ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ»

ΕΙΔΙΚΕΥΣΗ: ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ ΚΑΙ ΑΝΑΛΥΤΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

HOT SPOT ΑΝΑΛΥΣΗ ΣΕ ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ ΤΡΟΧΙΩΝ ΟΧΗΜΑΤΩΝ ΣΕ ΟΔΙΚΟ ΔΙΚΤΥΟ

Ξενοφώντας Ψυχής – Φίλης

ΑΜ: 1827

Επιβλέπων: Χρήστος Δουλκερίδης, Επίκουρος Καθηγητής

ΠΕΙΡΑΙΑΣ

ΦΕΒΡΟΥΑΡΙΟΣ 2020

HOT SPOT ΑΝΑΛΥΣΗ ΣΕ ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ ΤΡΟΧΙΩΝ ΟΧΗΜΑΤΩΝ ΣΕ ΟΔΙΚΟ ΔΙΚΤΥΟ

Η ολοκλήρωση της διπλωματικής εργασίας χρηματοδοτήθηκε από το ΙΚΥ στο πλαίσιο του «προγράμματος χορήγησης υποτροφιών για μεταπτυχιακές σπουδές πρώτου κύκλου (Master) στην Ελλάδα με ένταξη στην αγορά εργασίας, στο πλαίσιο συνεργασίας του Ιδρύματος Κρατικών Υποτροφιών (ΙΚΥ) και της Εθνικής Τράπεζας της Ελλάδος (ΕΤΕ), ακαδημαϊκού έτους 2018-2019».

ΠΕΙΡΑΙΑΣ

ΦΕΒΡΟΥΑΡΙΟΣ 2020

Περίληψη

Στη παρούσα διπλωματική εργασία υλοποιήθηκε μια προσέγγιση για την εύρεση σημείων κυκλοφοριακής συμφόρησης (Hot-Spot) στο οδικό δίκτυο. Με τη ραγδαία αύξηση της του διαδικτύου σε φορητές συσκευές, όλο και περισσότεροι είναι οι χρήστες που επιλέγουν να χρησιμοποιούν μια συσκευή η οποία να μπορεί να τους διευκολύνει στις μετακινήσεις τους στο δρόμο. Τέτοια παραδείγματα συσκευών είναι τα κινητά τηλέφωνα, τα tablets και οι συσκευές πλοήγησης (GPS). Μέσω αυτών, ο κάθε χρήστης μπορεί να μετακινηθεί ανά πάσα στιγμή από ένα σημείο σε ένα άλλο, ακολουθώντας τις οδηγίες ενός ηλεκτρονικού χάρτη.

Κάθε μία από τις προαναφερθέντες συσκευές, εκπέμπει στοιχεία τοποθεσίας του χρήστη κάθε χρονική στιγμή που έχει ενεργοποιημένη την υπηρεσία πλοήγησης. Με τον τρόπο αυτό, η εφαρμογή, μπορεί να κατευθύνει το χρήστη στο σημείο ενδιαφέροντός του μέσα από μια συγκεκριμένη διαδρομή.

Για τη βέλτιστη μετάβαση του χρήστη, λαμβάνεται υπόψη η κυκλοφοριακή συμφόρηση που παρατηρείται εκείνη τη χρονική στιγμή στους δρόμους ενδιαφέροντος. Με τη χρήση χωροχρονικών δεδομένων αλλά και τεχνικές επεξεργασίας Μεγάλων Δεδομένων, προσπαθούμε να εντοπίσουμε περιπτώσεις όπου οι οδικές αρτηρίες αντιμετωπίζουν προβλήματα συμφόρησης, καθώς και την επιρροή των γειτονικών τους δρόμων σε αυτές.

Λέξεις Κλειδιά: Hot Spot, Τροχιές, Γράφος, Αλγόριθμος GetisOrd, Παράλληλη Επεξεργασία

Abstract

In the present thesis, an approach was implemented for finding Hot-Spots on road networks. With the rise of the Internet on mobile devices, more and more users are choosing to use a device that can facilitate their mobility on the road. Examples of such devices are cell phones, tablets and navigation devices (GPS-based). Through such devices, any user can move from one point to another at any time, following the instructions of an electronic map.

Each of the aforementioned devices transmits information of the user's location each time the navigation service is activated. In this way, the application can guide the user to their point of interest through a specific route.

For the optimal user's route, the traffic congestion at the roads of interest, is taken into account. Using spatial data and Big Data processing techniques, we try to identify cases where roads face congestion problems and the influence that neighboring roads have on them.

Key Words: Hot Spot, trajectories, graphs, GetisOrd Algorithm, Parallel Distribution

Περιεχόμενα

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον Επιβλέπων της εργασίας μου, κ. Χρήστο Δουλκερίδη, Επίκουρο Καθηγητή στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς, για την ανάθεση του θέματος και τη συνεχή βοήθειά που μου προσέφερε, για τις πολύτιμες συμβουλές του, για τη συνεργασία και το ενδιαφέρον που έδειξε από την αρχή της εργασίας μέχρι και τη περάτωσή της. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον Υποψήφιο Διδάκτορα κ. Παναγιώτη Νικητόπουλο, για το ενδιαφέρον που έδειξε στο θέμα. Ιδιαίτερες ευχαριστίες θα ήθελα να δώσω στο συμφοιτητή μου Αλέξανδρο Βώσσο, με τον οποίο αφιερώσαμε πολλές ώρες αναλύοντας σκέψεις και προβληματισμούς μας κατά τη διάρκεια των σπουδών μας, βοηθώντας παράλληλα ο ένας τον άλλο ώστε να γίνουμε καλύτεροι. Έπειτα, θα ήθελα να ευχαριστήσω το Ίδρυμα Κρατικών Υποτροφιών και την Εθνική Τράπεζα της Ελλάδος, που αποτέλεσαν αρωγοί στη περάτωση των σπουδών μου αλλά και την ένταξή μου στην αγορά εργασίας. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την υπομονή και τη στήριξή τους καθ’ όλη τη διάρκεια των σπουδών μου.

Ξενοφώντας Ψυχής – Φίλης

Πειραιάς,

Φεβρουάριος 2020

Κεφάλαιο 1

Εισαγωγή

Η σημερινή εποχή διέπεται από σημαντικές εξελίξεις στις τεχνολογίες που βασίζονται στο GPS, με αποτέλεσμα τα δεδομένα θέσης των κινητών αντικειμένων να βρίσκονται πάντα και παντού. Μέσω συστημάτων εντοπισμού οχημάτων, προσώπων αλλά ακόμη και ζώων, συλλέγονται τεράστιες ποσότητες δεδομένων αντικειμένων τροχιάς, τα οποία με τη σειρά τους υπόκεινται σε διάφορες εργασίες εξόρυξης δεδομένων. Τα τελευταία χρόνια, ιδιαίτερα η ανάλυση των δεδομένων κίνησης έχει μεγάλο ενδιαφέρον, καθώς τα αποτελέσματα που προκύπτουν από την ανάλυση παλαιών και τρεχόντων δεδομένων θέσης οχημάτων παρέχουν σημαντική συμβολή στη διαχείριση της κυκλοφορίας, στον έλεγχο και στον προγραμματισμό. Ένα βασικό χαρακτηριστικό των αντίστοιχων προσεγγίσεων είναι η ανίχνευση Hot Spot, δηλαδή οδικών τμημάτων και διασταυρώσεων που αντιμετωπίζουν υψηλό φορτίο κίνησης και συνεπώς απαιτούν ιδιαίτερη προσοχή στη διαχείριση της κυκλοφορίας.

Αντικείμενο Διπλωματικής

Στόχος της παρούσας διπλωματικής εργασίας είναι η ανεύρεση σημείων συμφόρησης (Hot Spot) στο οδικό δίκτυο, με τη χρήση δεδομένων GPS από τροχιές οχημάτων. Η προσέγγιση που μελετάμε αντικατοπτρίζει τον οδικό χάρτη σαν γράφο, όπου κάθε διασταύρωση δύο ή περισσότερων δρόμων είναι ένας κόμβος και ο δρόμος – η γραμμή που ενώνει τον ένα κόμβο με τον άλλο – είναι μια ακμή. Χρησιμοποιώντας τις πληροφορίες μιας τροχιάς, της ακμής στην οποία κινείται αλλά και των γειτόνων αυτής, μπορούμε να εντοπίσουμε αν η ακμή ενδιαφέροντος είναι Hot Spot ή όχι. Στη συνέχεια της αναφοράς, περιγράφεται η μέθοδος που βοηθά στην επίτευξη του στόχου.

Οργάνωση Κειμένου

Αναφορικά με τη δομή της διπλωματικής εργασίας, στο Κεφάλαιο 2, παρουσιάζονται εργασίες σχετικές με το αντικείμενο της διπλωματικής. Το Κεφάλαιο 3 συζητά τις τεχνολογίες που χρησιμοποιήθηκαν στη παρούσα διπλωματική εργασία. Στο Κεφάλαιο 4 αναφερόμαστε στους αλγορίθμους που αναπτύχθηκαν για την επίλυση του προβλήματος. Εν συνεχεία, στο Κεφάλαιο 5 περιγράφουμε τον τρόπο υλοποίησης των αλγορίθμων καθώς και τα δεδομένα τα οποία χρησιμοποιήθηκαν. Στο Κεφάλαιο 6 εμπεριέχονται οι πειραματισμοί μας και οι μετρήσεις αξιολόγησης των αποτελεσμάτων μας. Τέλος, στο Κεφάλαιο 7 αναφέρουμε τα συμπεράσματα της έρευνας, ενώ στο Κεφάλαιο 8 παραθέτουμε τη βιβλιογραφία που χρησιμοποιήθηκε για τη διεκπεραίωση της διπλωματικής αυτής εργασίας.

Κεφάλαιο 2: Σχετικές Εργασίες

Στο κεφάλαιο αυτό, θα αναφερθούμε περιληπτικά σε εργασίες που αφορούν την Ανάλυση Hot Spot . Καθώς η τεχνολογία εξελίσσεται με ταχύτατους ρυθμούς, η ανάγκη για εξερεύνηση νέων τεχνικών ανεύρεσης Hot Spot με δεδομένα τροχιών γίνεται ολοένα και μεγαλύτερη.

Hot Spot Ανάλυση είναι μια ανάλυση που μελετά την αναγνώριση της ομαδοποίησης χωρικών φαινομένων. Αυτά τα χωρικά φαινόμενα απεικονίζονται ως περιοχές σε ένα χάρτη και αναφέρονται σε τοποθεσίες γεγονότων ή αντικειμένων. Για τις ανάγκες της παρούσας εργασίας, μελετήθηκαν επιστημονικά άρθρα και εργασίες, που μας ενέπνευσαν και μας βοήθησαν στη κατανόηση και την επίλυση του προβλήματος που μελετάμε.

