.0-->116.0-->147.0-->return at depot



Εικόνα 14 Γραφική αναπαράσταση λύσης FCM συνόλου rc105C5

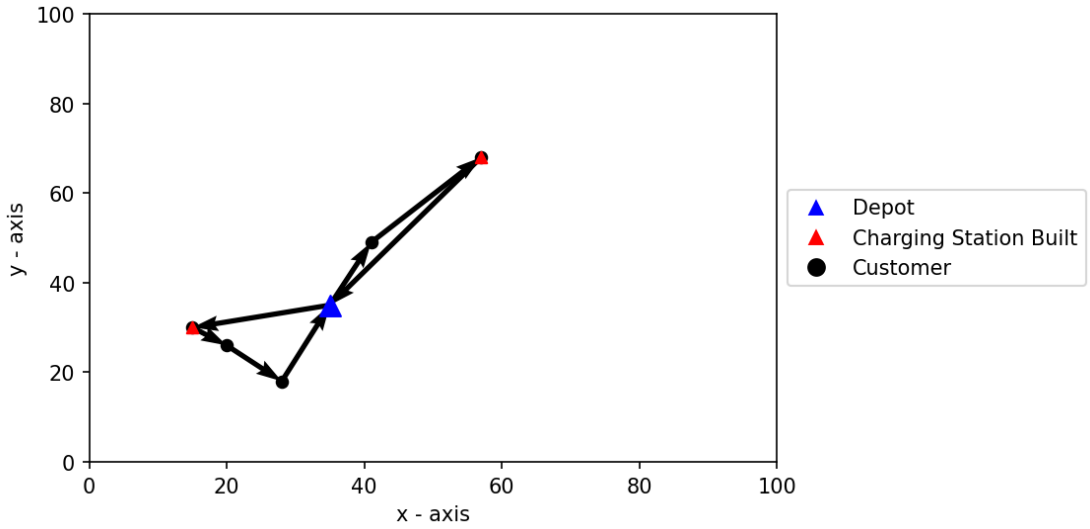
R104C5

LRP

time = 1.468 s.
gap = 0%
Objective = 136.44

ROUTE 1
Route Order: 0-->2-->1-->0
Time Line: 0-->46.0-->80.84-->return at depot
battery line 60.63-->45.4-->0--> return at depot

ROUTE 2
Route Order: 0-->3-->5-->4-->0
Time Line: 0-->138.1862-->154.69-->176.0-->return at depot
battery line 60.63-->40.01-->54.23-->18.38--> return at depot

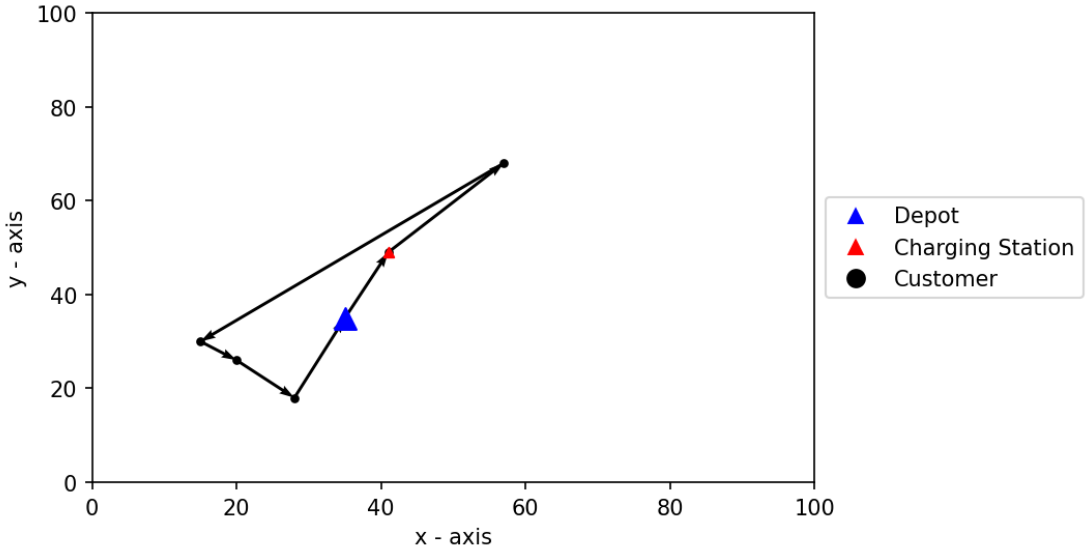


Εικόνα 15 Γραφική αναπαράσταση λύσης LRP συνόλου r104C5

FCM

time = 0 s.
gap = 0%
Objective = 120.89

ROUTE 1
Route Order: 0-->2-->1-->3-->5-->4-->0
Time Line: 0-->36.0-->70.84-->137.48-->153.88-->175.19-->return at depot



Εικόνα 16 Γραφική αναπαράσταση λύσης FCM συνόλου r104c5

5.1.2 Σύνολα 10 πελατών

R102C10

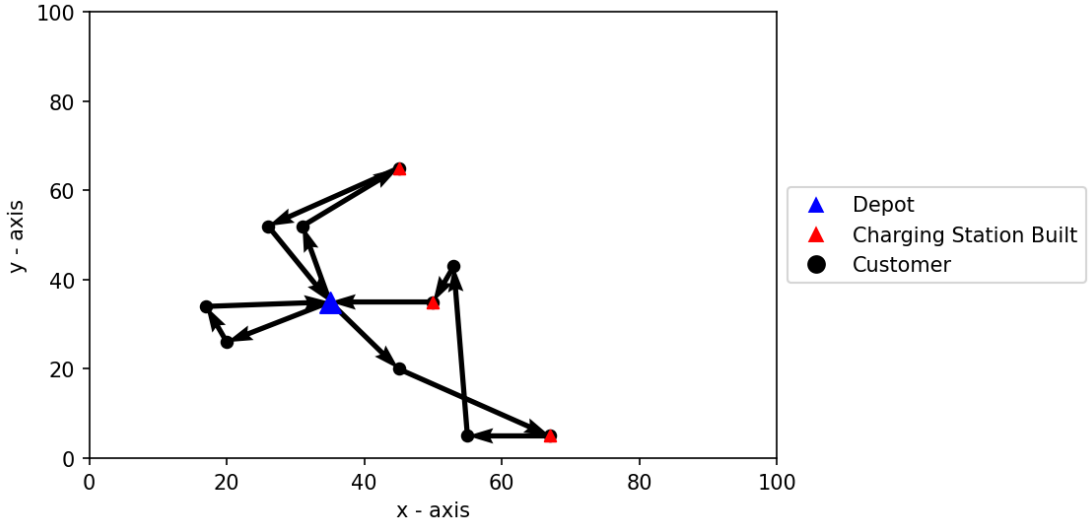
LRP

time = 7205.92 s.
gap = 4.08079%
Objective = 241.13

ROUTE 1
Route Order: 0-->1-->7-->5-->0
Time Line: 0-->25.0-->77.0-->166.0-->return at depot
battery line 60.63-->43.17-->0-->19.24--> return at depot

ROUTE 2
Route Order: 0-->8-->4-->0
Time Line: 0-->138.0-->201.0-->return at depot
battery line 60.63-->43.14-->18.03--> return at depot

ROUTE 3
Route Order: 0-->10-->3-->2-->6-->9-->0
Time Line: 0-->18.03-->54.66-->97.0-->160.0-->178.54-->return at depot
battery line 60.63-->42.6-->0-->46.59-->8.54-->-0.0--> return at depot



Εικόνα 17 Γραφική αναπαράσταση λύσης LRP συνόλου r102C10

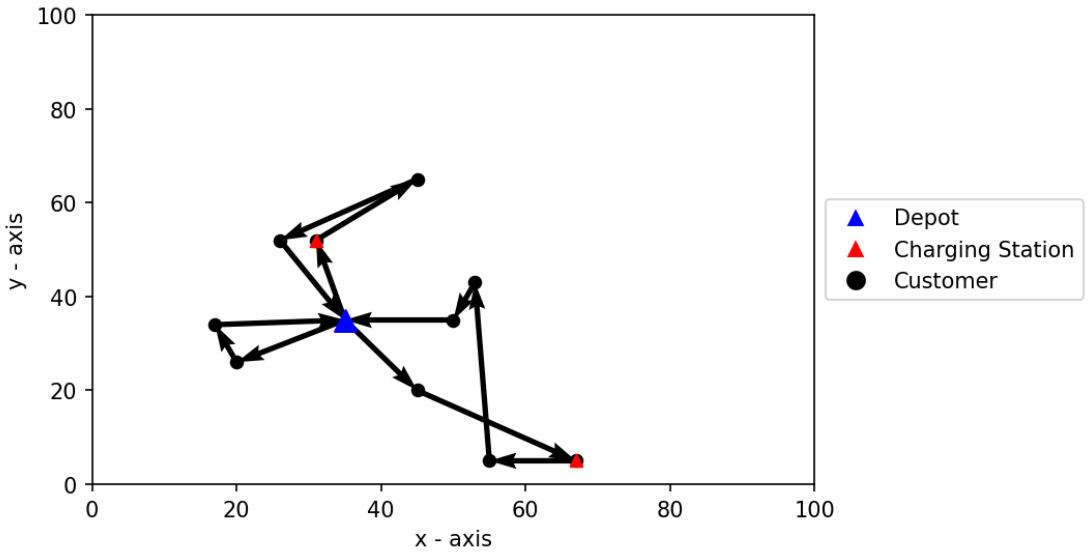
FCM

time = 0 s.
gap = 0%
Objective = 136.36

ROUTE 1
Route Order: 0-->1-->7-->5-->0
Time Line: 0-->25.0-->77.0-->166.0-->return at depot

ROUTE 2
Route Order: 0-->8-->4-->0
Time Line: 0-->148.0-->201.0-->return at depot

ROUTE 3
Route Order: 0-->10-->3-->2-->6-->9-->0
Time Line: 0-->18.03-->75.0-->97.0-->150.0-->187.0-->return at depot



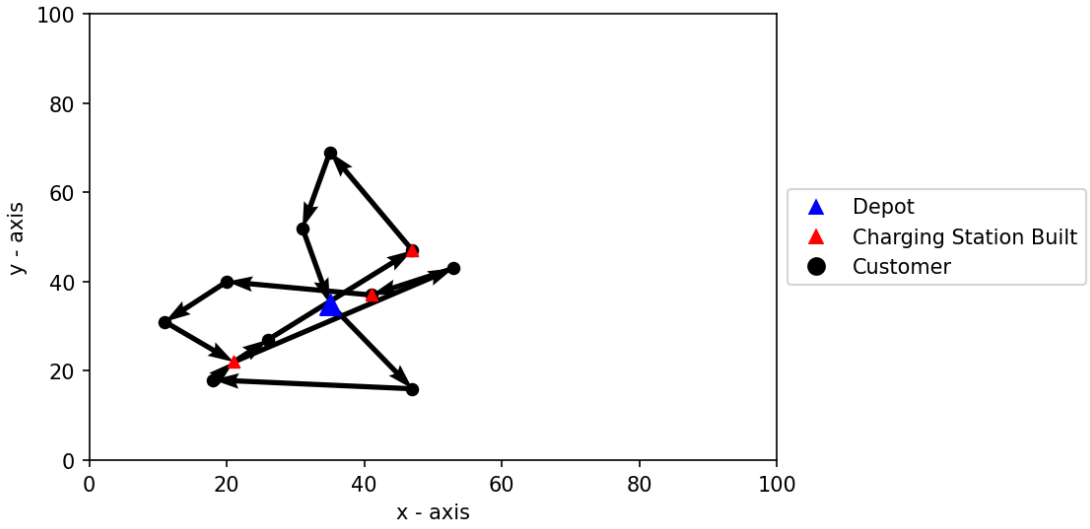
Εικόνα 18 Γραφική αναπαράσταση λύσης FCM συνόλου r102C10

R201C10

LRP

time = 7203.44 s.
gap = 19.2149%
Objective = 251.67999999999998

ROUTE 1
Route Order: 0-->8-->10-->13-->1-->4-->3-->7-->13-->9-->2-->5-->6-->0
Time Line: 0-->106.317-->145.387-->160.387-->224.0-->357.0-->407.9187-->448.0-->508.9746-->539.2657-->578.2657-->621.0-->648.46-->return at depot
battery line 60.63-->34.07-->5.0-->0-->13.42-->0-->39.42-->26.69-->13.24-->53.56-->24.56-->34.92-->17.46-->0

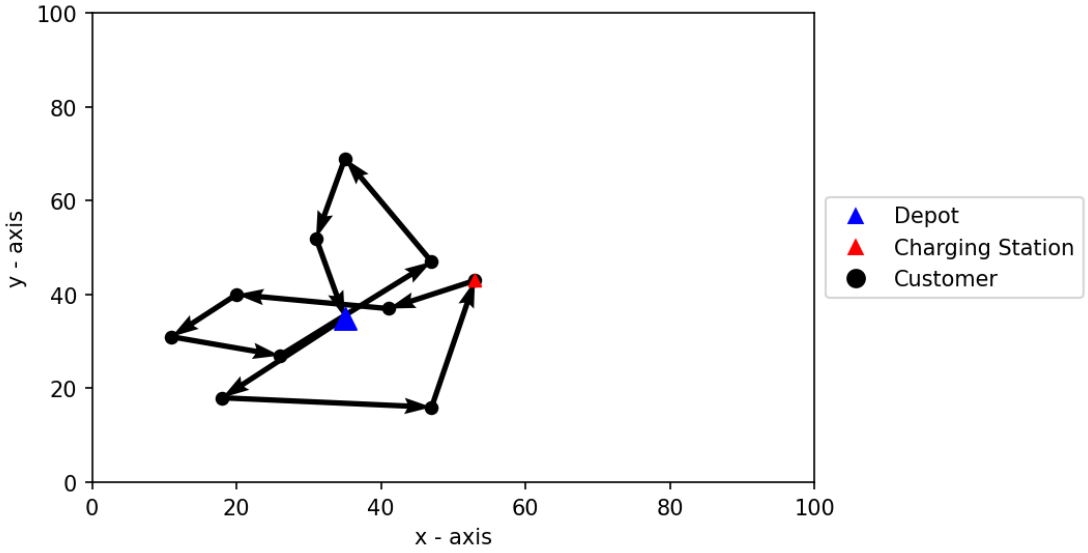


Εικόνα 19 Γραφική αναπαράσταση λύσης LRP συνόλου R201C10

FCM

time = 0 s.
gap = 0%
Objective = 146.6399

ROUTE 1
Route Order: 0-->10-->8-->1-->4-->3-->7-->9-->2-->5-->6-->0
Time Line: 0-->118.0-->157.07-->194.73-->371.79-->403.0-->448.0-->546.94-->
585.94-->621.0-->648.46-->return at depot