Έχουν αναπτυχθεί καινοτόμα μοντέλα και αλγόριθμοι για την ανίχνευση στατιστικά σημαντικών γραμμικών Hotspot, με τη βοήθεια γράφων. Δοθέντος ενός χωρικού δικτύου και μια συλλογής από γεγονότα (π.χ. αναφορές εγκλημάτων, αναφορές ατυχημάτων), εντοπίζονται όλα τα σύντομα μονοπάτια στο δίκτυο αυτό όπου η συγκέντρωση δραστηριοτήτων είναι υψηλά στατιστικά σημαντική. Λαμβάνοντας υπόψη τους γειτονικούς κόμβους μια ακμής, τα συντομότερα μονοπάτια (shortest paths) και χρησιμοποιώντας τη συνάρτηση Monte Carlo και τη τεχνική κλαδέματος δέντρου (tree pruning), προσπαθούν να ανακαλύψουν στατιστικά σημαντικές ακμές συσσώρευσης δραστηριοτήτων. Ως μετρική αξιολόγησης του αποτελέσματος χρησιμοποιείται το στατιστικό μέτρο p -value το οποίο αν ξεπερνά μια προκαθορισμένη τιμή κατωφλίου, κατηγοριοποιεί το αποτέλεσμα σε Hot Spot ή μη [1].

Μία ακόμα μελέτη, είναι αυτή του προβλήματος της εύρεσης διαδρόμων από δεδομένα τροχιάς. Η λύση στο πρόβλημα αυτό έγκειται στον εντοπισμό τοποθεσιών που εμφανίζονται συχνά μαζί, χρησιμοποιώντας τη Λανθάνουσα κατανομή Dirichlet (LDA), που επιτρέπει σε ομάδες παρατηρήσεων να ερμηνεύονται από άγνωστες ομάδες, που εξηγούν γιατί ορισμένα τμήματα των δεδομένων είναι παρόμοια [2]. Η προσέγγιση αυτή χωρίζει το χώρο σε ίσα κελιά και κατηγοριοποιεί τις υπάρχουσες τροχιές σε υπό – τροχιές για κάθε ζευγάρι κελιών. Αναζητούνται τα κελιά της τροχιάς έναντι των συχνών συνόλων, για τα κοινά κελιά. Δημιουργείται μια υπό-τροχιά χρησιμοποιώντας τα κελιά μεταξύ του πρώτου και του τελευταίου κελιού της τροχιάς, που είναι κοινά με το πιο σύνηθες σύνολο. Αν υπάρχουν περισσότερα από θ διαδοχικά αταίριαστα κελιά, η τροχιά χωρίζεται σε δύο νέες υπό - τροχιές.

Προκειμένου να συγκεντρωθούν οι υπό - τροχιές μαζί, εφαρμόστηκε ο αλγόριθμος ιεραρχικής συσταδοποίησης στις υπό - τροχιές κάθε συχνού συνόλου. Οι συστάδες σταματούν να συγχωνεύονται όταν οι αποστάσεις εντός της συστάδας υπερβαίνουν ένα προκαθορισμένο κατώφλι. Οι διάδρομοι δημιουργούνται για κάθε συχνό σετ δίνοντας μια δέσμη παρόμοιων τροχιών. Ο χαμηλός ρυθμός δειγματοληψίας GPS προκαλεί αβεβαιότητα στις μεμονωμένες τροχιές, οπότε κατασκευάζεται ένα κατευθυνόμενο γράφημα με ακρίβεια κατηγοριοποίησης. Αυτό περιλαμβάνει ένα σύνολο κορυφών που αντιστοιχούν σε κελιά και κατευθυνόμενες ακμές που συνδέουν γειτονικά κελιά. Το βάρος είναι η πιθανότητα να μετακινηθεί σε εκείνη την κορυφή και υπολογίζεται με βάση τη συχνότητα επισκέψεων του κελιού και την πιο κοινή κατεύθυνση των κινούμενων αντικειμένων. Η πιο πιθανή διαδρομή εξάγεται χρησιμοποιώντας το συντομότερο μονοπάτι και χρησιμοποιείται για να ολοκληρώσει ελλιπή τμήματα των τροχιών. Κάθε πλήρης διαδρομή προστίθεται στον κατάλογο των διαδρόμων εάν η ελάχιστη απόσταση μεταξύ αυτής και των ήδη εισαγόμενων διαδρόμων δεν υπερβαίνει ένα προκαθορισμένο όριο. [3]

Ενδιαφέρον συγκεντρώνει και μια νέα προσέγγιση για την ανάλυση των επιμέρους διασταυρώσεων, που επιτρέπει στους αναλυτές της κυκλοφορίας να υπολογίζουν τα μήκη ουράς και τον εκτιμώμενο χρόνο για να περάσουν μια διασταύρωση. Τέτοιες πληροφορίες μπορούν να χρησιμοποιηθούν για να επαληθεύσουν ότι οι σηματοδοτούμενες διασταυρώσεις είναι σωστά συντονισμένες και έτσι οι κατασκευαστές συσκευών GPS να μπορούν να συμβουλεύουν σε πραγματικό χρόνο τους οδηγούς σχετικά με την αναμενόμενη συμπεριφορά των σηματοδοτημένων διασταυρώσεων. Οι τροχιές χρησιμοποιούνται για τον υπολογισμό δεικτών όπως ο χρόνος που χρειάζεται για να περάσει η τροχιά το σημείο τομής, ο αριθμός των στάσεων που πραγματοποιεί μια τροχιά για να διασχίσει έναν δρόμο, την ελεύθερη ταχύτητα ροής κλπ. Ο αριθμός των στάσεων που πραγματοποιούνται από ένα όχημα μπορεί να αποτελεί ένδειξη ελαττωματικών ανιχνευτών βρόχου, ανεπαρκώς χρονισμένων σηματοδοτών κ.ά.. Ο αριθμός των στάσεων μαζί με την απόσταση που καλύπτεται μεταξύ τους, μπορεί να υποδεικνύει τον μέγιστο αριθμό αυτοκινήτων που μπορούν να περάσουν σε κάθε κύκλο μιας διασταύρωσης. Η ελεύθερη ταχύτητα ροής μπορεί να βρεθεί υπολογίζοντας κατά μέσο όρο την ταχύτητα των κορυφαίων x% ταχύτερων τροχιών [4].

Κεφάλαιο 3: Τεχνολογικά Εργαλεία

Python

Η Python είναι μια ερμηνευμένη (interpreted), υψηλού επιπέδου (high level) γλώσσα προγραμματισμού. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991, με τη φιλοσοφία σχεδίασης της Python να βασίζεται στην αναγνωσιμότητα του κώδικα. Η κατασκευή της και η αντικειμενοστραφής προσέγγιση που διαθέτει, στοχεύουν να βοηθήσουν τους προγραμματιστές να γράψουν έναν σαφή, λογικό κώδικα για μικρά και μεγάλα έργα. Επίσης, διαθέτει αυτόνομη διαχείριση μνήμης και δυναμική ανάλυση ονομάτων (late binding), η οποία δεσμεύει τα ονόματα μεθόδων και μεταβλητών κατά την εκτέλεση του προγράμματος.

Ο σχεδιασμός της Python προσφέρει μερική υποστήριξη για συναρτησιακό προγραμματισμό ακολουθώντας την παράδοση της Lisp. Διαθέτει συναρτήσεις όπως map, filter και reduce, ενώ διαχειρίζεται τύπους δεδομένων σε λίστες (lists), λεξικά (dictionaries), σύνολα (set) και εκφράσεις (generator expressions). Η βασική βιβλιοθήκη αποτελείται δύο ενότητες (itertools και functools) που υλοποιούν λειτουργικά εργαλεία βασισμένα στις γλώσσες προγραμματισμού Haskell και Standard ML. Επίσης, αντί να έχει όλες τις λειτουργίες του ενσωματωμένες στον πυρήνα της, η Python σχεδιάστηκε για να είναι εξαιρετικά επεκτάσιμη (extensible). Αυτή η συμπαγής διαμόρφωση την έχει καταστήσει ιδιαίτερα δημοφιλή ως μέσο προσθήκης προγραμματιζόμενων διεπαφών σε υπάρχουσες εφαρμογές.

Μια τέτοια εφαρμογή είναι και το εργαλείο μεγάλων δεδομένων που χρησιμοποιήθηκε στη παρούσα εργασία, το Spark Streaming, το οποίο είναι βασισμένο σε περιβάλλον Java και Scala. Μέσα από τη λειτουργία του Spark Python API (pyspark), ο χρήστης μπορεί να εφαρμόσει τις διαδικασίες του μετασχηματισμού και των πράξεων του Spark με τη χρήση κώδικα Python.

Αρχιτεκτονική Μεγάλων Δεδομένων

Κατανεμημένα Συστήματα: DFS - HDFS

Με την ανάγκη για ανάλυση μεγάλου όγκου δεδομένων να αυξάνεται όλο και περισσότερο και με την υπολογιστική δύναμη ενός μόνο υπολογιστή να μη μπορεί να ανταπεξέλθει στις ανάγκες της εποχής, η ανάπτυξη της παράλληλης επεξεργασίας δεδομένων τείνει να γίνει μονόδρομος.

Η θεωρία πίσω από τη δημιουργία μιας τέτοιας δυνατότητας βασίζεται σε ένα κατανεμημένο σύστημα αρχείων (DFS). Στην επιστήμη των υπολογιστών, ένα κατανεμημένο ή δικτυακό σύστημα αρχείων, είναι κάθε σύστημα αρχείων που δίνει τη δυνατότητα στο χρήστη να έχει πρόσβαση σε δεδομένα που διαμοιράζονται σε ένα δίκτυο υπολογιστών και είναι αποθηκευμένα σε ένα διακομιστή. Τα δεδομένα προσπελάζονται και επεξεργάζονται σαν να ήταν αποθηκευμένα στο τοπικό μηχάνημα. Το DFS διευκολύνει την ανταλλαγή πληροφοριών και αρχείων μεταξύ χρηστών σε ένα δίκτυο με ελεγχόμενο τρόπο. Ο διακομιστής επιτρέπει στους χρήστες του προγράμματος να μοιράζονται αρχεία και να αποθηκεύουν δεδομένα ακριβώς όπως αποθηκεύουν τοπικά τις πληροφορίες. Ωστόσο, οι διακομιστές έχουν πλήρη έλεγχο των δεδομένων και παρέχουν έλεγχο πρόσβασης στους πελάτες. Παράδειγμα ενός τέτοιου συστήματος είναι το Hadoop Distributed File System (HDFS), πάνω στο οποίο στηρίχθηκε και το σύστημα παράλληλης επεξεργασίας Apache Spark.

Το HDFS έχει πολλές ομοιότητες με τα υπάρχοντα κατανεμημένα συστήματα αρχείων. Ωστόσο, οι διαφορές από άλλα κατανεμημένα συστήματα αρχείων είναι σημαντικές. Το HDFS είναι εξαιρετικά ανεκτικό σε σφάλματα και έχει σχεδιαστεί για χρήση σε υλικό χαμηλού κόστους και για εφαρμογές που διαθέτουν μεγάλα σύνολα δεδομένων. Το HDFS χαλαρώνει μερικές απαιτήσεις POSIX για να επιτρέψει τη ροή δεδομένων σε δεδομένα αρχείων. Δημιουργήθηκε αρχικά ως υποδομή για το πρόγραμμα μηχανών αναζήτησης Apache Nutch, ενώ τώρα είναι ένα υποπρόγραμμα του Apache Hadoop.

Κάποια πράγματα που διαχωρίζουν το HDFS από τα υπόλοιπα κατανεμημένα συστήματα είναι πως έχει σχεδιαστεί με μέριμνα το λάθος, έχει κτιστεί για μεγάλο όγκο δεδομένων

και υποστηρίζει ετερογενείς συστάδες υπολογιστών. Τα δεδομένα σε μια συστάδα Hadoop αναλύονται σε μικρότερες μονάδες (αποκαλούμενα μπλοκ) και κατανέμονται σε όλο τη συστάδα υπολογιστών. Κάθε τετράγωνο αντιγράφεται δύο φορές (για συνολικά τρία αντίγραφα), με τα δύο αντίγραφα που είναι αποθηκευμένα σε δύο κόμβους σε ένα rack κάπου αλλού στη συστάδα. Δεδομένου ότι τα αντίγραφα των δεδομένων είναι τρία, είναι εξαιρετικά ανεκτικό σε σφάλματα. Αν ένα αντίγραφο χαθεί (εξαιτίας της βλάβης του μηχανήματος, για παράδειγμα), το HDFS θα δημιουργήσει αυτόματα άλλο αντίγραφο σε άλλο σημείο της συστάδας, διασφαλίζοντας ότι διατηρείται ο τριπλός παράγοντας αναπαραγωγής.

Το HDFS βασίζεται σε αρχιτεκτονική leader / follower. Κάθε συστάδα υπολογιστών αποτελείται συνήθως από ένα μοναδικό NameNode, έναν προαιρετικό SecondaryNameNode (για ανάκτηση δεδομένων σε περίπτωση αποτυχίας) και έναν αυθαίρετο αριθμό DataNodes.

Εκτός από τη διαχείριση του χώρου ονομάτων του συστήματος αρχείων και των συναφών μεταδεδομένων (αντιστοίχιση αρχείου-μπλοκ), το NameNode ενεργεί ως κύριος μεσολαβητής για πρόσβαση στα αρχεία από τους πελάτες (αν και μετά τη μεσολάβηση, οι πελάτες επικοινωνούν απευθείας με DataNodes). Το NameNode λειτουργεί εξ ολοκλήρου στη μνήμη, διατηρώντας την κατάστασή του στο δίσκο. Για να μπορέσουν να μετριάσουν τις περιπτώσεις αποτυχίας, οι συστάδες παραγωγής συνήθως διατηρούν την κατάσταση του NameNode σε δύο τοπικούς δίσκους (σε περίπτωση αποτυχίας ενός δίσκου) και επίσης σε έναν όγκο εγκατεστημένο σε NFS (σε περίπτωση πλήρους βλάβης του μηχανήματος). Στη λειτουργία υψηλής διαθεσιμότητας, το Hadoop διατηρεί ένα NameNode σε κατάσταση αναμονής για να αποφύγει τις αποτυχίες.

Αρχιτεκτονική Spark

Το Apache Spark είναι ένα γρήγορο και γενικευμένο σύστημα συστάδων υπολογιστών. Παρέχει API υψηλού επιπέδου σε Java, Scala, Python και R και έναν βελτιστοποιημένο μηχανισμό που υποστηρίζει γενικά γραφήματα εκτέλεσης. Υποστηρίζει επίσης ένα

πλούσιο σύνολο εργαλείων υψηλότερου επιπέδου, όπως Spark SQL για SQL και δομημένη επεξεργασία δεδομένων, MLlib για εκμάθηση μηχανών, GraphX για επεξεργασία γραφημάτων και Spark Streaming. Το Apache Spark έχει μια καλά καθορισμένη και πολύ επίπεδη αρχιτεκτονική όπου όλα τα συστατικά και τα στρώματα είναι χαλαρά συνδεδεμένα και ενσωματωμένα με διάφορες επεκτάσεις και βιβλιοθήκες. Η αρχιτεκτονική Apache Spark βασίζεται σε δύο βασικές αρχές, τα Resilient Distributed Datasets (RDD) και το Directed Acyclic Graph (DAG).

Τα RDD είναι συλλογή στοιχείων δεδομένων που χωρίζονται σε κατατμήσεις και μπορούν να αποθηκευτούν σε μνήμη στους εργάτες κόμβους του cluster. Η δημιουργία ενός RDD μπορεί να γίνει είτε με παραλληλοποίηση μιας υπάρχουσας συλλογής δεδομένων στον τοπικό δίσκο, είτε σε ένα εξωτερικό σύστημα αποθήκευσης, όπως ένα κοινόχρηστο σύστημα αρχείων, HDFS, HBASE ή μια οποιοδήποτε πηγή δεδομένων στα πρότυπα του Hadoop.Το Spark RDD υποστηρίζει δύο διαφορετικούς τύπους πράξεων, Μετασχηματισμούς και Δράσεις (Transformations and Actions).

Το DAG είναι μια ακολουθία υπολογισμών που εκτελούνται σε δεδομένα όπου κάθε κόμβος είναι ένα διαμέρισμα RDD και η ακμή είναι ένας μετασχηματισμός πάνω από τα δεδομένα. Επίσης, αποτρέπει τις συνεχείς εφαρμογές της τεχνικής Map - Reduce και παρέχει βελτιώσεις επιδόσεων σε σχέση με την Hadoop.

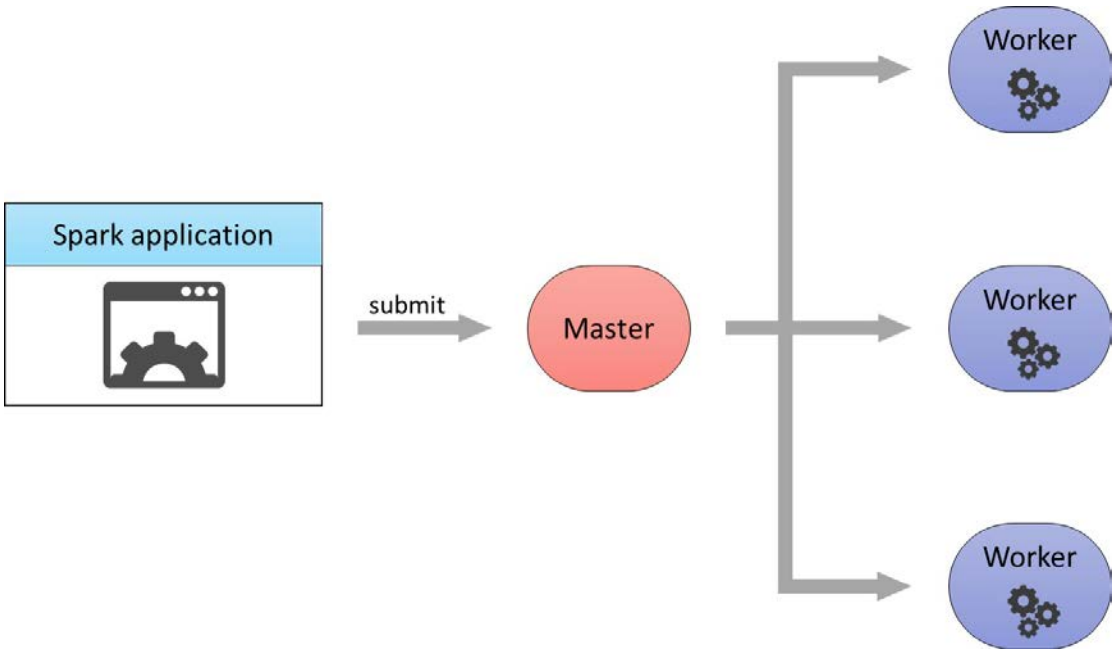
Το Apache Spark ακολουθεί κι αυτό όπως και το Hadoop την αρχιτεκτονική leader / follower με δύο βασικούς Daemons και έναν διαχειριστή συστάδας Master Daemon - διαδικασία Master / Driver - Διαδικασία Slave. Μια συστάδα Spark αποτελείται από ένα Master Node και μπορεί να έχει παραπάνω από ένα Slaves. Ο οδηγός και οι εκτελεστές εκτελούν τις μεμονωμένες διεργασίες Java και οι χρήστες μπορούν να τα εκτελέσουν στην ίδια συστάδα υπολογιστών ή σε ξεχωριστές μηχανές.

Ο Driver είναι το κεντρικό σημείο εισόδου μέσω της γραμμής εντολών (Scala, Python, and R). Το πρόγραμμα οδήγησης εκτελεί τη λειτουργία main() της εφαρμογής και είναι ο τόπος όπου δημιουργείται το περιβάλλον Spark. Το Spark Driver περιέχει διάφορα στοιχεία - DAGScheduler, TaskScheduler, BackendScheduler και BlockManager που είναι

υπεύθυνοι για τη μετάφραση του κώδικα χρήστη σε πραγματικές εργασίες Spark που εκτελούνται στη συστάδα υπολογιστών. Το πρόγραμμα οδήγησης που εκτελείται στον κύριο κόμβο της συστάδας, προγραμματίζει την εκτέλεση της εργασίας και διαπραγματεύεται με τον διαχειριστή της συστάδας. Μεταφράζει τα RDD στο γράφημα εκτέλεσης και χωρίζει το γράφημα σε πολλαπλά στάδια. Ο οδηγός αποθηκεύει τα μεταδεδομένα για όλες τις κατανεμημένες βάσεις δεδομένων και τις κατατμήσεις τους. Το πρόγραμμα οδήγησης μετατρέπει μια εφαρμογή χρήστη σε μικρότερες μονάδες εκτέλεσης γνωστές ως εργασίες (tasks). Οι εργασίες εκτελούνται στη συνέχεια από τους εκτελεστές (workers).

Ο εκτελεστής είναι ένας διανεμημένος “πράκτορας” υπεύθυνος για την εκτέλεση των εργασιών. Κάθε εφαρμογή Spark έχει τη δική της διαδικασία εκτέλεσης. Οι εκτελεστές τρέχουν συνήθως καθ’ όλη τη διάρκεια της εφαρμογής Spark και αυτό το φαινόμενο είναι γνωστό ως "Static Allocation of Executors". Ωστόσο, οι χρήστες μπορούν επίσης να επιλέξουν δυναμικές κατανομές των εκτελεστών όπου μπορούν να προσθέσουν ή να αφαιρέσουν δυναμικά τους spark εκτελεστές ώστε να ταιριάζουν με το συνολικό φόρτο εργασίας. Επίσης, ο εκτελεστής, διαβάζει και γράφει δεδομένα σε εξωτερικές πηγές, αποθηκεύει τα υπολογιστικά αποτελέσματα στη μνήμη ή στο σκληρό δίσκο και τέλος αλληλεπιδρά με τα συστήματα αποθήκευσης.