Εικόνα 20 Γραφική αναπαράσταση λύσης FCM συνόλου R201C10

5.1.3 Σύνολα 15 πελατών

C106C15

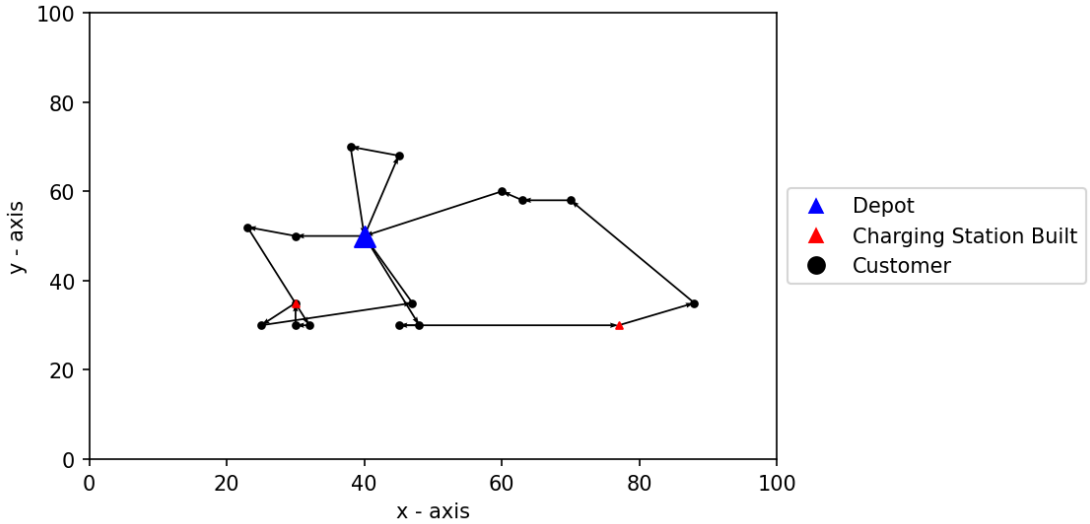
LRP

time = 169.672 s.
gap = 0.00974906%
Objective = 271.09000000000003

ROUTE 1
Route Order: 0-->5-->10-->18-->13-->14-->2-->3-->0
Time Line: 0-->204.0-->297.0-->419.0-->641.79-->761.0-->877.0-->970.61-->return at depot
battery line 77.75-->56.21-->53.21-->13.5367-->62.18-->32.97-->25.97-->22.36--> return at depot

ROUTE 2
Route Order: 0-->7-->6-->1-->9-->15-->8-->11-->0
Time Line: 0-->10.0-->263.0-->411.0-->503.0-->664.0-->761.07-->957.0-->return at depot
battery line 77.75-->67.75-->60.47-->36.7-->34.7-->29.7-->48.5666-->16.55--> return at depot

ROUTE 3
Route Order: 0-->12-->4-->0
Time Line: 0-->19.0-->760.0-->return at depot
battery line 77.75-->27.38-->20.1--> return at depot



Εικόνα 21 Γραφική αναπαράσταση λύσης LRP συνόλου C106C15

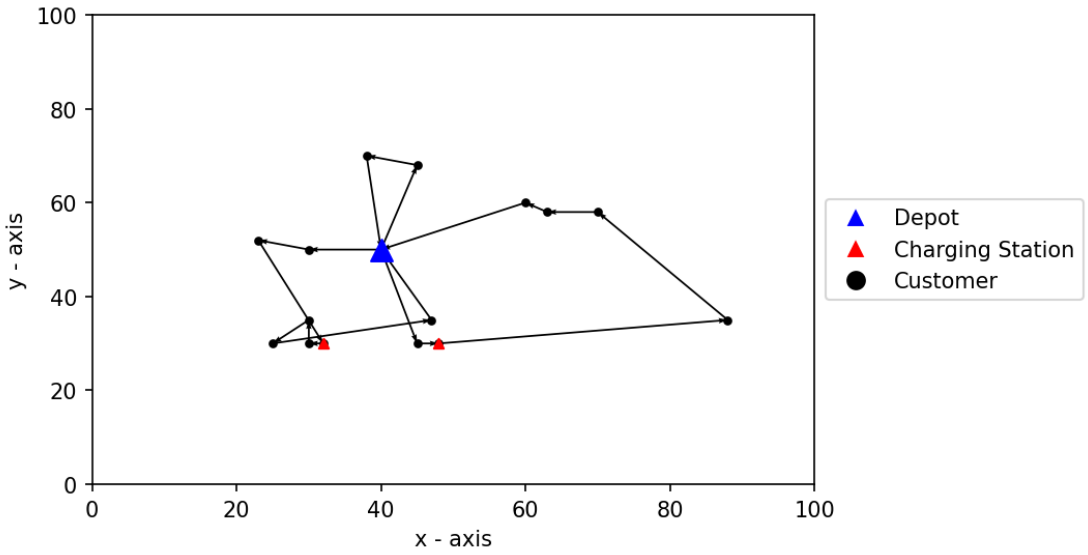
FCM

time = 0.015 s.
gap = 0%
Objective = 86.66

ROUTE 1
Route Order: 0-->7-->6-->1-->9-->15-->8-->11-->0
Time Line: 0-->10.0-->263.0-->411.0-->503.0-->664.0-->761.07-->957.0-->return at depot

ROUTE 2
Route Order: 0-->10-->5-->13-->14-->2-->3-->0
Time Line: 0-->277.0-->370.0-->597.0-->761.0-->877.0-->1123.0-->return at depot

ROUTE 3
Route Order: 0-->12-->4-->0
Time Line: 0-->19.0-->760.0-->return at depot



Εικόνα 22 Γραφική αναπαράσταση λύσης FCM συνόλου C106C15

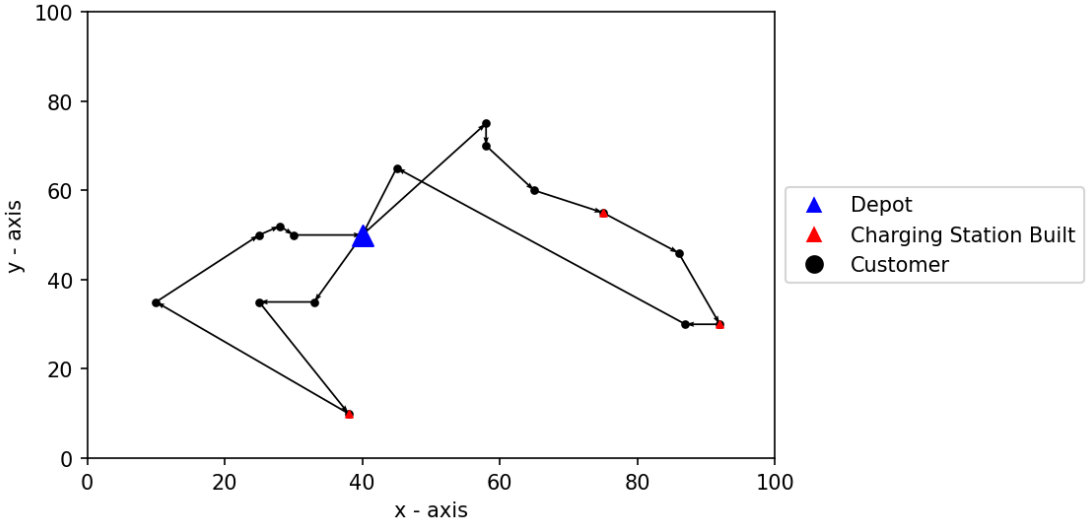
C208C15

LRP

time = 7218.91 s.
gap = 13.6205%
Objective = 293.9

ROUTE 1
Route Order: 0-->3-->15-->6-->8-->14-->12-->5-->4-->0
Time Line: 0-->79.0-->174.0-->276.21-->377.39-->869.1759-->976.2659-->1082.0-->1226.67-->return at depot
battery line: 77.75-->46.94-->41.94-->29.73-->18.55-->63.54-->46.45-->70.48-->15.81--> return at depot

ROUTE 2
Route Order: 0-->13-->11-->9-->7-->1-->10-->2-->0
Time Line: 0-->17.0-->162.0-->1056.0-->2020.0-->2645.0-->2738.61-->2831.44-->return at depot
battery line: 77.75-->61.2-->53.2-->25.02-->40.21-->16.44-->12.83-->10.0--> return at depot



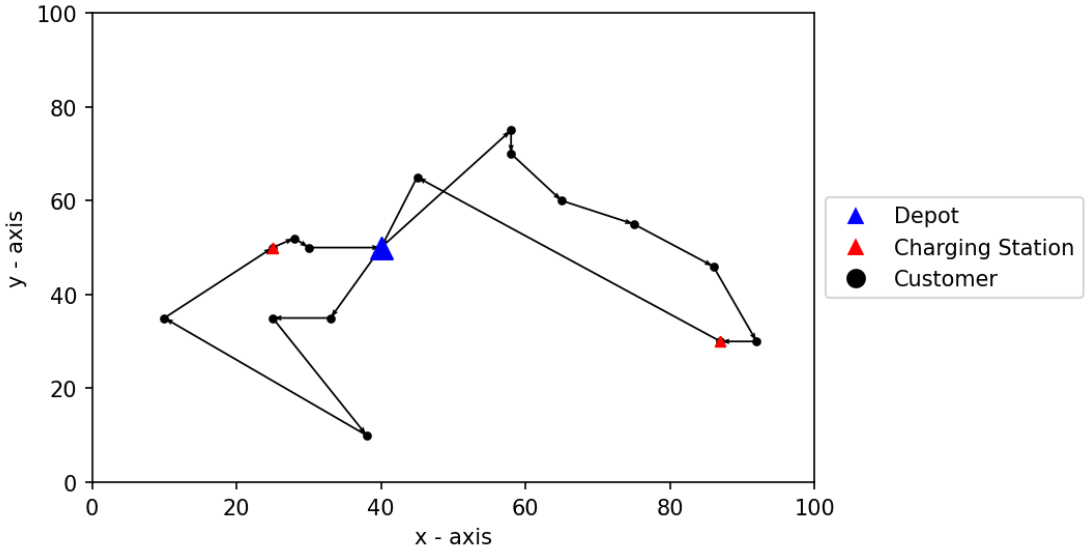
Εικόνα 23 Γραφική αναπαράσταση λύσης LRP συνόλου C208C15

FCM

time = 0.016 s.
gap = 0%
Objective = 140.96

ROUTE 1
Route Order: 0-->3-->15-->6-->8-->14-->12-->5-->4-->0
Time Line: 0-->469.79-->564.79-->667.0-->768.18-->879.91-->987.0-->1082.0-->1226.67--> return at depot

ROUTE 2
Route Order: 0-->13-->11-->9-->7-->1-->10-->2-->0
Time Line: 0-->64.0-->162.0-->1056.0-->2020.0-->2645.0-->2738.61-->2831.44--> return at depot



Εικόνα 24 Γραφική αναπαράσταση λύσης FCM συνόλου C208C15

5.1.4 Σύνολο 100 πελατών

R201C100

LRP

Συνολική απόσταση: 1350.2290

Πλήθος σταθμών: 17

Πλήθος οχημάτων: 13

Οι επιπρόσθετες πιθανές θέσεις CS που ανατέθηκαν σε κάθε ομάδα παρουσιάζονται στην συνέχεια, με βάση το κριτήριο της κοντινότερης γειτνίασης:

Ομάδα 1η: [112]

Ομάδα 2η: [114,115,116]

Ομάδα 3η:[117,118]

Ομάδα 4η:[119,120]

Ομάδα 5η:[102,103,121]

Ομάδα 6η:[102,103]

Ομάδα 7η:[104,105]

Ομάδα 8η:[107]

Ομάδα 9η:[106,108,109]

Ομάδα 10η:[110,111,113]

Ομάδα 1η: [14, 16, 37, 38, 44, 59, 61, 85, 86, 91, 93, 96, 98, 99, 100]

time = 2811.39 s.
gap = 0.00968112%
Objective = 137.6

ROUTE 1
Route Order: 0-->99-->59-->100-->14-->37-->98-->61-->93-->85-->91-->16-->44-->38 -->86-->96-->0
Time Line: 0-->101.0-->113.24-->148.0-->304.0-->500.0-->533.38-->551.0-->567.0--> 579.83-->592.99-->608.09-->684.6346-->733.0-->917.88-->948.0-->return at depot
battery line 77.75-->60.26-->58.02-->59.3936-->50.8536-->39.6736-->47.4768--> 39.8568-->33.8568-->31.0268-->27.8668-->22.7668-->0-->0-->36.4816-->15.26-->0

Ομάδα 2η: [2, 6, 13, 15, 42, 43, 57, 58, 87, 92, 94, 95, 97]

time = 7210.66 s.
gap = 19.5892%
Objective = 154.25

ROUTE 1
Route Order: 0-->13-->2-->42-->95-->92-->87-->97-->94-->6-->0
Time Line: 0-->81.49-->100.92-->123.0-->179.0-->223.0-->530.0-->566.0-->642.0--> 708.0-->return at depot
battery line 77.75-->66.57-->57.14-->45.06-->33.02-->29.41-->22.2-->17.96-->11.88--> >11.18-->0
ROUTE 2
Route Order: 0-->58-->57-->43-->15-->0
Time Line: 0-->511.71-->536.27-->559.0-->861.0-->return at depot
battery line 77.75-->64.98-->50.42-->37.69-->30.41-->0