Τέλος, ο Master Manager είναι μια εξωτερική υπηρεσία υπεύθυνη για την απόκτηση πόρων στη συστάδα και την κατανομή τους σε μια εργασία. Υπάρχουν 3 διαφορετικοί τύποι διαχειριστών συμπλεγμάτων. Μια εφαρμογή Spark μπορεί να αξιοποιήσει την κατανομή και την απομετάλλευση διαφόρων φυσικών πόρων, όπως μνήμη για εργασίες πελατών, μνήμη CPU, κλπ. Η επιλογή ενός διαχειριστή σε μια συστάδα υπολογιστών για οποιαδήποτε εφαρμογή εξαρτάται από τους στόχους της εφαρμογής, επειδή όλοι οι διαχειριστές συμπλεγμάτων παρέχουν διαφορετικές δυνατότητες προγραμματισμού.



Εικόνα 1: Αρχιτεκτονική Spark

MapReduce – Secondary Sort

Καθώς το Spark είναι βασισμένο στο σύστημα του Hadoop, ως βασική του λειτουργία έχει το MapReduce. Το MapReduce παρουσιάστηκε για πρώτη φορά από τη Google το 2004. Ήταν η λύση που έδωσε όταν δημιουργήθηκε η ανάγκη για την επεξεργασία μεγάλων όγκων δεδομένων (όπως ο υπολογισμός του αριθμού των backlinks για εκατομμύρια σελίδες) με τη διανομή υπολογισμών σε χιλιάδες μηχανές. Τα συστήματα MapReduce παρέχουν δεδομένα από έναν mapper σε έναν reducer, ομαδοποιημένα με ένα κλειδί. Η ιδέα πίσω από το MapReduce είναι αρκετά απλή. Ξεκινά με το map - μετατροπή των δεδομένων σε μια μορφή που ταιριάζει με το πρόβλημα, συνεχίζοντας με την ανακατεύθυνση – μετακίνηση (shuffle) δεδομένων μεταξύ των μηχανών μέχρι όλα τα δεδομένα που πρέπει να υποβληθούν ταυτόχρονα σε επεξεργασία στο ίδιο μηχάνημα με μια φάση μείωσης – επεξεργασίας (reduce) κατά την οποία υπολογίζεται το πραγματικό αποτέλεσμα. Όλες οι φάσεις που παρουσιάζονται εκτελούνται σε μεγάλο αριθμό μηχανών παράλληλα. Εάν κατά τη διάρκεια της εκτέλεσης του MapReduce ένα μηχάνημα αποτύχει, η όλη διαδικασία συνεχίζεται επαναλαμβάνοντας τις εργασίες του μηχανήματος και μη επανεκκινώντας την πλήρη διαδικασία. Πέραν του ότι ένα μόνο μηχάνημα δεν μπορούσε να χειριστεί τέτοιους όγκους δεδομένων, η εκτέλεση μιας

τέτοιας εργασίας σε ένα μόνο μηχάνημα θα σήμαινε αναγκαστικά την επανεκκίνηση ολόκληρης της διαδικασίας σε περίπτωση αποτυχίας.

Από το 2004 κι έπειτα, γεννήθηκαν πολλαπλά προϊόντα τύπου MapReduce που ακολουθούν περισσότερο ή λιγότερο τις τρεις φάσεις του MapReduce για να επεξεργάζονται μεγάλους όγκους δεδομένων. Έχοντας πολλά προϊόντα που μπορούν να επιλύσουν το ίδιο εύρος προβλημάτων, οι μηχανικοί των δεδομένων άρχισαν να σχεδιάζουν μοτίβα σχεδίασης για τα συνήθη προβλήματα μεγάλου αριθμού δεδομένων. Τα πρότυπα σχεδίασης στην επιστήμη των υπολογιστών είναι πρότυπα που αποσκοπούν στην επίλυση των επαναλαμβανόμενων προβλημάτων και στη διευκόλυνση της επικοινωνίας μεταξύ των μηχανικών λογισμικού. Ένα τέτοιο πρότυπο είναι το Secondary Sort. Η “δευτερογενής ταξινόμηση” δεν είναι παρά η ταξινόμηση δεδομένων με δύο τιμές. Αυτό που κάνει το πρόβλημα δύσκολο είναι η ταξινόμηση μεγάλων όγκων δεδομένων όσο το δυνατόν γρηγορότερα, χωρίς να εξαντληθεί η μνήμη και συνεπώς να δημιουργηθούν προβλήματα στα μηχανήματα. Για το λόγο αυτό, στο Secondary Sort γίνεται ομαδοποίηση των δεδομένων με το κλειδί και ταξινόμηση των τιμών σε κάθε ζευγάρι τιμών προτού εκτελεστούν περαιτέρω υπολογισμοί [5].

Spark Streaming

Το εργαλείο πάνω στο οποίο αναπτύχθηκε η παρούσα διπλωματική εργασία, είναι το Apache Spark Streaming. Το εργαλείο αυτό είναι ένα κλιμακώσιμο σύστημα επεξεργασίας ροής που αντέχει σε σφάλματα, το οποίο υποστηρίζει εγγενώς τόσο τα batch όσο και τα streaming φορτία εργασίας. Το Spark Streaming είναι μια επέκταση του βασικού API Spark που επιτρέπει στους μηχανικούς δεδομένων και στους επιστήμονες δεδομένων να επεξεργάζονται δεδομένα σε πραγματικό χρόνο από διάφορες πηγές, συμπεριλαμβανομένων των Kafka, Flume, Amazon Kinesis και άλλων. Αυτά τα επεξεργασμένα δεδομένα μπορούν να αποθηκευτούν σε συστήματα αρχείων, βάσεις δεδομένων και σε συστήματα απεικόνισης δεδομένων σε πραγματικό χρόνο. Η βασική του αφαίρεση είναι το επονομαζόμενο DStream, το οποίο αντιπροσωπεύει μια ροή δεδομένων χωρισμένη σε μικρές παρτίδες. Τα DStreams βασίζονται σε RDD, τη βασική

συλλογή δεδομένων του Spark. Αυτό επιτρέπει στο Spark Streaming να ενσωματωθεί άψογα σε οποιοδήποτε άλλο στοιχείο Spark όπως MLlib και Spark SQL.

Το Spark Streaming είναι διαφορετικό από άλλα συστήματα που είτε διαθέτουν μια μηχανή επεξεργασίας σχεδιασμένη μόνο για streaming, είτε έχουν παρόμοια API batch και streaming, αλλά καταρτίζονται εσωτερικά σε διαφορετικές μηχανές. Η μοναδική μηχανή εκτέλεσης του Spark και το ενιαίο μοντέλο προγραμματισμού για batch και streaming οδηγούν σε μερικά μοναδικά πλεονεκτήματα σε σχέση με άλλα παραδοσιακά συστήματα ροής. Πιο συγκεκριμένα, τα τέσσερα πιο αξιοσημείωτα είναι η γρήγορη αποκατάσταση από αποτυχίες και παραμορφώσεις, η καλύτερη εξισορρόπηση φορτίου και χρήση πόρων, ο συνδυασμός δεδομένων ροής με στατικά σύνολα δεδομένων και η ενσωμάτωση ήδη υπαρχόντων βιβλιοθηκών επεξεργασίας όπως SQL, μηχανικής μάθησης και επεξεργασία γραφημάτων. Αυτή η ενοποίηση των διαφορετικών δυνατοτήτων επεξεργασίας δεδομένων είναι ο βασικός λόγος για την ταχεία υιοθέτηση του Spark Streaming. Είναι πολύ εύκολο για τους προγραμματιστές να χρησιμοποιούν ένα ενιαίο πλαίσιο για να ικανοποιήσουν όλες τις ανάγκες επεξεργασίας τους.

Micro – Batch Processing

Η επεξεργασία Micro-batch είναι μια πρακτική συλλογής δεδομένων σε μικρές παρτίδες (batches) με σκοπό την επεξεργασία τους. Σε αντίθεση με τη batch επεξεργασία, που συναντάμε στο Spark, η Micro-batch χρησιμοποιείται για την επεξεργασία μικρού όγκου δεδομένων, με τη συλλογή των δεδομένων όμως να γίνει κι αυτή βάση κάποιου προκαθορισμένου ορίου (threshold) ή συχνότητας. Στα πλαίσια της εργασίας αυτής, χρησιμοποιείται η τεχνική αυτή, με την οποία τροφοδοτούμε το Spark Streaming με υποσύνολα δεδομένων προς επεξεργασία.

Κεφάλαιο 4: Θεωρητικό Υπόβαθρο

Γράφοι

Στα μαθηματικά, και πιο συγκεκριμένα στη θεωρία γραφημάτων, ένα γράφημα είναι μια δομή που αντιστοιχεί σε ένα σύνολο αντικειμένων στα οποία κάποια ζεύγη αντικειμένων είναι κατά κάποιο τρόπο "συγγενή". [6] Τα αντικείμενα αντιστοιχούν σε μαθηματικές αφαιρέσεις που ονομάζονται κορυφές ή κόμβοι (nodes) και κάθε ένα από τα σχετικά ζεύγη κόμβων ονομάζεται ακμή (edge). Τυπικά, ένα γράφημα απεικονίζεται σε διαγραμματική μορφή ως σύνολο κουκκίδων ή κύκλων για τις κορυφές, ενώνονται με γραμμές ή καμπύλες για τις άκρες. Τα γραφήματα είναι ένα από τα αντικείμενα της μελέτης στα διακριτά μαθηματικά.

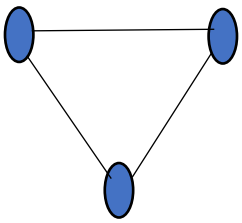
Οι ακμές μπορεί να είναι κατευθυνόμενες ή μη κατευθυνόμενες. Για παράδειγμα, εάν οι κόμβοι αντιπροσωπεύουν την αρχή και το τέλος ενός δρόμου (A,B) ο οποίος είναι διπλής κατεύθυνσης, τότε το γράφημα αυτό δεν είναι κατευθυνόμενο καθώς κάθε όχημα μπορεί να κινηθεί και προς τη μεριά του κόμβου Α αλλά και προς τον κόμβο Β. Αντίθετα, αν ο δρόμος αυτός είναι μονής κατεύθυνσης, τότε η κάθε τροχιά μπορεί να κινηθεί προς μια συγκεκριμένη κατεύθυνση. Ο πρώτος τύπος γραφήματος ονομάζεται μη κατευθυνόμενος γράφος ενώ ο δεύτερος τύπος γραφήματος ονομάζεται κατευθυνόμενος γράφος.

Ένας γράφος, είναι ένα ζεύγος $G = (V, E)$ όπου V είναι ένα σύνολο των οποίων ονομάζονται κόμβοι (nodes), και το E είναι ένα σύνολο δύο κόμβων–, των οποίων τα στοιχεία ονομάζονται ακμές (μερικές φορές συνδέσεις ή γραμμές). Επίσης, ένα πολυγραφικό (multigraph) είναι μια γενίκευση που επιτρέπει πολλαπλές ακμές δίπλα στο ίδιο ζεύγος κορυφών.

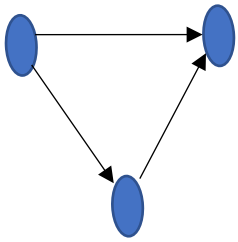
Γενικά, το σύνολο των κορυφών V θεωρείται ότι είναι πεπερασμένο κάνοντας και το σύνολο των ακμών να είναι επίσης πεπερασμένο. Πάρα πολλά γραφήματα μελετώνται αρκετές φορές, αλλά θεωρούνται πιο συχνά ως ένα ειδικό είδος δυαδικής σχέσης, καθώς τα περισσότερα αποτελέσματα σε πεπερασμένα γραφήματα δεν επεκτείνονται στην άπειρη περίπτωση ή χρειάζονται μια μάλλον διαφορετική απόδειξη.

Ένα κενό γράφημα είναι ένα γράφημα που έχει ένα άδειο σύνολο κόμβων (και επομένως ένα άδειο σύνολο ακμών). Η σειρά ενός γράφου είναι ο αριθμός των κορυφών $|V|$. Το μέγεθος ενός γράφου είναι ο αριθμός των άκρων του $|E|$. Ωστόσο, σε ορισμένα πλαίσια, όπως για παράδειγμα για την έκφραση της υπολογιστικής πολυπλοκότητας των αλγορίθμων, το μέγεθος είναι $|V| + |E|$ (διαφορετικά, ένα μη-κενό γράφημα θα μπορούσε να έχει μέγεθος 0). Ο βαθμός ενός κόμβου είναι ο αριθμός των ακμών που προκαλούνται από αυτό. Οι άκρες ενός γραφήματος ορίζουν μια συμμετρική σχέση στις κορυφές, που ονομάζεται σχέση γειτνίασης. Συγκεκριμένα, δύο κορυφές x και y είναι γειτονικές εάν $\{x, y\}$ είναι μια ακμή.

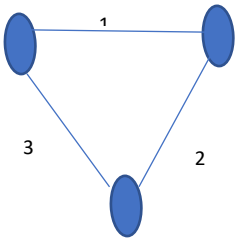
Τέλος, ένα σταθμισμένο γράφημα ή ένα δίκτυο είναι ένα γράφημα στο οποίο ένας αριθμός (βάρος) αντιστοιχεί σε κάθε ακμή. Αυτά τα βάρη θα μπορούσαν να αντιπροσωπεύουν, για παράδειγμα, το βαθμό στον οποίο μπορεί η συμφόρηση ενός δρόμου να επηρεάσει τη κυκλοφορία σε έναν γειτονικό του. Τέτοια γραφήματα προκύπτουν σε πολλά περιβάλλοντα, για παράδειγμα σε προβλήματα βέλτιστης διαδρομής, αλλά και σε προβλήματα όπως αυτό που εξετάζουμε στη παρούσα εργασία.



Εικόνα 2: Μη κατευθυνόμενος γράφος με τρεις κορυφές και τρεις ακμές



Εικόνα 3: Κατευθυνόμενος Γράφος με τρεις κορυφές και τρεις ακμές



Εικόνα 4: Σταθμισμένος Γράφος με βάρη 1,2,3

Αλγόριθμος Getis-Ord

Η προσέγγιση της παρούσας εργασίας, βασίζεται στον αλγόριθμο Getis-Ord ή αλλιώς στο στατιστικό Getis-Ord Gi*. Το στατιστικό αυτό ουσιαστικά υπολογίζει τις τιμές z-score εξετάζοντας κάθε χαρακτηριστικό (edge) στο πλαίσιο των γειτονικών σε αυτό χαρακτηριστικών. Οι προσκόπτουσες τιμές δείχνουν που συσσωρεύονται χωρικά τα χαρακτηριστικά με υψηλές ή χαμηλές τιμές. Ένα χαρακτηριστικό με υψηλή τιμή παρουσιάζει ενδιαφέρον, αλλά μπορεί να μην είναι ένα στατιστικά σημαντικό Hot Spot. Για να θεωρηθεί ένα χαρακτηριστικό, στατιστικά σημαντικό, εκτός του ότι πρέπει να έχει υψηλή τιμή, θα πρέπει και τα χαρακτηριστικά που το περιβάλλουν να έχουν εξίσου υψηλές τιμές. Το τοπικό άθροισμα για ένα χαρακτηριστικό και τους γείτονές του, συγκρίνεται αναλογικά με το άθροισμα όλων των χαρακτηριστικών. Όταν το τοπικό άθροισμα έχει πολύ μεγάλη απόκλιση από την αναμενόμενη τιμή του αθροίσματος και όταν αυτή η διαφορά είναι πολύ μεγάλη για να είναι αποτέλεσμα τυχαίας πιθανότητας, προκύπτει ένα στατιστικά σημαντικό αποτέλεσμα z-score.

Πιο συγκεκριμένα, η προσέγγιση της παρούσας εργασίας για την επίλυση του προβλήματος ανεύρεσης σημείων Hot Spot σε οδικό δίκτυο, αφορά την ανακάλυψη στατιστικά σημαντικών χωρο-χρονικών ακμών, όπου η σημαντικότητα της ακμής e_i είναι συνάρτηση του attribute value της ακμής x_i αλλά και των attribute value των γειτονικών, σε αυτή, ακμών.

Η συνάρτηση του στατιστικού GetisOrd Statistic G_i^*, ορίζεται ως εξής:

$$G^*{}_i = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{[n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2]}{n-1}}} \quad (1)$$

όπου x_i είναι το attribute value για την ακμή j, $w_{i,j}$ είναι το βάρος μεταξύ των ακμών i και j, n είναι ο συνολικώς αριθμός των ακμών και τα $\bar{X}$ και S , ο μέσος όρος και η τυπική απόκλιση αντίστοιχα, που υπολογίζονται ως εξής:

$$\bar{X} = \frac{\sum_{j=1}^{n} x_i}{n} \tag{2}$$

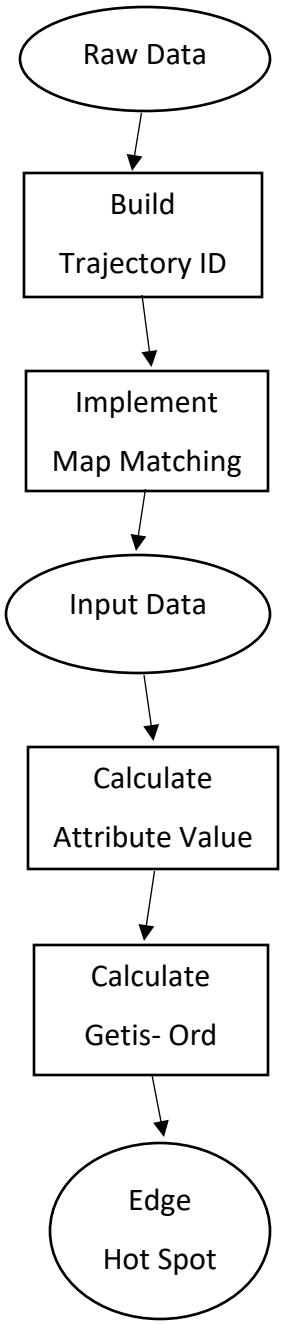
$$S = \sqrt{\frac{\Sigma_{j=1}^{n} x_j^2}{n} - (\bar{X})^2} \tag{3}$$

Κεφάλαιο 5: Προσέγγιση Προβλήματος

Η παρούσα εργασία προσπαθεί να λύση το πρόβλημα της εύρεσης Hot Spot σε οδικό δίκτυο. Για την επίλυση του συγκεκριμένου προβλήματος, προσεγγίσαμε το οδικό δίκτυο ως ένα κατευθυνόμενο γράφο, με τις κορυφές του να αντιπροσωπεύουν διασταυρώσεις δρόμων και τις ακμές τους δρόμους που δημιουργούνται από τις κορυφές αυτές. Απώτερος σκοπός είναι η εύρεση στατιστικά σημαντικών ακμών στο δίκτυο που εξετάζουμε, όπου η σημαντικότητα της ακμής είναι συνάρτηση του Attribute Value xi αυτής αλλά και των γειτονικών ακμών αυτής με τα αντίστοιχα Attribute Values. Για την επίτευξη αυτού του στόχου, υπολογίζεται το στατιστικό μέγεθος $Gi*$, από τον αλγόριθμο του Getis-Ord:

$$G^*_i = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{[n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2]}{n-1}}}$$

Για την εφαρμογή του αλγορίθμου, χρειάστηκε ο υπολογισμός του Attribute Value xi της εξεταζόμενης ακμής, οι γειτονικές ακμές καθώς και τα βάρη τους $w_{i,j}$. Στη παρούσα εργασία κάναμε κάποιες υποθέσεις ως προς την εύρεση των παραπάνω και κατά συνέπεια των υπολογισμό του στατιστικού $Gi*$, όπως φαίνεται και στα διάγραμμα 1.



Διάγραμμα 1 Προσέγγιση Προβλήματος

Δεδομένα Εισαγωγής

Αρχικά, τα δεδομένα εισαγωγής πρέπει να περιέχουν τις εξής πληροφορίες:

1. Αναγνωριστικό Οχήματος (Vehicle ID)
2. Αναγνωριστικό Τροχιάς (Trajectory ID)
3. Χρονικό Αποτύπωμα (Timestamp)
4. Συντεταγμένες της τροχιάς τη δεδομένη χρονική στιγμή (Latitude, Longitude)
5. Το όριο ταχύτητας του δρόμου που μελετάμε (Road Speed)
6. Το μήκος του δρόμου που μελετάμε (Length)
7. Τις δύο κορυφές της εξεταζόμενης ακμής, με τις συντεταγμένες τους (Node1, Node1Lat, Node1Lot, Node2, Node2Lat, Node2Lot)

Το σύνολο δεδομένων που είχαμε στη διάθεσή μας, είχε πέντε από τις παραπάνω πληροφορίες που μας είναι απαραίτητες, το αναγνωριστικό του οχήματος, το χρονικό αποτύπωμα του οχήματος, τις συντεταγμένες του οχήματος, το όριο ταχύτητας του δρόμου στο οποίο κινείται το όχημα καθώς και το μήκους του δρόμου αυτού. Έχοντας αυτά ως δεδομένα, δημιουργούμε τα υπόλοιπα πεδία, όπως αναλύεται στη συνέχεια.