Ομάδα 3η: [21, 22, 23, 40, 41, 53, 56, 72, 73, 74, 75]

time = 768.484 s.
gap = 0.00826173%
Objective = 121.039

ROUTE 1
Route Order: 0-->21-->72-->23-->74-->75-->56-->22-->41-->73-->0
Time Line: 0-->67.0-->104.0-->274.0-->455.0-->635.0-->767.0-->885.0-->912.0--> 960.0-->return at depot
battery line 77.75-->59.72-->55.25-->41.65-->29.61-->15.3018-->11.1818-->34.56--> 30.32-->20.12-->0

ROUTE 2
Route Order: 0-->53-->40-->0
Time Line: 0-->388.0-->733.0-->return at depot
battery line 77.75-->73.28-->11.18-->0

Ομάδα 4η: [4, 25, 26, 39, 55, 67]

time = 90.688 s.
gap = 0.0096609%
Objective = 103.50999999999999

ROUTE 1
Route Order: 0-->26-->25-->67-->39-->55-->4-->0
Time Line: 0-->134.0-->173.87-->199.0-->284.0-->420.0-->753.0-->return at depot
battery line 77.75-->58.79-->36.43-->21.3-->11.4-->0-->25.0-->0

Ομάδα 5η: [12, 24, 54, 68, 80]

time = 7.593 s.
gap = 0%
Objective = 72.71

Route Order: 0-->54-->80-->68-->24-->12-->0
Time Line: 0-->144.0-->184.0-->466.0-->825.0-->850.0-->return at depot
battery line 77.75-->54.95-->46.89-->44.89-->35.04-->15.0-->0

Ομάδα 6η: [3, 28, 29, 33, 34, 76, 77, 78, 79, 81]

time = 83.219 s.
gap = 0.00763009%
Objective = 131.06

ROUTE 1
Route Order: 0-->77-->79-->81-->34-->78-->33-->76-->28-->29-->3-->0
Time Line: 0-->86.0-->102.4-->118.72-->138.77-->311.23-->329.23-->350.0-->495.0-->847.0-->945.0-->return at depot
battery line 77.75-->52.2606-->45.8606-->39.5406-->29.4906-->72.75-->64.75-->53.98-->55.39-->31.85-->22.36-->0

Ομάδα 7η :[1, 9, 20, 35, 50, 51, 65, 66, 71]

time = 7203.11 s.
gap = 26.6937%
Objective = 169.64

ROUTE 1
Route Order: 0-->1-->65-->9-->50-->20-->66-->71-->35-->51-->0
Time Line: 0-->102.0-->215.0-->558.0-->583.26-->679.0-->697.94-->717.37-->868.6402-->894.2902-->return at depot
battery line 77.75-->62.52-->27.52-->59.37-->44.11-->26.0-->17.06-->7.63-->42.58-->26.93-->0

Ομάδα 8η :[27, 30, 32, 69, 70, 90]

time = 2.875 s.
gap = 0%
Objective = 104.78999999999999

ROUTE 1
Route Order: 0-->70-->27-->90-->32-->30-->69-->0

Time Line: 0-->120.0-->405.9125-->532.0-->546.47-->566.77-->592.0-->return at depot
battery line 77.75-->0-->61.63-->40.28-->35.81-->25.51-->12.17-->0

Ομάδα 9η :[7, 10, 11, 19, 31, 36, 49, 52, 62, 63, 64, 88]

time = 7206.08 s.
gap = 25.7939%
Objective = 172.77

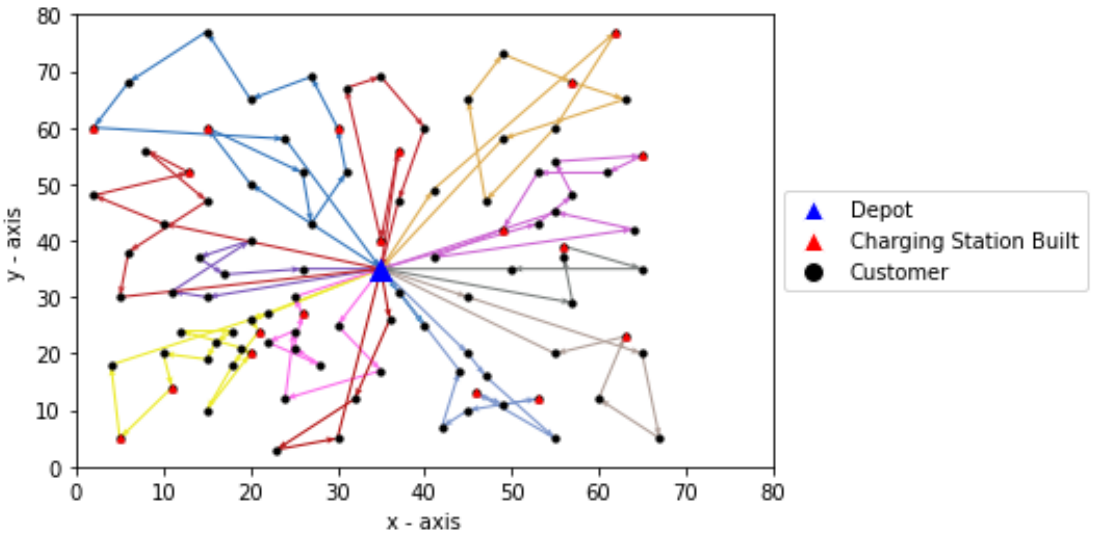
ROUTE 1
Route Order: 0-->7-->19-->88-->52-->31-->10-->63-->11-->64-->49-->36-->62-->0
Time Line: 0-->110.0-->233.0-->371.94-->391.0-->591.0-->609.06-->779.0-->797.06-->820.06-->842.79-->861.73-->964.0-->return at depot
battery line 77.75-->56.54-->34.9131-->57.4342-->48.3742-->38.5242-->30.4642-->67.2134-->59.1534-->46.1534-->33.4234-->24.4834-->25.5-->0

Ομάδα 10η :[5, 8, 17, 18, 45, 46, 47, 48, 60, 82, 83, 84, 89]

time = 7208.64 s.
gap = 29.5089%
Objective = 182.86

ROUTE 1
Route Order: 0-->5-->84-->18-->83-->60-->89-->0
Time Line: 0-->21.0-->314.0-->513.0-->545.0-->680.94-->700.0-->return at depot
battery line 77.75-->57.13-->41.74-->29.01-->22.3-->18.06-->9.0-->0

ROUTE 2
Route Order: 0-->8-->46-->48-->47-->82-->45-->17-->0
Time Line: 0-->71.0-->194.0-->418.0-->675.6-->697.0-->779.0-->797.06-->return at depot
battery line 77.75-->51.5-->42.07-->5.3581-->71.35-->59.95-->38.47-->30.41-->0



Εικόνα 25 Γραφική αναπαράσταση λύσης LRP συνόλου R201C100

FCM

Συνολική απόσταση= 1096.98
Πλήθος σταθμών: 8
Πλήθος οχημάτων: 8

ROUTE 1: 164.92000000000002
ROUTE 2: 102.33999999999997
ROUTE 3: 88.19999999999999
ROUTE 4: 153.19
ROUTE 5: 80.03
ROUTE 6: 130.46
ROUTE 7: 235.54999999999998
ROUTE 8: 142.29000000000002

ROUTE 1
Route Order: 0-->1-->70-->30-->32-->66-->9-->35-->71-->20-->51-->3-->29-->24-->12-->0
Time Line: 0-->16.0-->34.06-->520.0-->547.0-->675.68-->700.0-->726.0-->742.71-->765.08-->783.14-->807.46-->847.0-->864.07-->889.07-->return at depot

ROUTE 2
Route Order: 0-->2-->42-->100-->14-->43-->57-->87-->97-->94-->6-->0
Time Line: 0-->100.92-->123.0-->141.49-->174.0-->528.06-->550.79-->568.0-->582.24-->642.0-->655.16-->return at depot

ROUTE 3
Route Order: 0-->7-->46-->8-->18-->83-->60-->89-->0
Time Line: 0-->32.0-->60.11-->79.54-->513.0-->545.0-->589.0-->700.0-->return at depot

ROUTE 4
Route Order: 0-->13-->95-->59-->99-->5-->84-->61-->92-->37-->98-->93-->85-->91-->16-->44-->38-->15-->41-->73-->0
Time Line: 0-->12.0-->27.1-->88.76-->101.0-->117.4-->314.0-->378.8-->399.0-->500.0-->541.0-->565.0-->577.83-->617.0-->665.0-->681.08-->701.9-->861.0-->883.17-->903.37-->return at depot

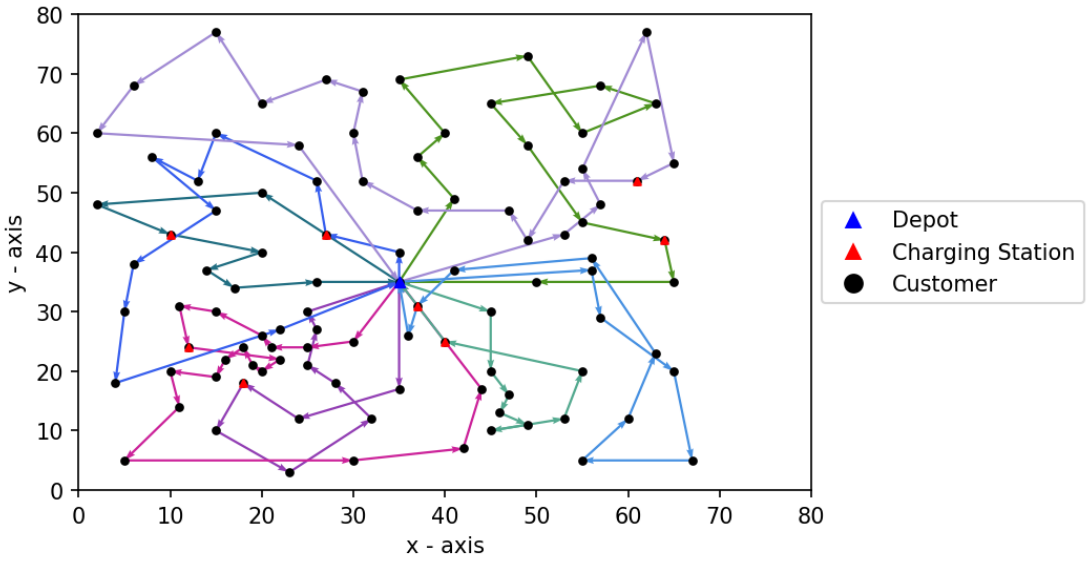
ROUTE 5
Route Order: 0-->26-->21-->72-->74-->75-->22-->56-->4-->40-->0
Time Line: 0-->134.0-->213.0-->227.47-->455.0-->537.0-->765.0-->783.25-->803.19-->829.0-->return at depot

ROUTE 6

Route Order: 0-->27-->52-->88-->19-->48-->47-->82-->45-->17-->86-->96-->0
Time Line: 0-->301.8-->320.34-->339.4-->363.0-->558.0-->763.0-->784.4-->807.13-->825.19-->877.0-->907.12-->return at depot

ROUTE 7
Route Order: 0-->77-->79-->81-->65-->34-->78-->33-->76-->50-->69-->31-->10-->90-->63-->11-->64-->49-->36-->62-->0
Time Line: 0-->86.0-->108.44-->124.76-->158.8-->191.0-->230.0-->249.0-->269.77-->507.0-->527.0-->591.0-->609.06-->636.0-->779.0-->797.06-->820.06-->842.79-->861.73-->964.0-->return at depot

ROUTE 8
Route Order: 0-->80-->54-->25-->67-->23-->39-->55-->68-->28-->53-->58-->0
Time Line: 0-->22.0-->144.0-->173.87-->199.0-->221.0-->284.0-->420.0-->447.46-->472.59-->489.8-->515.0-->return at depot



Εικόνα 26 Γραφική αναπαράσταση λύσης FCM συνόλου R201C100

5.2 Σχολιασμός αποτελεσμάτων προβλήματος και γενικές παρατηρήσεις

Το LRP μοντέλο απαιτεί μεγάλο πλήθος μεταβλητών, περιορισμών και απαιτεί μεγάλη υπολογιστική δύναμη για την επίλυση του. Οπότε, σε ορισμένα από τα παραδείγματα που παρουσιάστηκαν δεν έχουμε την βέλτιστη λύση λόγω εξάντλησης του χρόνου. Αντιθέτως, στο FCM ο χρόνος ήταν ικανοποιητικός και όλα τα παραδείγματα επιλύθηκαν εντός χρονικού περιθωρίου.

Όσον αφορά την δρομολόγηση των οχημάτων και στις δυο προσεγγίσεις, ως επί το πλείστων, οι διαδρομές είναι πανομοιότυπες. Οι διαφορές στα παραδείγματα των 5, 10, 15 σημείων πελατών παρουσιάζονται στα παραδείγματα R104C5, R201C10, C106C15. Στο παράδειγμα με τα 5 σημεία πελατών η διαφορά παρουσιάζεται γιατί στο LRP λαμβάνεται υπόψιν η εμβέλεια των οχημάτων και το διαθέσιμο πλήθος σταθμών που μπορούν να εγκατασταθούν. Αναλυτικά, παρόλο που και στα δύο μοντέλα η αντικειμενική στοχεύει στην εύρεση της ελάχιστης απόστασης, στο LRP χρησιμοποιούνται δύο οχήματα προκειμένου να ικανοποιούνται οι περιορισμοί που αναφέρθηκαν προηγουμένως, γεγονός που δεν ισχύει για το FCM καθώς οι περιορισμοί αυτοί δεν περιλαμβάνονται στο μοντέλο και όπως φαίνεται η απόσταση που διανύεται με το ένα όχημα είναι μικρότερη σε σύγκριση με τα δύο οχήματα. Στο παράδειγμα R201C10 και στις δύο περιπτώσεις μοντέλων χρησιμοποιείται μόνο ένα όχημα, στην δρομολόγηση όμως παρουσιάζονται διαφορές για τους ίδιους λόγους όπως στο παράδειγμα R104C5. Τέλος, στο Σύνολο C106C15 τα δύο από τα τρία δρομολόγια είναι πανομοιότυπα, ενώ στο τρίτο υπάρχουν διαφορές. Στην LRP επίλυση, το εν λόγω δρομολόγιο είναι το ROUTE 1, ενώ στο FCM το ROUTE 2. Σε αυτό το παράδειγμα, η διαφορά προκύπτει εξαιτίας του απαιτούμενου χρόνου για φόρτιση και των χρονικών παραθύρων των πελατών που δύναται να είναι και σταθμοί φόρτισης.