Αναγνωριστικό Τροχιάς (Trajectory ID)

Αφορμώμενοι από τη στήλη του αναγνωριστικού τροχιάς, που περιέχεται στο σύνολο δεδομένων που πειραματιστήκαμε, θέλαμε να δώσουμε ένα δικό μας προσδιορισμό, ως προς τον ορισμό του Trajectory ID που θα θέλαμε να έχουμε για είσοδο στα δεδομένα της εφαρμογής μας. Λαμβάνοντας υπόψη το αναγνωριστικό του οχήματος αλλά και τη χρονική στιγμή της πληροφορίας, LocalDate, δημιουργήσαμε το αναγνωριστικό της τροχιάς όπως φαίνεται στον παρακάτω ψευδοκώδικα.

Trajectory ID Implementation: Build Trajectory ID

```
1: Input: V,D
2: Output: T
3: function
4: SET count to 1
5: FOR each index, row in DataSet
6:     IF index is not 0
7:         IF V at previous index is not equal to V at present index
               or D at present index minus D at previous index is larger than 5 minutes
8:             Add 1 to count
9:         END IF
10:    END IF
11:    SET new column T to V-count
12: END FOR
13: end function
```

Θεωρώντας πως τα δεδομένα είναι ταξινομημένα ανά όχημα V και ανά χρονική στιγμή D με αύξουσα σειρά, η παραπάνω υλοποίηση δέχεται ως είσοδο τη στήλη Vehicle_ID (V) και τη στήλη LocalDate (D) και επιστρέφει τη στήλη Traj_ID (T). Αρχικά αναθέτουμε σ' έναν μετρητή τη τιμή ένα για να κρατήσουμε τις φορές που αλλάζει το αναγνωριστικό τροχιάς βάση της υλοποίησης. Για κάθε τιμή της στήλης V με χρονική στιγμή D ελέγχουμε αν η επόμενη τιμή της V είναι ίδια με τη παρούσα που εξετάζουμε ή αν η χρονική στιγμή της παρούσας τιμής είναι διαφορετική κατά πέντε λεπτά από την επόμενη χρονική στιγμή. Αν ισχύει ένα από τα προαναφερθέντα, τότε αυξάνουμε το μετρητή κατά μια μονάδα και δίνουμε στη νέα στήλη T τη τιμή V-count+1, αλλιώς τη τιμή V-count. Ο περιορισμός που θέσαμε στη τιμή D, να είναι δηλαδή η τιμή του μεγαλύτερη από πέντε λεπτά, ορίστηκε με την υπόθεση ότι είναι εξαιρετικά απίθανο μια τροχιά να μείνει στο ίδιο σημείο παραπάνω από αυτή τη τιμή και έτσι θεωρούμε πως αλλάζει η τροχιά.

Προσδιορισμός θέσεων στο χάρτη (Map Matching)

Για τις ανάγκες της υλοποίησης, πρέπει να τοποθετήσουμε τις συντεταγμένες των σημείων στον οδικό χάρτη. Με τη χρήση της βιβλιοθήκης OSMnx της Python, δημιουργήσαμε ένα boundary box από τις συντεταγμένες του συνόλου δεδομένων, με κορυφές του τετραγώνου να είναι Min(Latitude) , Max(Latitude), Min(Longitude) και Max(Longitude). Χρησιμοποιώντας το πλαίσιο αυτό δημιουργούμε ένα γράφο - χάρτη G, ο οποίος περιέχει πληροφορίες για κάθε δρόμο – edge που περιέχεται σε αυτόν. Από εκεί με τη συνάρτηση get_nearest_edges, κάνουμε το map matching των συντεταγμένων της κάθε τροχιάς στη χρονική στιγμή LocalDate, στο γράφο G. Με το τρόπο αυτό παράγεται εκ νέου ένα αρχείο το οποίο εμπλουτίζεται σε σχέση με το αρχικό, με πληροφορίες της κάθε τροχιάς, για το μήκος της ακμής στην οποία κινείται (length), τις κορυφές από τις οποίες δημιουργείται (Node1, Node2) καθώς και από τις συντεταγμένες των κορυφών αυτών (Node1_Lon, Node1_Lat, Node2_Lon, Node2_Lat).

Map Matching Implementation

```
1: Input: Lat, Lon
2: Output: N1, N2, N1Lat, N1Lon, N2Lat, N2Lon, L
3: function
4: SET maxLat equals to max value of Latitude
5: SET minLat equals to min value of Latitude
6: SET maxLon equals to max value of Longitude
7: SET minLon equals to max value of Longitude
8: G = graphFromBbox (maxLat, minLat, maxLon, minLon)
9: NE = getNearestEdges(G, Lon, Lat, “BallTree”)
10: N1 = NE[0]
11: N2 = NE[1]
12: FOR each index, row in DataSet
13:     N1Lon = G.nodes(NE[N1,x])
14:     N1Lat  = G.nodes(NE[N1,y])
15:     N2Lon = G.nodes(NE[N2,x])
```

```
16:    N2Lat  = G.nodes(NE[N2,x])
17:    L = Distance((N1Lat, N1Lot), (N2Lat, N2Lot))
18: END FOR
19: end function
```

Η υλοποίηση του Map Matching, δέχεται ως είσοδο τις συντεταγμένες τις κάθε τροχιάς (Lat, Lon) για κάθε χρονική στιγμή D. Αποτέλεσμα της υλοποίησης είναι η εύρεση των κορυφών της ακμής στην οποία κινείται η τροχιά, οι συντεταγμένες των κορυφών αυτών καθώς και το μήκος της ακμής. Από τα ήδη υπάρχοντα δεδομένα, χρησιμοποιούμε τις συντεταγμένες της τροχιάς για να δημιουργήσουμε ένα bounding box και να το αποτυπώσουμε σε ένα γράφο G. Στη συνέχεια από το γράφο G και από τις συντεταγμένες των δεδομένων μας ως σημεία, βρίσκουμε τις κοντινότερες ακμές στα σημεία αυτά, με τη μέθοδο Ball Tree. Η μέθοδος Ball Tree βασίζεται σε ένα δυαδικό δέντρο στο οποίο κάθε κόμβος ορίζει μια σφαίρα που περιέχει ένα υποσύνολο των σημείων προς αναζήτηση. Κάθε εσωτερικός κόμβος του δέντρου χωρίζει τα σημεία δεδομένων σε δύο διαφορετικά σύνολα, που συνδέονται με διαφορετικές μπάλες. Ενώ οι μπάλες μπορούν να διασταυρωθούν μεταξύ τους, κάθε σημείο ανήκει στη μία ή στην άλλη μπάλα στο διαμέρισμα, ανάλογα με την απόσταση του από το κέντρο της μπάλας. Κάθε κόμβος φύλλων στο δέντρο ορίζει μια μπάλα και απαριθμεί όλα τα σημεία δεδομένων μέσα σε αυτή τη σφαίρα [7].

Ως συνέχεια των παραπάνω, βρίσκουμε τις συντεταγμένες για κάθε κορυφή της ακμής πάνω στην οποία κινείται η τροχιά, αλλά και το μήκος L της ακμής αυτής. Στην επόμενη ενότητα περιγράφεται πιο αναλυτικά η συνάρτηση της απόστασης αυτής αλλά και η χρησιμότητά της.

Υλοποίηση Αλγορίθμου Attribute Value

Όπως αναφέραμε και νωρίτερα, ο αλγόριθμος του Getis-Ord λαμβάνει υπόψη μια παράμετρο attribute value για κάθε ακμή j με βάρος $w_{i,j}$. Η επιλογή του attribute value έχει ως σκοπό την αντιμετώπιση τριών βασικών προβλημάτων:

1. Τα μήκη των ακμών που διαφέρουν μεταξύ τους.
2. Το όριο ταχύτητας της κάθε ακμής που μελετάμε.
3. Το πλήθος λωρίδων διαφέρει στις ακμές, για αυτό παίρνουμε τον μέσο όρο και όχι το άθροισμα των attribute value των τροχιών.

Για τις ανάγκες της παρούσας εργασίας, ορίσαμε τη παράμετρο του Attribute Value ως το όριο ταχύτητας της ακμής διά την μέση ταχύτητα του οχήματος εντός της ακμής, με τύπο:

$$x_i = \frac{V_{ref}}{(\frac{l}{t_{exit} - t_{in}})} \tag{4}$$

όπου,

$$t_{in} = \frac{d_{in}}{(\frac{t_2 - t_1}{d_{points}})} = \frac{d_{in}}{v_{points}} \tag{5}$$

Οι παράμετροι V_{ref}, l είναι ήδη γνωστές από τα προηγούμενα βήματα. Ο υπολογισμός των χρονικών ορίων (t_{in} και t_{exit}) έγινε με τη βοήθεια της βιβλιοθήκης **geopy** και της συνάρτησης **distance**. Η συνάρτηση αυτή υπολογίζει την Γεωδαισιακή Απόσταση (Geodesic Distance), δηλαδή το μήκος κάθε κοντινού μονοπατιού μεταξύ των δύο δοθέντων σημείων και βρίσκει εφαρμογή τόσο στον υπολογισμό του d_{in} όσο και στο d_{points} [8]. Έπειτα υπολογίζεται η διαφορά από τη μία χρονική στιγμή στην επόμενη (ΔΤ

LocalDate) και γίνεται η διαίρεση για τον υπολογισμό του t_{node}, τη χρονική στιγμή που η τροχιά πέρασε από τον εξεταζόμενο κόμβο. Αυτή η χρονική στιγμή αποθηκεύεται ως το t_{exit} της ακμής που καταλήγει στον κόμβο και ως t_{in}της ακμής που ξεκινάει από αυτόν τον κόμβο. Εφόσον έχει υπολογιστεί προηγουμένως το t_{in} για την ακμή που καταλήγει στον εξεταζόμενο κόμβο υπολογίζεται το attribute value με βάση το αποθηκευμένο t_{in} και το t_{exit} που υπολογίστηκε μόλις.