Σε όλα τα παραδείγματα παρουσιάζονται διαφορές στην χωροθέτηση των σταθμών φόρτισης. Οι διαφορές αυτές είναι αρχικά στο πλήθος των σταθμών που ορίζονται, καθώς στο FCM θεωρείται ότι μπορεί να καλυφθεί όλη η ζήτηση της διαδρομής από μόνο έναν σταθμό, γεγονός που σε αρκετά παραδείγματα δεν ισχύει, μιας και η συνολική ζήτηση συνήθως ξεπερνάει την χωρητικότητα της μπαταρίας. Στην συνέχεια, εντοπίζεται ότι οι θέσεις που έχουν επιλεχθεί είναι διαφορετικές. Αυτό προκύπτει από το γεγονός ότι το LRP λαμβάνει υπόψιν την εμβέλεια του οχήματος, τον χρόνο που απαιτείται για την φόρτιση και την δυνατότητα φόρτισης σε ένα σημείο πολλαπλές φορές. Συγκεκριμένα, στο C208C15 στην διαδρομή 2 ο σταθμός έχει εγκατασταθεί στο σημείο 9 για να μπορεί, να υπάρχει δηλαδή η απαιτούμενη ενέργεια, για να διασχιστεί στην συνέχεια η διαδρομή προς το 7. Αντίστοιχα, στο R201C10 είναι απαραίτητη η εγκατάσταση πολλαπλών φορτιστών για την εξυπηρέτηση των σημείων καθώς η συνολική διαδρομή ξεπερνάει το τριπλάσιο της αυτονομίας του οχήματος. Επιπλέον, σε αυτό το παράδειγμα παρουσιάζεται και η δυνατότητα πολλαπλών επισκέψεων σε ένα σημείο φόρτισης(ROUTE 1, CS13). Τέλος, στο παράδειγμα C103C15 παρατηρείται η σημασία των θέσεων σταθμών φόρτισης

που δεν αντιστοιχούν σε τοποθεσίες πελατών(ROUTE 1, CS 18) μιας και ο χρόνος που απαιτείται για φόρτιση είναι περισσότερος απ’ ότι ο χρόνος που είναι διαθέσιμος ο πελάτης, σε συνδυασμό με την μεγάλη απόσταση μεταξύ πελατών που απαιτεί ενδιάμεση στάση για να μπορεί να διασχιστεί.

Τέλος, διαφορές παρουσιάζονται ανάμεσα στις λύσεις για το σύνολο R201C100. Αρχικά, στο LRP δρομολογήθηκα 13 οχήματα και χρειάστηκαν 17 CS, ενώ στο FCM μόλις 8 οχήματα και 8 CS. Προφανώς, διαφορές παρουσιάζονται στην δρομολόγηση ανάμεσα στις λύσεις του συνόλου R201C100. Οι διαφορές αυτές προκύπτουν από το γεγονός ότι δημιουργήθηκαν συστάδες για την χρήση του LRP μοντέλου και απαιτήθηκαν περισσότερα οχήματα, ενώ για το FCM χρησιμοποιήθηκε η λύση που προέκυψε από την δρομολόγηση και των 100 σημείων πελατών ταυτοχρόνως. Σημαντικό είναι επίσης να αναφερθεί ότι στις δύο προσεγγίσεις οι θέσεις CS που έχουν επιλεχθεί αποτελούν μόνο σημεία πελατών.

5.3 Μελλοντική έρευνα

Η μελλοντική έρευνα θα πρέπει να υιοθετήσει μια ολοκληρωμένη προσέγγιση στη μοντελοποίηση του σχεδιασμού και της χωροθέτησης των CS υποδομών. Μπορεί να πραγματοποιηθεί έρευνα με τη χρήση πραγματικών δεδομένων, ενώ λαμβάνονται υπόψιν η έννοια της μη γραμμικής φόρτισης και η αλληλεπίδραση της ζήτησης και του κόστους φόρτισης. Τέλος, μελλοντική έρευνα χρειάζεται να επικεντρωθεί σε πιο αποδοτικές μεθόδους επίλυσης, καθώς τα ρεαλιστικά προβλήματα αποτελούνται από περισσότερα σημεία πελατών.