Algorithm Attribute Value

```
1: Input: Traj group, Rest Paramet
2: Output: tuple(Edge, AV)
3: function
4: FOR item in group
5:     previous = Point of last item
6:     current = Point of item
7:         IF different edge from previous point
8:             IF current Node1 is equal to previous one
9:                 node_time = common node timestamp
10:                t_bracket = diff(node_time, dataset_min_time) / step of time
11:                entry_edge_id = (current Node1, current Node2)
12:                exit_edge_id = (previous Node2, previous Node1)
13:                set entry_dict{entry_edge_id : node_time}
18:                IF entry_dict.get(exit_edge_id) exists
19:                    AV = Attribute Value Calculation
20:                    return (exit_edge_id, t_bracket), AV
21:                END IF
22:            END IF
23:        END IF
24: END FOR
25: end function
```

Υλοποίηση Αλγορίθμου Getis-Ord

Όπως αναφερθήκαμε και σε προηγούμενο κεφάλαιο, το στατιστικό G^* εφαρμόζεται με σκοπό την ανακάλυψη στατιστικά σημαντικών χωρο-χρονικών ακμών του γράφου που αντικατοπτρίζει ένα οδικό δίκτυο. Η συνάρτηση δίνεται από τον τύπο,

$$G^*_i = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{[n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2]}{n-1}}}$$

με τα $\bar{X}$ και S να δίνονται από τους τύπους:

$$\bar{X} = \frac{\sum_{j=1}^{n} x_i}{n}$$

$$S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\bar{X})^2}$$

Έχοντας υπολογίσει νωρίτερα, το Attribute Value της κάθε ακμής και έχοντας τα δεδομένα στην απαιτούμενη μορφή, μένει να ορίσουμε τα βάρη των ακμών και την ακτίνα των γειτόνων από την εξεταζόμενη ακμή. Για κάθε ακμή υπολογίζονται βάρη με βάση την απόσταση τους (στο γράφο) από την εξεταζόμενη ακμή. Η σχέση μεταξύ αυτών υποθέτουμε πως είναι εκθετική ($w_{ij} = \frac{1}{a}, \frac{1}{a^2}, \frac{1}{a^3}$ κλπ.). Για την ακτίνα (radius) που θα υπολογίζονται οι γείτονες έχουμε εκτελέσει πειράματα με τις τιμές 3, 10, 25 καθώς και με τις συνολικές ακμές κάθε γράφου, όπως παρουσιάζονται σε επόμενο κεφάλαιο.

Algorithm Getis-Ord

```
1: Input: u, v, AV, a, r
2: Output: u, v, G*
3: function
4: SET attmean = AV.mean, SET attstd = AV.std
5: Create Graph with Weights of edges
6: FOR each u, v in edges
7:     Create EgoGraph(Graph, v, r)
8:         IF number of edges at EgoGraph > 2
9:             SET attsum = 0
10:            SET weightsum = 0
11:            SET squaresum = 0
12:            FOR each ue, ve, AV in EgoGraph weighted edges
13:                CALC weight = neighbor degree / a
14:                SET attsum += AV*weight
15:                SET weightsum += weight
16:                SET squaresum += weight**2
17:            G = (attsum – attmean*weightsum) / (attstd*((n*squaresum-
           weightsum**2) / (n-1)) ** 0.5
18:            END FOR
19:        END IF
20: END FOR
21: end function
```

Αρχικά, δημιουργούμε ένα γράφο (Ego Graph) ο οποίος περιλαμβάνει το κύριο γράφο G, μία ακμή v και μια ακτίνα r και επιστρέφει έναν υπο-γράφο με τους γείτονες, που έχουν ως κέντρο την ακμή v και με ακτίνα r. Έπειτα, για κάθε ακμή υπολογίζονται βάρη με βάση την απόσταση τους (στο γράφο) από την εξεταζόμενη ακμή. Η σχέση μεταξύ αυτών υποθέτουμε πως είναι εκθετική και υπολογίζεται από το πηλίκο του αριθμού των ακμών που γειτνιάζουν με τον εξεταζόμενο κόμβο v προς τη σταθερά a. Τέλος, γίνει η εφαρμογή του αλγορίθμου, με το τελικό αποτέλεσμα να είναι ένα σύνολο από ακμές με τα αντίστοιχα z-score αποτελέσματά τους.

Υλοποίηση Spark Streaming

Λαμβάνοντας υπόψη όλα τα παραπάνω, καταλήγουμε στο τελευταίο στάδιο της υλοποίησης, που είναι η υλοποίηση της εφαρμογή μας στο εργαλείο μεγάλων δεδομένων Spark Streaming. Για τις ανάγκες τις παρούσας εργασίας, ο υπολογισμός του Getis- Ord, εκτελέστηκε σε διακριτά κομμάτια του Stream χρησιμοποιώντας τη τεχνική του micro-batching και όχι με τη τεχνική του sliding window. Η εφαρμογή δέχεται μια ροή δεδομένων (DStream), τη διαχωρίζει σε τμήματα (batches) των 10 δευτερολέπτων, στα οποία εφαρμόζονται οι πράξεις ή λειτουργίες που του ορίζουμε να κάνει και στο τέλος να παράγει το απαιτούμενο αποτέλεσμα με τη μορφή DStream. Για να μπορέσουμε να εφαρμόσουμε τους παραπάνω αλγορίθμους, στο συγκεκριμένου εργαλείο, απαιτούνται κάποιες τροποποιήσεις ως προς τη μορφή των δεδομένων μας. Η μορφή με την οποία διαχειρίζεται το Spark τα δεδομένα, είναι με της μορφής (key, value) ζευγαριών. Πάνω σε αυτά τα ζευγάρια γίνονται οι όποιοι μετασχηματισμοί και πράξεις χρειάζονται ώστε να παραχθούν τα επιθυμητά αποτελέσματα.

Το σύνολο των δεδομένων που πρέπει να έχουμε, πριν εισαχθούν στο Spark, πρέπει να αποτελείται από τις στήλες Vehicle, Traj_ID, LocalDate, Latitude, Longitude, Road_Speed, Length, Node1, Node1_Lat, Node1_Lon, Node2, Node2_Lat, Node2_Lon.

Spark Streaming Implementation

1: **Input**: DStream of Data

2: **Output**: Calculated Batches of Data

3: **function**

4: DStream → DStream[RDD1, RDD2,…]

.create_pair_keys()

.groupByKey()

.mapValues(sort_keys_by_LocalDate)

.flatMap(AttributeValueAlgorithm)

.transform(aggregateByKey().mean())

5: FOR each RDD in DStream

6: dataRDD → RDD[Traj_ID, rest..]

.collect()

.createGraph(parameters)

.flatMap(Getis-Ord Algorithm)

7: END FOR

8: **end function**

Πρώτο μας μέλημα είναι να φέρουμε τα δεδομένα μας σε μορφή (key, value) ζευγαριών. Καθώς η μελέτη μας αφορά μελέτη τροχιών σε οδικό δίκτυο, ως κλειδί ορίζουμε το Trajectory ID και ως αξίες όλες τις υπόλοιπες στήλες. Με τον τρόπο αυτό θα μπορέσουμε να ομαδοποιήσουμε τα δεδομένα μας με βάση το αναγνωριστικό τροχιάς κι έπειτα να τα ταξινομήσουμε χρονικά, με τη στήλη LocalDate. Στο σημείο αυτό έχουμε φέρει τα δεδομένα μας στη μορφή που χρειάζεται ώστε να εφαρμόσουμε τον αλγόριθμο του Attribute Value και να υπολογίσουμε για κάθε ακμή τη τιμή του. Για τον υπολογισμό αυτό, χρησιμοποιούμε τη τεχνική του Secondary Sort που έχουμε αναφέρει σε

προηγούμενο κεφάλαιο, ώστε το αποτέλεσμα να είναι ένα RDD με κλειδί τις κορυφές της ακμής που μελετάμε και η αξία να είναι η τιμή του Attribute Value.

Στη συνέχεια, όπως αναφέρεται στο κεφάλαιο 4, βρίσκουμε το μέσο όρο των τιμών των τροχιών εντός της ακμής, αθροίζοντας τις τιμές των Attribute Value που βρήκαμε παραπάνω, με τη συχνότητα που εμφανίζονται (sum/count).

Τέλος, χρειάζεται να συγκεντρώσουμε όλα τα παραπάνω δεδομένα από το RDD, με τη μέθοδο του collect, ώστε να δημιουργήσουμε ένα γράφο με τα χαρακτηριστικά που εξηγήσαμε στη προηγούμενη ενότητα, για να μπορέσουμε τελικά να εφαρμόσουμε τον αλγόριθμο του Getis-Ord.

Κεφάλαιο 6: Πειραματική Μελέτη

Στην ενότητα θα παρουσιάσουμε το βαθμό απόδοσης της προσέγγισής για την ανακάλυψη Hot Spot στο οδικό δίκτυο. Η προεπεξεργασία των δεδομένων μας, μέχρι να καταλήξουν στην επιθυμητή τους μορφή, έγινε με τη χρήση της γλώσσας προγραμματισμού Python, έκδοσης 3.7.3. Για τις βασικές μας υλοποιήσεις των αλγορίθμων του Attribute Value και του Getis- Ord, χρησιμοποιήθηκε το εργαλείο Spark Python API την έκδοση 2.4.3. Η ανάπτυξη και εκτέλεση του κώδικα, έγινε σε τοπικό μηχάνημα, με τα χαρακτηριστικά του να είναι 2 cores, 4 threads, 12GB RAM και 1.5TB χώρο αποθήκευσης.

Τα δεδομένα με τα οποία πειραματιστήκαμε αφορούν σήματα αυτοκινήτων από GPS. Τα σύνολο των δεδομένων ήταν σε μορφή csv με διαχωριστή semicolon (;). Το πλήθος των δεδομένων ήταν συνολικά 1.789.992 γραμμές, χωρισμένα σε 1950 αρχεία, με κατά μέσο όρο 917 γραμμές το καθένα. Οι στήλες που το αποτελούν είναι 75 . Από αυτές επιλέξαμε να αξιοποιήσουμε το αναγνωριστικό του οχήματος (Vehicle), το αναγνωριστικό της τροχιάς (Traj_ID), την ημερομηνία και ώρα που πέρασαν αυτές από το σημείο (LocalDate), τις συντεταγμένες των σημείων των τροχιών (Longitude, Latitude) καθώς επίσης και το όριο ταχύτητας του δρόμου (Road_Speed) στον οποίο κινούνται.

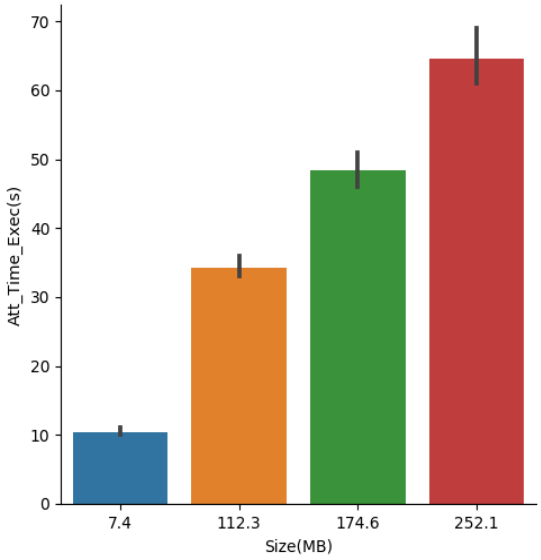
Το βασικότερο μέτρο αποτίμησης των αποτελεσμάτων ήταν ο χρόνος εκτέλεσης της υλοποίησης μέσω του Spark Streaming, η οποία χωρίστηκε σε δύο φάσης. Η πρώτη φάση είναι από τη στιγμή που διαβάζει τα δεδομένα το πρόγραμμα μέχρι και την εκτέλεση του αλγορίθμου Attribute Value και η δεύτερη ξεκινά από τη στιγμή που τελειώνει ο πρώτος αλγόριθμος μέχρι τη στιγμή που τελειώνει ο αλγόριθμος του Getis-Ord.