Παράρτημα

Code LRP

```
#επιστρέφει την θέση στην λίστα των εικονικών κόμβων στη οποία είναι ο εικονικός κομβος
def get_index(val,dictionary):
    key=get_key(val,dictionary)
    index=dictionary[key].index(val)
    return index

#επιστρεφει τον πραγματικό κ
#κομβο που αντιστοιχεί ο εικόνικός κομβος
def get_key(val,dictionary):
    for key, value in dictionary.items():
        for i in value:
            if val == i:
                return key

    return "key doesn't exist"

#επιστρέφει μεταβλητή που χρησιμοποιείται για την αντιστοίχιση των δεδομένων απόστασης matrix[i,j] για τους εικονικούς κόμβους
def get_node(i):
    if i in total_nodes:
        node=i
        #return node
    elif i in nodes_0_n1:
        node=0
        #return node
    else:
        node=get_key(i,dct)
        #return node
    return node


import numpy as np
#from scipy.spatial import distance
import pandas as pd

#Διαβάζω το αρχείο με τους κόμβους από πάνω οι πελάτες και κάτω οι σταθμοί, σωστή διαμόρφωση αρχείου
column_names = ["StringID", "Type", "x", "y", "demand", "ReadyTime", "DueDate", "ServiceTime"]
df = pd.read_csv("r201C100.csv", names=column_names,skiprows=1)
#print(df)
```

```
xloc=df.x.to_list()
yloc=df.y.to_list()
StringID=df.StringID.to_list()
Type=df.Type.to_list()
demand=df.demand.to_list()
ReadyTime=df.ReadyTime.to_list()
DueDate=df.DueDate.to_list()
ServiceTime=df.ServiceTime.to_list()

for i in range(len(xloc)):
   xloc[i]=int(xloc[i])
   yloc[i]=int(yloc[i])
   demand[i]=float(demand[i])
   ReadyTime[i]=float(ReadyTime[i])
   DueDate[i]=float(DueDate[i])
   ServiceTime[i]=float(ServiceTime[i])

nodeSet=[]
for i in range(len(StringID)):
   nodeloc=[xloc[i],yloc[i]]
   nodeSet.append(nodeloc)
#nodeSet

df = pd.DataFrame(nodeSet, columns=['xcord', 'ycord'], index=StringID)
#print(df)
n_df=(df.values)
#print(n_df)

#print((df.values).shape)

matrix=np.zeros(((df.values).shape[0],(df.values).shape[0]))
#print(matrix)


for i in range((df.values).shape[0]):
   for j in range((df.values).shape[0]):
      matrix[i,j]=round(np.sqrt(np.sum((n_df[i]-n_df[j])**2)),2)


#δημιουργία πίνακα χρόνων, ορισμός ταχύτητας πρώτα
velocity=1
Tij=matrix.copy()
Tij=(velocity**(-1))*Tij
#print(Tij)


read_c=[]
```

```
read_cs=[]
read_depot=[]
for i in range(len(StringID)):
   if Type[i]=='d':
      read_depot.append(i)
   elif Type[i]=='c':
      read_c.append(i)
   elif StringID[i]=='S0':
      continue
      #break
   elif Type[i]=='f':
      read_cs.append(i)
print(read_c,'\n',read_depot,'\n',read_cs)

dct = {}
customers =[21, 22, 23, 40, 41, 53, 56, 72, 73, 74, 75]
stations=[117,118]
total_nodes=customers.copy()

for i in stations:
   total_nodes.append(i)



lists_All_Dummies=[]
#περιέχει τις λιστες με τους εικονικούς σταθμους
#η καθε λίστα περιέχει τους εικονικούς σταθμους που αντιστοιχουν σε κάθε ενα κομβο ξεχωριστά

list_all_nodes=total_nodes.copy()#λιστα με όλους τους κομβους πελατων, σταθμών και εικονικών σταθμών.

#lastdummy=total_nodes[-1]
lastdummy=121
list_of_dummies=[]
#επαναληπτική για να δημιουργω εικονικους για κάθε κομβο φορτισης
for j in total_nodes:
   list_Dummies_per_Station=[]

   #δημιουργία εικονικών σταθμών
   for i in customers:
      lastdummy=lastdummy+1
      list_all_nodes.append(lastdummy)
      #list_of_dummies.append(lastdummy) # creates a list with all the dummies
      list_Dummies_per_Station.append(lastdummy)

   lists_All_Dummies.append(list_Dummies_per_Station)
```

```
   #λεξικό που περιέχει ανα κλειδί(πιθανό σταθμό) την λίστα με του εικονικούς σταθμούς που του αντιστοιχούν
   dct[j] = list_Dummies_per_Station
Only_Dummies=list_all_nodes[len(total_nodes):]  # creates a list with all the dummies (less time and effort)

################
#create all sets of nodes
V=list_all_nodes.copy()
V_0=list_all_nodes.copy()
V_0.append(0)
V_0_n1=list_all_nodes.copy()
V_0_n1.append(V_0_n1[-1]+1)
V_0_n1.append(0)
V_n1=V_0_n1[:-1]
C=customers.copy()
C_0=customers.copy()
C_0.append(0)
S=Only_Dummies.copy()
Si=dct.copy()
V_S=total_nodes.copy()
V_C=stations.copy()
for i in Only_Dummies:
   V_C.append(i)
#print('V_C = ',V_C)
#print('V = ',V,'\n')
#print('V_0 = ',V_0,'\n')
#print('V_0_n1 = ',V_0_n1,'\n')
#print('V_n1 = ',V_n1,'\n')
nodes_0_n1=[]
nodes_0_n1.append(0)
nodes_0_n1.append(V_n1[-1])
#print('nodes_0_n1=',nodes_0_n1)
#print('total_nodes=',total_nodes)
lastnode=V_n1[-1]

#batery capacity
Q=77.75
#consumption rate
O=1
#recharge rate
r=3.47
#Charging stations
OUB=3
#CS lower Bound
OLB=0
#vehicles
```

```
NUB=2
#vehicle lower bound
NLB=0
#freight cap
F=700




from copy import deepcopy
V_S_0=V_S.copy()
V_S_0.append(0)
V_S_n1=V_S.copy()
V_S_n1.append(lastnode)
V_S_0_n1=V_S_0.copy()
V_S_0_n1.append(lastnode)
#V_S_0
#V_S_0_n1
#infeasible due to time windows
Feasible_arc={}
InFeasible_arc={}
for i in V_S_0:
   node_i=get_node(i)
   feasible_arc_ends=[]
   infeasible_arc_ends=[]
   for j in V_S_n1:
      node_j=get_node(j)
      if
ReadyTime[node_i]+ServiceTime[node_i]+Tij[node_i,node_j]<=DueDate[node_j]:
         feasible_arc_ends.append(j)
      else:
         infeasible_arc_ends.append(j)
   Feasible_arc[i]=feasible_arc_ends
   InFeasible_arc[i]=infeasible_arc_ends

#infeasible due to time windows depot return
last_node=V_n1[-1]
depot=get_node(last_node)
for i in V_S_0:
   node_i=get_node(i)
   for j in V_S:
      node_j=get_node(j)

      if
ReadyTime[node_i]+ServiceTime[node_i]+Tij[node_i,node_j]+ServiceTime[node_j]+T
ij[node_j,depot]<=DueDate[depot]:
```

```
            if j in Feasible_arc[i]:
                pass
            elif j in InFeasible_arc[i]:
                pass
            else:
                Feasible_arc[i].append(j)

        else:
            if j in InFeasible_arc[i]:
                pass
            else:
                InFeasible_arc[i].append(j)

#infeasible due to capacity
for i in C:
    for j in C:

        if demand[i]+demand[j]<=F:
            if j in Feasible_arc[i]:
                pass
            elif j in InFeasible_arc[i]:
                pass
            else:
                Feasible_arc[i].append(j)
        else:
            if j in InFeasible_arc[i]:
                pass
            else:
                InFeasible_arc[i].append(j)

#infeasible due to distance
for i in V_S_0:
    node_i=get_node(i)
    for j in V_S_0_n1:
        node_j=get_node(j)

        if O*matrix[node_i,node_j]<=Q:
            if j in Feasible_arc[i]:
                pass
            elif j in InFeasible_arc[i]:
                pass
            else:
                Feasible_arc[i].append(j)
        else:
            if j in InFeasible_arc[i]:
                pass
            else:
```

```
            InFeasible_arc[i].append(j)

#add dummy vertices for the real feasible nodes
Feasible_arc_final=deepcopy(Feasible_arc)
for i in Feasible_arc:

   for j in Feasible_arc[i]:
      if j in V_S_n1 and (j!=i):
         if i!=0 and j!=lastnode:
            for dummy in dct[j]:
               Feasible_arc_final[i].append(dummy)
         if i==0 and j!=lastnode:
            for dummy in dct[j]:
               Feasible_arc_final[i].append(dummy)



   if i == 0:
      print(i,len(Feasible_arc_final[i]))
   else:
      print(i,len(Feasible_arc_final[i]))


#print(Feasible_arc_C)
#expand dctionary for the dummy vertices
keys=Feasible_arc.keys()
for key in keys:
   if key!=0:
      #print(key)
      for dummy in dct[key]:
         #print(dummy)
         if key in Feasible_arc_final[key]:
            Feasible_arc_final[dummy]=deepcopy(Feasible_arc_final[key])
            Feasible_arc_final[dummy].remove(key)
            Feasible_arc_final[dummy].append(dummy)
         else:
            Feasible_arc_final[dummy]=deepcopy(Feasible_arc_final[key])

keys=Feasible_arc_final.keys()
for key in keys:
   Feasible_arc_final[key].remove(0)

if last_node in Feasible_arc_final[0]:
   Feasible_arc_final[0].remove(last_node)

#model
```

```
from docplex.mp.model import Model
#from docplex.mp.sdetails import
m = Model(name='Charging station location problem cs objective r201C100')
#m.clear()
m.set_time_limit(7200)

########################################

Xij={}
for i in V_0:
   for j in V_n1:
      Xij[i,j]=m.binary_var(name=f'Xij({i},{j})')

########################################

qi={}
for i in V_0_n1:
   qi[i]=m.continuous_var(0,Q,name=f'qi({i})')
########################################

wi={}
for i in V:
   wi[i]=m.continuous_var(0,Q,name=f'wi({i})')

########################################
MinReadyTime=10000
for i in ReadyTime:
   if MinReadyTime>=i:
      MinReadyTime=i
#print(MinReadyTime)
MaxDueDate=0
for i in DueDate:
   if MaxDueDate<=i:
      MaxDueDate=i
#print(MaxDueDate)

ti={}

for i in V_0_n1:
   #node_i=get_node(i)
   ti[i]=m.continuous_var(MinReadyTime,MaxDueDate,name=f'ti({i})')

########################################

Yi={}
for i in V:
   Yi[i]=m.binary_var(name=f'Yi({i})')
```

```
########################################

fi={}
for i in V_0_n1:
   fi[i]=m.integer_var(0,F,name=f'fi({i})')




#####################################
#1st set of constraints : each customer is visited exactly once
for i in C:
   expr=0
   for j in Feasible_arc_final[i]:
      if i!=j:
         expr+=Xij[i,j]
   m.add_constraint(expr==1,f'(1) Customer(i{i}) is visited exactly once')

#2nd set of constraints : the same as 1st but relaxed for recharging visits
for i in V_C:
   expr=0
   for j in Feasible_arc_final[i]:
      if i!=j:
         expr+=Xij[i,j]
   m.add_constraint(expr<=Yi[i],f'(2) Recharge point(i{i}) is visited')

#3 set of constraints : Flow conservation
for j in V:
   expr=0

   for i in Feasible_arc_final[j]:
      if i!=j:
         expr+=Xij[j,i]

   for k in V_0:
      if k!=j:
         if j in Feasible_arc_final[k]:
            expr-=Xij[k,j]
   m.add_constraint(expr==0,f'(3) Flow conservation throught (j{j}) vertex ')


#4th set of constraints : Time flow conservation is enforced for customers

for i in C_0:
   #node1=get_node(i)
   for j in Feasible_arc_final[i]:
```

```
        node2=get_node(j)
        if i!=j:
            expr=ti[i]+((Tij[i,node2]+ServiceTime[i])*Xij[i,j])-(DueDate[0]*(1-Xij[i,j]))-ti[j]
            m.add_constraint(0>=expr,f'(4) Time conservation for customer (i{i}) service')


#5th set of constraints : Time flow conservation is enforced for/through recharging stations

for i in V:
    node1=get_node(i)
    for j in Feasible_arc_final[i]:
        node2=get_node(j)
        if i!=j:
            #m.add_constraint(ti[j]>=ti[i]+Tij[node1,node2]*Xij[i,j]-DueDate[0]*(1-
Xij[i,j]),f'(5) Time conservation for recharging stops (i{i})')
            m.add_constraint(ti[j]>=ti[i]+Tij[node1,node2]*Xij[i,j]+r*wi[i]-
(DueDate[0]+r*Q)*(1-Xij[i,j]),f'(5) Time conservation for recharging stops (i{i})')



#6th set of constraints : Arrival time between earliest and latest time for service
for i in V_0_n1:
    node_i=get_node(i)
    m.add_constraint(ReadyTime[node_i]<=ti[i],f'(6left) Earliest time of arrival at
vertex (i{i})')
    m.add_constraint(ti[i]<=DueDate[node_i],f'(6Right) latest time of arrival at vertex
(i{i})')


#7th set of constraints : Maximum Recharge amound of recharging station

for i in V:
    m.add_constraint(wi[i]<=Q*Yi[i],f'(7) Recharge amound at station (i{i})')




#8th set of constraints : A charging station has to be build at vertex i if recharging takes place at vertex i

for i in S:
    m.add_constraint(Yi[i]<=wi[i],f'(8) CS (i{i}) built if recharge takes place')


#9th set of constraints : Maximum number of charging station that can be located(OUB)
```

```
#Minimum number of CS that have to be located
# OUB has to be predetermined
sumYi=0
for i in V_S:
   sumYi+=Yi[i]
m.add_constraint(sumYi<=OUB)
m.add_constraint(sumYi>=OLB)



#10th set of constraints : mirror siting decisions from real vertices to dummy vertices
for i in V_S:
   for j in Si[i]:
      m.add_constraint(Yi[i]>=Yi[j],f'(10) Real CS (i{i}) mirror to Dummy vertex (j{j}) ')


#11th set of constraints : Freight flow conservation
for i in V_0:
   for j in Feasible_arc_final[i]:
      expr=0
      if i!=j:
         if i in C:
            pi=demand[i]
         else:
            pi=0
         expr=fi[i]-pi*Xij[i,j]+F*(1-Xij[i,j])
         m.add_constraint(fi[j]<=expr,f'(11) Freight Flow Conservation between
(i{i},j{j})')
         #m.add_constraint(0<=fi[j],f'(11b) Non negative freight load at (j{j})')
         # probably this is not needed, when i create the variable fi



#13th set of constraints : Battery flow conservation

for j in V_n1:
   node=get_node(j)
   expr=qi[0]-O*matrix[0,node]*Xij[0,j]+Q*(1-Xij[0,j])
   m.add_constraint(qi[j]<=expr,f'(13) Battery flow conservation from Depot/start to
vertex(j{j})')



#14th set of constraints : Battery Flow conservation through customers or CS
for i in V:
   node1=get_node(i)
```

```
   for j in Feasible_arc_final[i]:
      node2=get_node(j)
      expr=0
      if i!=j:
         #expr=qi[i]-O*matrix[node1,node2]*Xij[i,j]+Q*(1-Xij[i,j])
         expr=qi[i]+wi[i]-O*matrix[node1,node2]*Xij[i,j]+Q*(1-Xij[i,j])-qi[j]
         m.add_constraint(0<=expr,f'(14) Battery flow conservation from vertex to
vertex(i{i},j{j})')
         #m.add_constraint(0<=qi[j],f'(14b) Non negative Battery load at (j{j})')


#15th set of constraints : Battery load and recharging amound less than Battery
capacity

for i in V:
   m.add_constraint(qi[i]+wi[i]<=Q,f'(15)  Battery load and recharging amound less
than Q at (i{i})')
   #m.add_constraint(0<=qi[i]+wi[i],f'(15)  Battery load and recharging amound non
negative at (i{i})')


#16th constraint total vehicles used
#minimum vehicles to be needed
vehicles=0
for j in V_n1:
   vehicles+=Xij[0,j]
m.add_constraint(vehicles<=NUB,'(16) Maximum number of vehicles')
m.add_constraint(vehicles>=NLB,'(16) minimum number of vehicles')

#m.add_constraint(qi[lastnode]>=0.1*Q)

##########################################

Objective=0
for i in V_0:
   node1=get_node(i)
   for j in V_n1:
      node2=get_node(j)
      Objective+=matrix[node1,node2]* Xij[i,j]

#for i in V_S:
#    Objective+=Yi[i]

m.minimize(Objective)


m.print_information()
```

```
m.export_as_lp('test rc105C5.lp')

# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
if sol:
   print('Solution found')
   #m.print_solution()
#    status = m.solution.get_status()
#    print ("Solution status = " , status, ":")
#    print (m.solution.status[status])
#    print ("Objective value = " , m.solution.get_objective_value())
#    m.print_solution()


else:
   print("No solution found")


print(m.solve_details)



active_arcs=[]

for i in V_0:
   for j in V_n1:
      if i!=j:
         value=Xij[i,j].solution_value
         if value>=0.99:
            active_arcs.append((i,j))
#active_arcs

vehicle_used=[]
number_of_routes=0 # Έυρεση αριθμού διαδρομών.
for i in range(len(active_arcs)):
   if active_arcs[i][0]==0:
      number_of_routes=number_of_routes+1
      vehicle=active_arcs[i]
      vehicle_used.append(vehicle)
number_of_routes

routes=[]
for i in range(len(active_arcs)):
   if active_arcs[i][0]==0:
      routes_i=[active_arcs[i][0],active_arcs[i][1]]
      routes.append(routes_i)
```

```
#routes

#Fixed
#print(routes)
number_of_ends=0
last_node=V_n1[-1]
while number_of_ends<number_of_routes:
   for i in range(len(active_arcs)-number_of_routes):
      for j in range(len(routes)):
         #Sorting process for each route
         if active_arcs[:-number_of_routes][i][0]==routes[j][len(routes[j])-1]:
            #creates the sorting of the arcs
            routes[j].append(active_arcs[:-number_of_routes][i][1])
            if active_arcs[:-number_of_routes][i][1]==last_node:
               # change the route
               number_of_ends=number_of_ends+1

print(active_arcs)
print(routes)


routesprinter=[]
bat=[]
cargo=[]
times=[]
for i in range(len(routes)):
   print('ROUTE '+str(i+1))
  # print('-------')
  # print('-------')
   route_order=''
   time_line=''
   battery_line=''
   cargo_line=''
   for j in range(len(routes[i])-1):
      route_order=route_order+str(routes[i][j])+'-->'

      nodeN=routes[i][j]
      #print(nodeN)
      timeN=round(ti[nodeN].solution_value,4)
      batteryN=round(qi[nodeN].solution_value,4)
      cargoN=round(fi[nodeN].solution_value,4)
      time_line=time_line+str(timeN)+'-->'
      battery_line=battery_line+str(batteryN)+'-->'
      cargo_line=cargo_line+str(cargoN)+'-->'

   time_of_return=ti[last_node].solution_value
   battery_at_depot=qi[last_node].solution_value
```

```
    route_order=route_order+str(0)
    time_line=time_line+'return at depot'
    battery_line=battery_line+str(battery_at_depot)
    cargo_line=cargo_line+'return at depot'
    print('Route Order: '+route_order)
    print('Time Line: '+ time_line)
    print('battery line ' + battery_line)
    print('cargo line:' + cargo_line)
    routesprinter.append(route_order)
    bat.append(battery_line)
    cargo.append(cargo_line)
    times.append(time_line)

distance_traveled=0
dist_route=[]
for route in routes:
    dist_route_i=0
    for node in range(len(route)-1):
        #print(route[node],route[node+1])
        dist_route_i+=matrix[get_node(route[node]),get_node(route[node+1])]
    dist_route.append(dist_route_i)
    distance_traveled+=dist_route_i
print(dist_route)
print('distance_traveled =',distance_traveled)

details=m.solve_details
with open('LRP solution storage r201C100 cluster3.txt','w') as f:
    obj=m.objective_value
    f.write(str(details))
    f.write('Objective = ')
    f.write(str(obj))
    f.write('\n')
    f.write('Solution values for Xij variables')
    f.write('\n')
    for i in V_0:
        for j in V_n1:
            value=Xij[i,j].solution_value
            if value>=0.99:
                variable=(f'Xij[{i},{j}]')+'='+str(value)
                f.write(variable)
                f.write('\n')
    f.write('Solution values for qi variables')
    f.write('\n')
    for i in V_0_n1:
        value= qi[i].solution_value
        if value>0:
            variable=(f'qi[{i}]')+'='+str(value)
```

```
        f.write(variable)
        f.write('\n')
f.write('Solution values for wi variables')
f.write('\n')
for i in V:
    value=wi[i].solution_value
    if value>0:
        variable=(f'wi[{i}]')+'='+str(value)
        f.write(variable)
        f.write('\n')
f.write('Solution values for ti variables')
f.write('\n')
for i in V_0_n1:
    value=ti[i].solution_value
    if value>=0:
        if i!=0 and i!=lastnode:
            for j in V_n1:
                xij_val=Xij[i,j].solution_value
                if xij_val>=0.99:
                    variable=(f'ti[{i}]')+'='+str(value)
                    f.write(variable)
                    f.write('\n')
        else:
            variable=(f'ti[{i}]')+'='+str(value)
            f.write(variable)
            f.write('\n')
f.write('Solution values for Yi variables')
f.write('\n')
for i in V:
    value=Yi[i].solution_value
    if value>=0.99:
        variable=(f'Yi[{i}]')+'='+str(value)
        f.write(variable)
        f.write('\n')
f.write('Solution values for fi variables')
f.write('\n')
for i in V_0_n1:
    value=fi[i].solution_value
    if value>0:
        variable=(f'fi[{i}]')+'='+str(value)
        f.write(variable)
        f.write('\n')
#include dct at txt solution file
keys=dct.keys()
for key in keys:
    line=dct[key]
    line=str(key)+':'+str(line)
```

```
            f.write(line)
            f.write('\n')
        #include feasible arc at txt solution file
        keys=Feasible_arc_final.keys()
        for key in keys:
            line=Feasible_arc_final[key]
            line=str(key)+':'+str(line)
            f.write(line)
            f.write('\n')
        #include routing solution to txt solution file

        for i in range(len(routes)):
            f.write('ROUTE '+str(i+1))
            f.write('\n')
            f.write('Route Order: '+routesprinter[i])
            f.write('\n')
            f.write('Time Line: '+ times[i])
            f.write('\n')
            f.write('battery line ' + bat[i])
            f.write('\n')
            f.write('cargo line:' + cargo[i])
            f.write('\n')

    print('solution txt created')
```

Lower Bounds

```
from docplex.mp.model import Model
#
#from docplex.mp.sdetails import
#
m = Model(name='lower bound for number of vehicles')
m.set_time_limit(100)


zi={}
for i in C:
   zi[i]=m.binary_var(name=f'zi({i})')

hij={}
for i in Z:
   for j in C:
      hij[i,j]=m.binary_var(name=f'hij({i},{j})')

#S.T.
for i in Z:
   expr=0
   for j in C:
      expr+=hij[i,j]*demand[j]
   m.add_constraint(expr<=F*zi[i],f'(1) freight capacity not exeeded (i{i}) vehicle')


for j in C:
   expr=0
   for i in Z:
      expr+=hij[i,j]
   m.add_constraint(expr==1,f'(1) each customer is visited once (i{i}) customer')


objective=0
for i in Z:
   objective+=zi[i]

m.minimize(objective)


m.print_information()
m.export_as_lp('binbag.lp')


# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
```

```
if sol:
   print('Solution found')
   m.print_solution()

else:
   print("No solution found")


print(m.solve_details)


J=[]
for j in C:
   if O*2*matrix[0,j]>=Q:
      J.append(j)
      print(O*matrix[0,j])

V_S_0=V_S.copy()
V_S_0.append(0)
print(V_S,V_S_0,J)

#
m = Model(name='lower bound for number of stations')
#
m.set_time_limit(100)

#V_S_0=V_S.copy()
#V_S_0.append(0)


ai={}
for i in V_S_0:
   ai[i]=m.binary_var(name=f'ai({i})')

bij={}
for i in V_S_0:
   for j in V_S_0:
      bij[i,j]=m.binary_var(name=f'bij({i},{j})')

kij={}
for i in V_S_0:
   for j in V_S_0:
      kij[i,j]=m.binary_var(name=f'kij({i},{j})')


for j in J:
   expr=0
```

```
    for i in V_S:
        expr+=bij[i,j]
    m.add_constraint(expr>=1,f'each customer is covered by at least one station j
({i,j})')


for j in J:
    for i in V_S_0:
        expr=bij[i,j]-ai[i]
        m.add_constraint(expr<=0,f'relationship between charger i customer j ({i,j})')

for j in J:
    for i in V_S_0:
        expr=bij[i,j]*O*matrix[i,j]-Q*ai[i]
        m.add_constraint(expr<=0,f'')

for j in J:
    for i in V_S_0:
        expr=kij[i,j]*O*matrix[i,j]-Q
        m.add_constraint(expr<=0,f'')


for i in V_S_0:
    for j in V_S_0:
        expr=2*kij[i,j]-ai[i]-ai[j]
        m.add_constraint(expr<=0,f'')


ai[0]=1


for i in V_S_0:
    expr=0
    for j in V_S_0:
        expr-=kij[i,j]
    expr=expr+ai[i]
    m.add_constraint(expr<=0,f'')


objective=0
for i in V_S:
    objective+=ai[i]

m.minimize(objective)


m.print_information()
```

```
m.export_as_lp('stationsLB.lp')


# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
if sol:
   print('Solution found')
   m.print_solution()

else:
   print("No solution found")


print(m.solve_details)
```

VRP

```
from docplex.mp.model import Model
#from docplex.mp.sdetails import
m = Model(name='Charging station location problem cs objective rc105c5')
#m.clear()
m.set_time_limit(500)

##########################################

Xij={}
for i in C_0:
    for j in C_n1:
        Xij[i,j]=m.binary_var(name=f'Xij({i},{j})')

##########################################


MinReadyTime=10000
for i in ReadyTime:
    if MinReadyTime>=i:
        MinReadyTime=i
#print(MinReadyTime)
MaxDueDate=0
for i in DueDate:
    if MaxDueDate<=i:
        MaxDueDate=i
#print(MaxDueDate)
```

```
#infeasible due to distance
#Feasible_arc={}

#for i in C_0:
#    for j in C_0_n1:
#        if O*matrix[i,j]<=Q:
#            if j in Feasible_arc[i]:
#                continue
#            else:
#                Feasible_arc[i].append(j)




ti={}
for i in C_0_n1:
    #node_i=get_node(i)
    ti[i]=m.continuous_var(MinReadyTime,MaxDueDate,name=f'ti({i})')

fi={}
for i in C_0_n1:
    fi[i]=m.integer_var(0,F,name=f'fi({i})')

#1st set of constraints : each customer is visited exactly once
for i in C:
    expr=0
    for j in Feasible_arc[i]:
        if i!=j:
            expr+=Xij[i,j]
```

```
    m.add_constraint(expr==1,f'(1) Customer(i{i}) is visited exactly once')


#2 set of constraints : Flow conservation
for j in C:
    expr=0

    for i in Feasible_arc[j]:
        if i!=j:
            expr+=Xij[j,i]

    for k in C_0:
        if k!=j:
            if j in Feasible_arc[k]:
                expr-=Xij[k,j]
    m.add_constraint(expr==0,f'(3) Flow conservation throught (j{j}) vertex ')



#4th set of constraints : Time flow conservation is enforced for customers

for i in C_0:
    #node1=get_node(i)
    for j in Feasible_arc[i]:

        if i!=j:
            expr=ti[i]+((Tij[i,j]+ServiceTime[i])*Xij[i,j])-(DueDate[0]*(1-Xij[i,j]))-ti[j]
            m.add_constraint(0>=expr,f'(4) Time conservation for customer (i{i}) service')
```

```
#6th set of constraints : Arrival time between earliest and latest time for service
for i in C_0_n1:

   m.add_constraint(ReadyTime[i]<=ti[i],f'(6left) Earliest time of arrival at vertex (i{i})')
   m.add_constraint(ti[i]<=DueDate[i],f'(6Right) latest time of arrival at vertex (i{i})')

#11th set of constraints : Freight flow conservation
for i in C_0:
   for j in Feasible_arc[i]:
      expr=0
      if i!=j:
         if i in C:
            pi=demand[i]
         else:
            pi=0
         expr=fi[i]-pi*Xij[i,j]+F*(1-Xij[i,j])
         m.add_constraint(fi[j]<=expr,f'(11) Freight Flow Conservation between (i{i},j{j})')
         #m.add_constraint(0<=fi[j],f'(11b) Non negative freight load at (j{j})')
         # probably this is not needed, when i create the variable fi

sum_2=0
for i in C_0:
   sum_2+=Xij[i,last_node]

sum_0=0
for j in C_n1:
      sum_0+=Xij[0,j]
m.add_constraint(sum_0<=NUB)
#m.add_constraint(sum_0==sum_2)
```

```
Objective=0
for i in C_0:

   for j in C_n1:

      Objective+=matrix[i,j]* Xij[i,j]

#for i in V_S:
#    Objective+=Yi[i]

m.minimize(Objective)



m.print_information()
m.export_as_lp('test c10 cs3 v1 objective.lp')

# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
if sol:
   print('Solution found')
   #m.print_solution()
#    status = m.solution.get_status()
#    print ("Solution status = " , status, ":")
#    print (m.solution.status[status])
#    print ("Objective value = " , m.solution.get_objective_value())
   m.print_solution()
```

```
else:
    print("No solution found")



print(m.solve_details)

active_arcs=[]

for i in C_0:
    for j in C_n1:
        if i!=j:
            value=Xij[i,j].solution_value
            if value>=0.99:
                active_arcs.append((i,j))
#active_arcs

vehicle_used=[]
number_of_routes=0 # Έυρεση αριθμού διαδρομών.
for i in range(len(active_arcs)):
    if active_arcs[i][0]==0:
        number_of_routes=number_of_routes+1
        vehicle=active_arcs[i]
        vehicle_used.append(vehicle)
number_of_routes

routes=[]
for i in range(len(active_arcs)):
```

```
    if active_arcs[i][0]==0:
        routes_i=[active_arcs[i][0],active_arcs[i][1]]
        routes.append(routes_i)
#routes

#Fixed
#print(routes)
number_of_ends=0
#last_node=C_n1[-1]
while number_of_ends<number_of_routes:
    for i in range(len(active_arcs)-number_of_routes):
        for j in range(len(routes)):
            #Sorting process for each route
            if active_arcs[:-number_of_routes][i][0]==routes[j][len(routes[j])-1]:
                #creates the sorting of the arcs
                routes[j].append(active_arcs[:-number_of_routes][i][1])
                if active_arcs[:-number_of_routes][i][1]==last_node:
                    # change the route
                    number_of_ends=number_of_ends+1

print(active_arcs)
print(routes)



routesprinter=[]
bat=[]
cargo=[]
times=[]
for i in range(len(routes)):
```

```
 print('ROUTE '+str(i+1))
# print('-------')
# print('-------')
 route_order=''
 time_line=''
 #battery_line=''
 cargo_line=''
 for j in range(len(routes[i])-1):
    route_order=route_order+str(routes[i][j])+'-->'

    nodeN=routes[i][j]
    #print(nodeN)
    timeN=round(ti[nodeN].solution_value,4)
    #batteryN=round(qi[nodeN].solution_value,4)
    cargoN=round(fi[nodeN].solution_value,4)
    time_line=time_line+str(timeN)+'-->'
    #battery_line=battery_line+str(batteryN)+'-->'
    cargo_line=cargo_line+str(cargoN)+'-->'

time_of_return=ti[last_node].solution_value
#battery_at_depot=qi[last_node].solution_value
route_order=route_order+str(0)
time_line=time_line+'return at depot'
#battery_line=battery_line+str(battery_at_depot)
cargo_line=cargo_line+'return at depot'
print('Route Order: '+route_order)
print('Time Line: '+ time_line)
#print('battery line ' + battery_line)
print('cargo line:' + cargo_line)
```

```
routesprinter.append(route_order)
#bat.append(battery_line)
cargo.append(cargo_line)
times.append(time_line)
```

FCM

```
#from scipy.spatial import distance

import pandas as pd

#Διαβάζω το αρχείο με τους κόμβους από πάνω οι πελάτες και κάτω οι σταθμοί, σωστή διαμόρφωση αρχείου

column_names = ["StringID", "Type", "x", "y", "demand", "ReadyTime", "DueDate", "ServiceTime"]

df = pd.read_csv("r201C100.csv", names=column_names,skiprows=1)

#print(df)


xloc=df.x.to_list()

yloc=df.y.to_list()

StringID=df.StringID.to_list()

Type=df.Type.to_list()

demand=df.demand.to_list()

ReadyTime=df.ReadyTime.to_list()

DueDate=df.DueDate.to_list()

ServiceTime=df.ServiceTime.to_list()


for i in range(len(xloc)):
   xloc[i]=int(xloc[i])
   yloc[i]=int(yloc[i])


read_c=[]

read_cs=[]

read_depot=[]

for i in range(len(StringID)):
   if Type[i]=='d':
      read_depot.append(i)
   elif Type[i]=='c':
```

```
        read_c.append(i)
    elif StringID[i]=='S0':
        continue
        #break
    elif Type[i]=='f':
        read_cs.append(i)

from docplex.mp.model import Model
#from docplex.mp.sdetails import
m = Model(name='Charging station location problem cs objective r102C10')
#m.clear()
m.set_time_limit(7200)


read_c.append(0)
N=read_c

Q=[[0, 1, 70, 30, 32, 66, 9, 35, 71, 20, 51, 3, 29, 24, 12, 0],
  [0, 2, 42, 100, 14, 43, 57, 87, 97, 94, 6, 0],
  [0, 7, 46, 8, 18, 83, 60, 89, 0],
  [0, 13, 95, 59, 99, 5, 84, 61, 92, 37, 98, 93, 85, 91, 16, 44, 38, 15, 41, 73, 0],
  [0, 26, 21, 72, 74, 75, 22, 56, 4, 40, 0],
  [0, 27, 52, 88, 19, 48, 47, 82, 45, 17, 86, 96, 0],
  [0, 77, 79, 81, 65, 34, 78, 33, 76, 50, 69, 31, 10, 90, 63, 11, 64, 49, 36, 62, 0],
  [0, 80, 54, 25, 67, 23, 39, 55, 68, 28, 53, 58, 0]]

Nq=[[ 1, 70, 30, 32, 66, 9, 35, 71, 20, 51, 3, 29, 24, 12],
  [ 2, 42, 100, 14, 43, 57, 87, 97, 94, 6],
  [ 7, 46, 8, 18, 83, 60, 89],
```

```
  [ 13, 95, 59, 99, 5, 84, 61, 92, 37, 98, 93, 85, 91, 16, 44, 38, 15, 41, 73],
  [26, 21, 72, 74, 75, 22, 56, 4, 40],
  [ 27, 52, 88, 19, 48, 47, 82, 45, 17, 86, 96],
  [ 77, 79, 81, 65, 34, 78, 33, 76, 50, 69, 31, 10, 90, 63, 11, 64, 49, 36, 62],
  [ 80, 54, 25, 67, 23, 39, 55, 68, 28, 53, 58]]

fq=[1,2,3,4,5,6,7,8]

P=8



Xk={}
for i in N:
   Xk[i]=m.binary_var(name=f'Xk({i})')

Yq={}
for i in range(len(Q)):
   Yq[i]=m.binary_var(name=f'Yq({i})')



for q in range(len(Q)):
   sum_k=0
   for k in Nq[q]:
      sum_k+=Xk[k]
   m.add_constraint(sum_k>=Yq[q],f' Route (q{q}) is covered')

sum_stations=0
for k in N:
```

```
    sum_stations+=Xk[k]
m.add_constraint(sum_stations==P,f'Total stations built')


Objective=0
for q in range(len(Q)):
    Objective+=fq[q]*Yq[q]


m.maximize(Objective)

m.print_information()
m.export_as_lp('r102C10 flow base.lp')

# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
if sol:
    print('Solution found')
    #m.print_solution()
#    status = m.solution.get_status()
#    print ("Solution status = " , status, ":")
#    print (m.solution.status[status])
#    print ("Objective value = " , m.solution.get_objective_value())
    m.print_solution()


else:
    print("No solution found")
```

```
print(m.solve_details)



details=m.solve_details


with open('solution storage r104c5 Flow capturing.txt','w') as f:
    f.write(str(details))
    obj=m.objective_value
    f.write('Objective = ')
    f.write(str(obj))
    f.write('\n')
    for i in N:
        value=Xk[i].solution_value
        if value>=0.99:
            variable=(f'Xi[{i}]')+'='+str(value)
            f.write(variable)
            f.write('\n')
    for i in range(len(Q)):
        value=Yq[i].solution_value
        if value>=0.99:
            variable=(f'Yq[{i}]')+'='+str(value)
            f.write(variable)
            f.write('\n')
print('solution txt created')
```

```
last_customer=len(read_c)+1
import matplotlib.pyplot as plt

routes_test=Q
x=[]
y=[]
#i=0
data=[]
plt.figure()
plt.axis([0,80,0,80])



colors=[]

import random
for route in routes_test:

   x_route=[]
   y_route=[]
   for stop in route:
      x_route.append(xloc[stop])
      y_route.append(yloc[stop])
   print('route=',route)
   y=y_route
   x=x_route
   print(x)
```

```
    print(y)
    dx=[]
    dy=[]
    for i in range(len(x)-1):
        dx.append(-(x[i]-x[i+1]))
        dy.append(-(y[i]-y[i+1]))
    print('dx=',dx)
    print('dy=',dy)
    r = random.randint(0,255)
    g = random.randint(0,255)
    b = random.randint(0,255)
    rgb = (r,g,b)
    colors.append(rgb)
    plt.quiver(x[:-1],y[:-
1],dx,dy,units='xy',angles='xy',scale=1,scale_units='xy',width=0.3,color= [rgb])
```

```
plt.scatter(xloc[1:last_customer], yloc[1:last_customer], color ='black' , marker="o",
s=10)
from matplotlib.lines import Line2D

for i in N:
    value=Xk[i].solution_value
    if value>=0.99:
        plt.scatter(xloc[i], yloc[i], color= "red",
                    marker= "^", s=10)

plt.scatter(xloc[0],yloc[0], color= "blue",marker= "^", s=20)
```

```
custom=[ Line2D([0], [0], marker='^', color='w',label='Depot',markerfacecolor='blue',markersize=10),
        Line2D([0], [0], marker='^', color='w',label='Charging Station',markerfacecolor='red',markersize=10),
        Line2D([0], [0], marker='o', color='w',label='Customer',markerfacecolor='black',markersize=10)]

plt.legend(handles=custom,loc='center left', bbox_to_anchor=(1, 0.5))



#for i in range(len(Q)):
#    value=Yq[i].solution_value
#    if value>=0.99:
#        plt.scatter(xloc[i], yloc[i], color= "green",
#                    marker= "^", s=10)

# x-axis label
plt.xlabel('x - axis')
# frequency label
plt.ylabel('y - axis')
# plot title
#plt.title('My plot!')
# showing legend
#plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig('r201C100_flow.png', format = "png", dpi = 150, bbox_inches = "tight")

# function to show the plot
plt.show()
```

```
kk=0
for route in routes_test:

    x_route=[]
    y_route=[]
    for stop in route:
        x_route.append(xloc[stop])
        y_route.append(yloc[stop])

    y=y_route
    x=x_route

    dx=[]
    dy=[]
    for i in range(len(x)-1):
        dx.append(-(x[i]-x[i+1]))
        dy.append(-(y[i]-y[i+1]))
    col=colors[kk]
    plt.scatter(xloc[1:last_customer], yloc[1:last_customer], color ='black' , marker="o", s=10)

    for i in N:
        value=Xk[i].solution_value
        if value>=0.99:
            plt.scatter(xloc[i], yloc[i], color= "red",
                        marker= "^", s=10)

    plt.scatter(xloc[0],yloc[0], color= "blue",marker= "^", s=20)
    plt.quiver(x[:-1],y[:-1],dx,dy,units='xy',angles='xy',scale=1,scale_units='xy',width=0.3,color= [col])
```

```
    plt.legend(handles=custom,loc='center left', bbox_to_anchor=(1, 0.5))

    kk=kk+1

    plt.savefig(f'r201C100_flow {kk}.png', format = "png", dpi = 150, bbox_inches = "tight")

    plt.show()
```

Clusters

```
import numpy as np

#from scipy.spatial import distance

import pandas as pd


#Διαβάζω το αρχείο με τους κόμβους από πάνω οι πελάτες και κάτω οι σταθμοί, σωστή διαμόρφωση αρχείου

column_names = ["StringID", "Type", "x", "y", "demand", "ReadyTime", "DueDate", "ServiceTime"]

df = pd.read_csv("r201C100.csv", names=column_names,skiprows=1)

#print(df)


xloc=df.x.to_list()

yloc=df.y.to_list()

StringID=df.StringID.to_list()

Type=df.Type.to_list()

demand=df.demand.to_list()

ReadyTime=df.ReadyTime.to_list()

DueDate=df.DueDate.to_list()

ServiceTime=df.ServiceTime.to_list()


for i in range(len(xloc)):

   xloc[i]=int(xloc[i])

   yloc[i]=int(yloc[i])

   demand[i]=float(demand[i])

   ReadyTime[i]=float(ReadyTime[i])

   DueDate[i]=float(DueDate[i])

   ServiceTime[i]=float(ServiceTime[i])


nodeSet=[]
```

```
for i in range(len(StringID)):
   nodeloc=[xloc[i],yloc[i]]
   nodeSet.append(nodeloc)
#nodeSet

df = pd.DataFrame(nodeSet, columns=['xcord', 'ycord'], index=StringID)
print(df)
n_df=(df.values)
#print(n_df)

#print((df.values).shape)

matrix=np.zeros(((df.values).shape[0],(df.values).shape[0]))
#print(matrix)



for i in range((df.values).shape[0]):
   for j in range((df.values).shape[0]):
      matrix[i,j]=round(np.sqrt(np.sum((n_df[i]-n_df[j])**2)),2)



#δημιουργία πίνακα χρόνων, ορισμός ταχύτητας πρώτα
velocity=1
Tij=matrix.copy()
Tij=(velocity**(-1))*Tij
#print(Tij)

import matplotlib.pyplot as plt
xdep=xloc[0]
```

```
ydep=yloc[0]
# x-axis values
x = xloc[1:101]
# y-axis values
y = yloc[1:101]
plot_demand=demand[1:101]
x1=xloc[102:]
y1=yloc[102:]
# plotting points as a scatter plot
plt.scatter(x, y, label= "customers", color= "green",
        marker= "^", s=10)
plt.scatter(x1, y1, label= "Charging stations", color= "red", s=10, marker='o')
plt.scatter(xdep,ydep,label= "depot", color= "yellow",
        marker= "*", s=10)
# x-axis label
plt.xlabel('x - axis')
# frequency label
plt.ylabel('y - axis')
# plot title
#plt.title('My scatter plot!')
# showing legend
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig('ScatterPlot_all_nodes.png')
# function to show the plot
plt.show()

def bubbleSort(arr,StringID):
   n = len(arr)
```

```
    # Traverse through all array elements
    for i in range(n):

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
                SID[j],SID[j+1] =SID[j+1],SID[j]



# Driver code to test above
arr = matrix[0][:100].copy()
SID=StringID[:100].copy()
bubbleSort(arr,SID)

#print ("Sorted array is:")
#for i in range(len(arr)):
#    indexx=StringID.index(SID[i])
#    print (arr[i], SID[i],indexx)

#for i in SID:
#    indexx=StringID.index(i)
#    print(i,indexx)
```

```
customers=[]
for i in range(1,101):
   customers.append(i)
print(customers)

seeds_needed=10
seeds=[]
for i in range(seeds_needed):
   seeds.append(StringID.index(SID[-(i+1)]))
seeds

seeds=[38,43,23,67,24,34,65,32,64,46]
dik=[]
customers=[]
for i in range(1,101):
   customers.append(i)

C_0=customers.copy()
C_0.append(0)


for i in customers:
   dikk=[]
   for k in seeds:
      a=(matrix[0,i]+matrix[i,k]+matrix[k,0],matrix[0,k]+matrix[k,i]+matrix[i,0])
      b=min(a)
      b=b-(matrix[0,k]+matrix[k,0])
      dikk.append(round(b,9))
   dik.append(dikk)
```

```
from docplex.mp.model import Model
#from docplex.mp.sdetails import
m = Model(name='General assidnment')
#m.clear()
m.set_time_limit(300)



Yik={}
for i in C_0:
    for k in range(len(seeds)):
        Yik[i,k]=m.binary_var(name=f'Yik({i},{k})')



for k in range(len(seeds)):
    sum_i=0
    for i in C_0:
        sum_i+=demand[i]*Yik[i,k]
    m.add_constraint(sum_i<=800,f'(1) cap vehicle (k{k}) ')

for i in range(1,101):
    sum_y=0
    for k in range(len(seeds)):
        sum_y+=Yik[i,k]
    m.add_constraint(sum_y==1, f'(2) assignment customer (i{i})')

sum_y_0=0
for k in range(len(seeds)):
```

```
    sum_y_0+=Yik[0,k]
m.add_constraint(sum_y_0==10, f'(2) assignment customer (i{0})')


for k in range(len(seeds)):
    sum_k=0
    for i in range(1,101):
        sum_k+=Yik[i,k]
    m.add_constraint(sum_k<=15,f'(3) maximum modes at each cluster (k{k})')

Objective=0
for i in range(100):
    for k in range(len(seeds)):
        Objective+=dik[i][k]*Yik[i+1,k]




m.minimize(Objective)


m.print_information()
m.export_as_lp('General assignment vs1.lp')

# ΕΠΙΛΥΣΗ ΜΟΝΤΕΛΟΥ
print("\nSolving model....")
sol = m.solve()
if sol:
    print('Solution found')
```

```
    #m.print_solution()
#    status = m.solution.get_status()
#    print ("Solution status = " , status, ":")
#    print (m.solution.status[status])
#    print ("Objective value = " , m.solution.get_objective_value())
#    m.print_solution()


else:
    print("No solution found")


print(m.solve_details)


clusters=[]
for k in range(len(seeds)):
    print('customers serviced by vehicle ',k+1)
    cluster_k=[]
    for i in range(len(customers)+1):
    #for i in customers:
        value=Yik[i,k].solution_value
        if value>=0.99:
            cluster_k.append(i)
            print('Customer ',i,' served' )
    clusters.append(cluster_k)

xlocmax_i=[]
ylocmax_i=[]
```

```
xlocmin_i=[]
ylocmin_i=[]
for cluster in clusters:
    xlocmax=0
    ylocmax=0
    xlocmin=1000
    ylocmin=1000
    for customer in cluster:
        if xlocmax<=xloc[customer]:
            xlocmax=xloc[customer]
        if ylocmax<=yloc[customer]:
            ylocmax=yloc[customer]
        if xlocmin>=xloc[customer]:
            xlocmin=xloc[customer]
        if ylocmin>=yloc[customer]:
            ylocmin=yloc[customer]
    xlocmax_i.append(xlocmax)
    ylocmax_i.append(ylocmax)
    xlocmin_i.append(xlocmin)
    ylocmin_i.append(ylocmin)

print('xlocmax_i = ',xlocmax_i)
print('ylocmax_i = ',ylocmax_i)
print('xlocmin_i = ',xlocmin_i)
print('ylocmin_i = ',ylocmin_i)

xloc_clusters=[]
yloc_clusters=[]
k=0
```

```
labels=[]
for i in clusters:
    k+=1
    l='cluster '+str(k)
    labels.append(l)
    xloc_cluster=[]
    yloc_cluster=[]
    for j in i:
        #print(j)
        xloc_cluster.append(xloc[j])
        yloc_cluster.append(yloc[j])
    #print(xloc_cluster)
    xloc_clusters.append(xloc_cluster)
    yloc_clusters.append(yloc_cluster)


plt.scatter(xloc_clusters[0][1:], yloc_clusters[0][1:], label=labels[0], color ='red' , marker="o", s=30)
plt.scatter(xloc_clusters[1][1:], yloc_clusters[1][1:], label=labels[1], color = 'blue', marker="o", s=30)
plt.scatter(xloc_clusters[2][1:], yloc_clusters[2][1:], label=labels[2], color = 'green', marker="o", s=30)
plt.scatter(xloc_clusters[3][1:], yloc_clusters[3][1:], label=labels[3], color ='turquoise' , marker="o", s=30)
plt.scatter(xloc_clusters[4][1:], yloc_clusters[4][1:], label=labels[4], color ='yellow' , marker="o", s=30)
plt.scatter(xloc_clusters[5][1:], yloc_clusters[5][1:], label=labels[5], color ='pink' , marker="o", s=30)
plt.scatter(xloc_clusters[6][1:], yloc_clusters[6][1:], label=labels[6], color ='lime' , marker="o", s=30)
plt.scatter(xloc_clusters[7][1:], yloc_clusters[7][1:], label=labels[7], color ='coral' , marker="o", s=30)
```

```
plt.scatter(xloc_clusters[8][1:], yloc_clusters[8][1:], label=labels[8], color ='peru' , marker="o", s=30)

plt.scatter(xloc_clusters[9][1:], yloc_clusters[9][1:], label=labels[9], color ='darkorchid' , marker="o", s=30)

plt.scatter(xdep,ydep,label= "depot", color= "black",marker= "^", s=45)

plt.scatter(x1, y1, label= "Charging stations", color= "black", s=30, marker='*')

# x-axis label

plt.xlabel('x - axis')

# frequency label

plt.ylabel('y - axis')

# plot title

#plt.title('My scatter plot')

# showing legend

#ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.savefig('ScatterPlot_cluster_nodes.png')

# function to show the plot

plt.show()



seeds=[38,43,23,67,24,34,65,32,64,46]

xloc_seed=[]

yloc_seed=[]

for seed in seeds:

   xloc_seed.append(xloc[seed])

   yloc_seed.append(yloc[seed])

plt.scatter(xloc_seed, yloc_seed, label='seeds', color ='red' , marker="o", s=30)

plt.scatter(xdep,ydep,label= "depot", color= "black",marker= "*", s=35)

# x-axis label

plt.xlabel('x - axis')
```

```
# frequency label
plt.ylabel('y - axis')
# plot title
plt.title('Seed scatter plot')
# showing legend
#ax.legend(loc='center left', bbox_to_anchor=(1,0.5))
plt.legend(loc='center left', bbox_to_anchor=(1,0.5))
plt.savefig('ScatterPlot_seeds.png')
# function to show the plot
plt.show()



read_c=[]
read_cs=[]
read_depot=[]
for i in range(len(StringID)):
   if Type[i]=='d':
      read_depot.append(i)
   elif Type[i]=='c':
      read_c.append(i)
   elif StringID[i]=='S0':
      continue
      #break
   elif Type[i]=='f':
      read_cs.append(i)
print(read_c,'\n',read_depot,'\n',read_cs)
def
get_closest_customer_cluster(charging_station_iD,min_dist_to_CS,matrix_row_of_c
s,clusters):
   row=matrix_row_of_cs.tolist()
```

```
    if min_dist_to_CS in row:
        index=row.index(min_dist_to_CS)
    #print(minCS, index)
    for cluster in clusters:
        if index in cluster:
            clusterID=clusters.index(cluster)
            #print(cluster)
    #return(clusterID,CS_to_cluster)
    return(clusterID)

min_dist_cust_to_CS=[]
CS_at_cluster={}
for cs in read_cs:
    row=matrix[cs][:102]#!!!!!!!!!
    dist=min(row)
    min_dist_cust_to_CS.append(dist)
    CS_at_cluster[cs]=get_closest_customer_cluster(cs,dist,row,clusters)
print(min_dist_cust_to_CS)
print(CS_at_cluster)

CS_assign_keys=CS_at_cluster.keys()
from copy import deepcopy
clusters_with_CS=deepcopy(clusters)
CS_by_cluster={}
#CS_to_cluster.append(charging_station_iD)
for CS in CS_assign_keys:
    cl=CS_at_cluster[CS]
    clusters_with_CS[cl].append(CS)
```

```
for i in range(len(clusters)):
    CS_by_cluster[i]=[]


for cs in read_cs:
    for i in range(len(clusters)):
        if CS_at_cluster[cs]==i:
            CS_by_cluster[i].append(cs)


xloc_clusters_cs=[]
yloc_clusters_cs=[]
k=0
labels=[]
for i in clusters_with_CS:
    k+=1
    l='cluster '+str(k)
    labels.append(l)
    xloc_cluster_cs=[]
    yloc_cluster_cs=[]
    for j in i:
        #print(j)
        xloc_cluster_cs.append(xloc[j])
        yloc_cluster_cs.append(yloc[j])
    #print(xloc_cluster)
    xloc_clusters_cs.append(xloc_cluster_cs)
    yloc_clusters_cs.append(yloc_cluster_cs)


plt.scatter(xloc_clusters_cs[0][1:], yloc_clusters_cs[0][1:], label=labels[0], color ='red'
, marker="o", s=30)
plt.scatter(xloc_clusters_cs[1][1:], yloc_clusters_cs[1][1:], label=labels[1], color =
'blue', marker="o", s=30)
```

```
plt.scatter(xloc_clusters_cs[2][1:], yloc_clusters_cs[2][1:], label=labels[2], color = 'green', marker="o", s=30)

plt.scatter(xloc_clusters_cs[3][1:], yloc_clusters_cs[3][1:], label=labels[3], color ='turquoise' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[4][1:], yloc_clusters_cs[4][1:], label=labels[4], color ='yellow' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[5][1:], yloc_clusters_cs[5][1:], label=labels[5], color ='pink' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[6][1:], yloc_clusters_cs[6][1:], label=labels[6], color ='lime' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[7][1:], yloc_clusters_cs[7][1:], label=labels[7], color ='coral' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[8][1:], yloc_clusters_cs[8][1:], label=labels[8], color ='peru' , marker="o", s=30)

plt.scatter(xloc_clusters_cs[9][1:], yloc_clusters_cs[9][1:], label=labels[9], color ='darkorchid' , marker="o", s=30)

plt.scatter(xdep,ydep,label= "depot", color= "black",marker= "^", s=45)

plt.scatter(x1, y1, label= "Charging stations", color= "black", s=10, marker='*')

# x-axis label

plt.xlabel('x - axis')

# frequency label

plt.ylabel('y - axis')

# plot title

#plt.title('My scatter plot')

# showing legend

#ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.savefig('ScatterPlot_cluster_nodes.png')

# function to show the plot

plt.show()
```

Βιβλιογραφία

[1] K. A. M. Iyaguchi and S. H. D. Emura, “S Pecific F Actors T Hat I Nfluence D Eciding,” no. June, pp. 2516–2522, 2010.

[2] “Road transport: Reducing CO_2 emissions from vehicles.” https://ec.europa.eu/clima/eu-action/transport-emissions/road-transport-reducing-co2-emissions-vehicles_el (accessed Dec. 07, 2021).

[3] C. Kouridis and C. Vlachokostas, “Towards decarbonizing road transport: Environmental and social benefit of vehicle fleet electrification in urban areas of Greece,” *Renew. Sustain. Energy Rev.*, vol. 153, no. January 2021, p. 111775, 2022, doi: 10.1016/j.rser.2021.111775.

[4] S. Pelletier, O. Jabali, and G. Laporte, “Goods distribution with electric vehicles: Review and research perspectives,” *Transp. Sci.*, vol. 50, no. 1, pp. 3–22, 2016, doi: 10.1287/trsc.2015.0646.

[5] S. M. Mirhedayatian and S. Yan, “A framework to evaluate policy options for supporting electric vehicles in urban freight transport,” *Transp. Res. Part D Transp. Environ.*, vol. 58, no. November 2017, pp. 22–38, 2018, doi: 10.1016/j.trd.2017.11.007.

[6] B. Al-Hanahi, I. Ahmad, D. Habibi, and M. A. S. Masoum, “Charging Infrastructure for Commercial Electric Vehicles: Challenges and Future Works,” *IEEE Access*, vol. 9, pp. 121476–121492, 2021, doi: 10.1109/ACCESS.2021.3108817.

[7] “Share of urban population worldwide in 2021, by continent.” https://www.statista.com/statistics/270860/urbanization-by-continent/.

[8] “Icy Temperatures Cut Electric Vehicle Range Nearly in Half.” https://newsroom.aaa.com/2019/02/cold-weather-reduces-electric-vehicle-range/ (accessed Jan. 08, 2022).

[9] S. Iwan *et al.*, “Electric mobility in European urban freight and logistics - status and attempts of improvement,” *Transp. Res. Procedia*, vol. 39, no. 2018, pp. 112–123, 2019, doi: 10.1016/j.trpro.2019.06.013.

[10] “Plug-in electric vehicles in Germany.” https://en.wikipedia.org/wiki/Plug-in_electric_vehicles_in_Germany#:~:text=1.58%25 in 2017.-,2019,plug-in hybrids 1.3%25. (accessed Jan. 08, 2022).

[11] R. M. L.M. Lombrana, “Electric car charging stations are finally about to take off,” 2020. https://www.bloomberg.com (accessed Dec. 12, 2021).

[12] C. Karolemeas, S. Tsigdinos, P. G. Tzouras, A. Nikitas, and E. Bakogiannis, “Determining electric vehicle charging station location suitability: A qualitative study of greek stakeholders employing thematic analysis and analytical hierarchy process,” *Sustain.*, vol. 13, no. 4, pp. 1–21, 2021, doi: 10.3390/su13042298.

[13] “2020 Guide On How To Charge Your Electric Car With Charging Stations.”

https://chargehub.com/en/electric-car-charging-guide.html (accessed Dec. 13, 2021).

[14] M. Kchaou-Boujelben, “Charging station location problem: A comprehensive review on models and solution approaches,” *Transp. Res. Part C Emerg. Technol.*, vol. 132, no. November, p. 103376, 2021, doi: 10.1016/j.trc.2021.103376.

[15] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, “Facility location and supply chain management - A review,” *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 401–412, 2009, doi: 10.1016/j.ejor.2008.05.007.

[16] V. Verter, “Uncapacitated and Capacitated Facility Location Problems,” in *Foundations of Location Analysis*, H. A. Eiselt and V. Marianov, Eds. New York, NY: Springer US, 2011, pp. 25–37.

[17] T. D. Chen, K. M. Kockelman, and M. Khan, “Parking-Based Assignment Method for Seattle , Washington,” 2006, doi: 10.3141/2385-04.

[18] S. Li, Y. Huang, and S. J. Mason, “A multi-period optimization model for the deployment of public electric vehicle charging stations on network,” *Transp. Res. Part C Emerg. Technol.*, vol. 65, pp. 128–143, 2016, doi: 10.1016/j.trc.2016.01.008.

[19] M. F. Anjos, B. Gendron, and M. Joyce-Moniz, “Increasing electric vehicle adoption through the optimal deployment of fast-charging stations for local and long-distance travel,” *Eur. J. Oper. Res.*, vol. 285, no. 1, pp. 263–278, 2020, doi: 10.1016/j.ejor.2020.01.055.

[20] R. Church and C. ReVelle, “The maximal covering location problem,” *Pap. Reg. Sci. Assoc.*, vol. 32, no. 1, pp. 101–118, 1974, doi: 10.1007/BF01942293.

[21] D. A. Giménez-Gaydou, A. S. N. Ribeiro, J. Gutiérrez, and A. P. Antunes, “Optimal location of battery electric vehicle charging stations in urban areas: A new approach,” *Int. J. Sustain. Transp.*, vol. 10, no. 5, pp. 393–405, 2016, doi: 10.1080/15568318.2014.961620.

[22] Y. W. Wang and C. C. Lin, “Locating road-vehicle refueling stations,” *Transp. Res. Part E Logist. Transp. Rev.*, vol. 45, no. 5, pp. 821–829, 2009, doi: 10.1016/j.tre.2009.03.002.

[23] Y. W. Wang and C. C. Lin, “Locating multiple types of recharging stations for battery-powered electric vehicle transport,” *Transp. Res. Part E Logist. Transp. Rev.*, vol. 58, pp. 76–87, 2013, doi: 10.1016/j.tre.2013.07.003.

[24] X. Zhang, D. Rey, and S. T. Waller, “Multitype Recharge Facility Location for Electric Vehicles,” *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 11, pp. 943–965, 2018, doi: 10.1111/mice.12379.

[25] J. Ko, T. T. Gim, and R. Guensler, “Locating refuelling stations for alternative fuel vehicles : a review on models and applications,” *Transp. Rev.*, vol. 0, no. 0, pp. 1–20, 2017, doi: 10.1080/01441647.2016.1273274.

[26] M. Ghamami, A. Zockaie, and Y. M. Nie, “A general corridor model for designing plug-in electric vehicle charging infrastructure to support intercity travel,” *Transp. Res. Part C Emerg. Technol.*, vol. 68, pp. 389–402, 2016, doi: 10.1016/j.trc.2016.04.016.

[27] Y. He, Z. Song, and Z. Liu, “Fast-charging station deployment for battery electric bus systems considering electricity demand charges,” *Sustain. Cities Soc.*, vol. 48, no. February, 2019, doi: 10.1016/j.scs.2019.101530.

[28] X. Wang, M. Shahidehpour, C. Jiang, and Z. Li, “Coordinated Planning Strategy for Electric Vehicle Charging Stations and Coupled Traffic-Electric Networks,” *IEEE Trans. Power Syst.*, vol. 34, no. 1, pp. 268–279, 2019, doi: 10.1109/TPWRS.2018.2867176.

[29] F. He, Y. Yin, and J. Zhou, “Deploying public charging stations for electric vehicles on urban road networks,” *Transp. Res. Part C Emerg. Technol.*, vol. 60, pp. 227–240, 2015, doi: 10.1016/j.trc.2015.08.018.

[30] M.J.Hodgson, “A flow capturing location-allocation model,” *Geogr.Anal.*, 1990.

[31] M. Kuby and S. Lim, “The flow-refueling location problem for alternative-fuel vehicles,” *Socioecon. Plann. Sci.*, vol. 39, no. 2, pp. 125–145, 2005, doi: 10.1016/j.seps.2004.03.001.

[32] I. Capar, M. Kuby, V. J. Leon, and Y. J. Tsai, “An arc cover-path-cover formulation and strategic analysis of alternative-fuel station locations,” *Eur. J. Oper. Res.*, vol. 227, no. 1, pp. 142–151, 2013, doi: 10.1016/j.ejor.2012.11.033.

[33] S. A. MirHassani and R. Ebrazi, “A flexible reformulation of the refueling station location problem,” *Transp. Sci.*, vol. 47, no. 4, pp. 617–628, 2013, doi: 10.1287/trsc.1120.0430.

[34] R. Xie *et al.*, “Stations on Highways : A Data-driven Robust Optimization Approach,” *IEEE Trans. Transp. Electrif.*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TTE.2018.2849222.

[35] G. Nagy and S. Salhi, “Location-routing: Issues, models and methods,” *Eur. J. Oper. Res.*, vol. 177, no. 2, pp. 649–672, 2006, doi: 10.1016/j.ejor.2006.04.004.

[36] M. Schiffer and G. Walther, “The electric location routing problem with time windows and partial recharging,” *Eur. J. Oper. Res.*, vol. 260, no. 3, pp. 995–1013, 2017, doi: 10.1016/j.ejor.2017.01.011.

[37] M. Schneider, A. Stenger, and D. Goeke, “The electric vehicle-routing problem with time windows and recharging stations,” *Transp. Sci.*, vol. 48, no. 4, pp. 500–520, 2014, doi: 10.1287/trsc.2013.0490.

[38] M. L. Fisher and R. Jaikumar, “A generalized assignment heuristic for vehicle routing,” *Networks*, vol. 11, no. 2, pp. 109–124, 1981, doi: 10.1002/net.3230110205.