Για να μπορέσουμε να τροφοδοτήσουμε το εργαλείο Spark Streaming, ώστε να δέχεται μια ροή δεδομένων, χωρίσαμε το σύνολο των δεδομένων μας σε υποσύνολα. Κάθε ένα υποσύνολο, έχει διαφορετικό μέγεθος (batch size), ώστε να μπορούμε να πειραματιστούμε με βάση το χρόνο εκτέλεσης ανά το μέγεθος του υποσυνόλου. Επίσης, σε κάθε ένα από αυτά τα υποσύνολα, εκτελέστηκε ο αλγόριθμος του Getis- Ord με τιμές 3, 10, 25 και με το συνολικό αριθμό γειτονικών ακμών, για τη παράμετρο ακτίνας r.

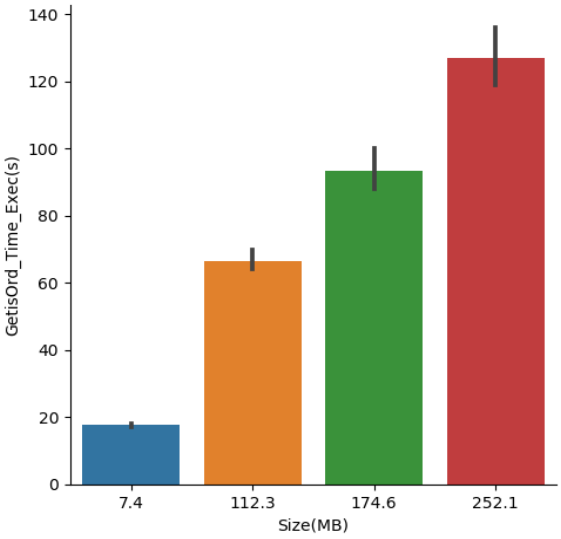
Παράμετροι	**Τιμές**			
Batch Size	7.4 (MB) 48.299 (rows)	112.3 (MB) 734.574 (rows)	174.6 (MB) 1.141.428 (rows)	252.1 (MB) 1.650.598 (rows)
Radius r	3	10	25	*all

Πίνακας 1 Παράμετροι Πειραματισμού

Εκτελώντας τα πειράματά μας βάση του Πίνακα 1, προκύπτουν τα αποτελέσματα των διαγραμμάτων που ακολουθούν.

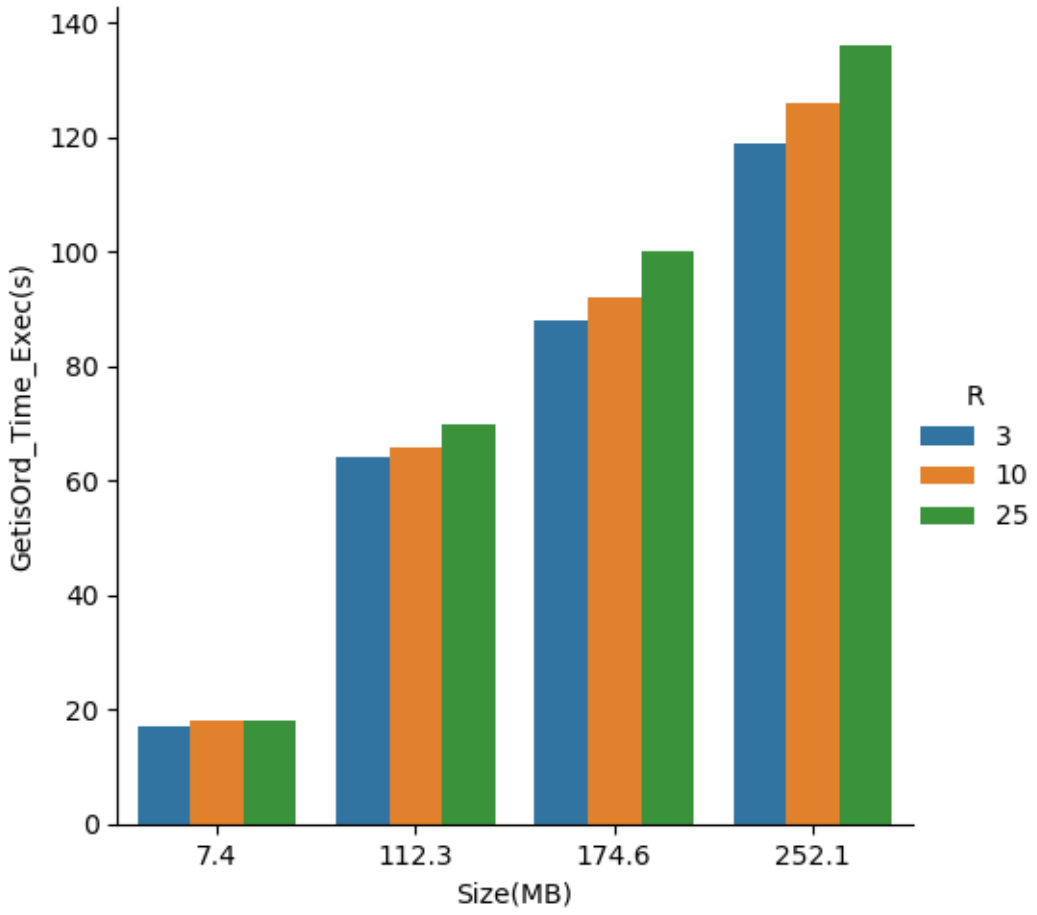


Διάγραμμα 2 Attribute Value Algorithm Execution Time VS Batch Size



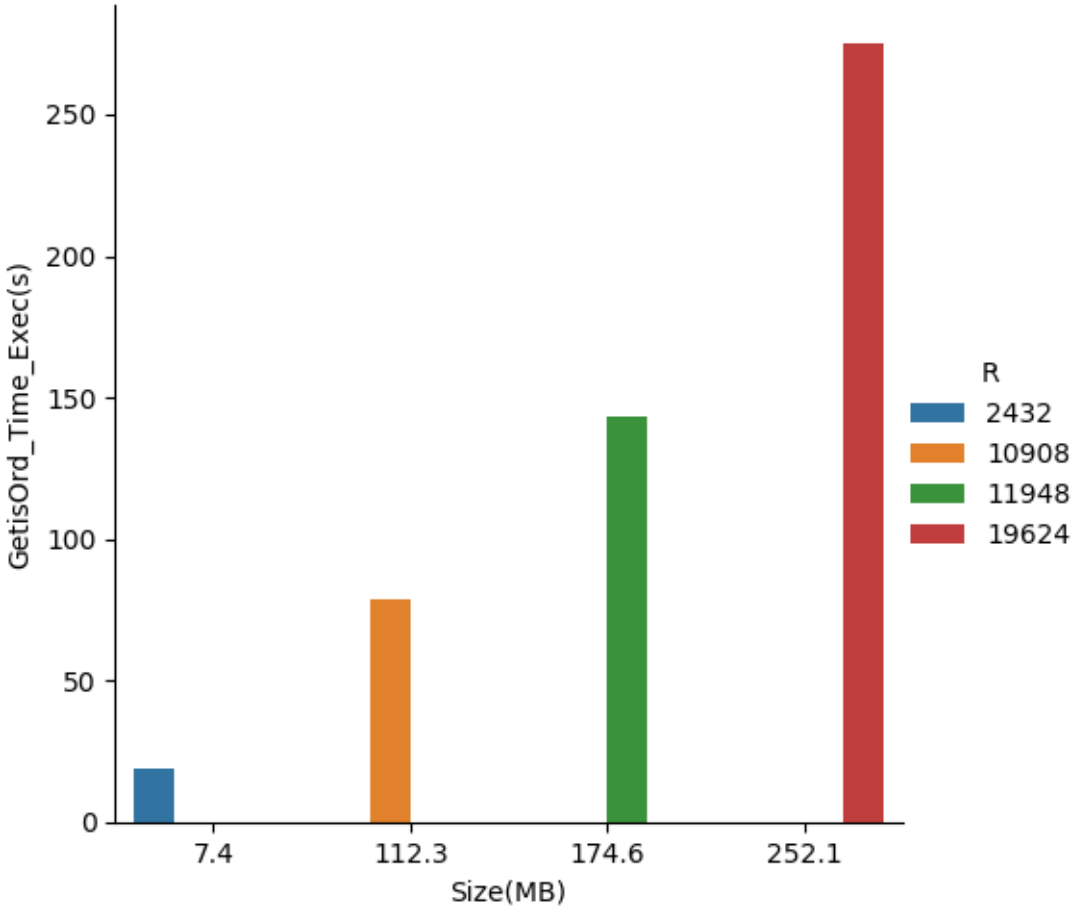
Διάγραμμα 3 Getis- Ord Algorithm Execution Time VS Batch Size

Στα Διαγράμματα 2 και 3, απεικονίζεται η σχέση του χρόνου εκτέλεσης των αλγορίθμων του Attribute Value και του Getis- Ord, σε σχέση με το batch size. Είναι εμφανές ότι όσο μεγαλύτερο είναι το μέγεθος του αρχείου που εφαρμόζεται ο αλγόριθμος, τόσο αυξάνεται και ο χρόνος εκτέλεσής του.



Διάγραμμα 4 Getis- Ord Algorithm Execution Time VS Batch Size over Radius r with values 3,10,25

Στο Διάγραμμα 4, απεικονίζει το χρόνο εκτέλεσης του αλγορίθμου Getis- Ord σε συνάρτηση με το μέγεθος του αρχείου που λαμβάνει ως είσοδο, για κάθε εκτέλεση με τη παράμετρο της ακτίνας r να παίρνει τις τιμές 3, 10, 25. Όπως και στη προηγούμενη περίπτωση, παρατηρούμε ότι όσο αυξάνεται το batch size, τόσο αυξάνεται κι ο χρόνος εκτέλεσης του αλγορίθμου. Παρόλα αυτά, για τις περιπτώσεις που το batch size είναι σχετικά μικρό, 7.4 MB και 112.3 MB, οι διαφορές μεταξύ των χρόνων εκτέλεσης για τις διαφορετικές τιμές του r δεν είναι ιδιαίτερα μεγάλες. Χαρακτηριστικά για το αρχείο με batch size 112.3 MB για r = 3 και r = 10, η διαφορά στο χρόνο εκτέλεσης ανέρχεται στα 2 δευτερόλεπτα, ενώ για r = 10 και r = 25 στα 8 δευτερόλεπτα.



Διάγραμμα 5 Getis- Ord Algorithm Execution Time VS Batch Size over Radius r of all edges

Στο Διάγραμμα 5, απεικονίζεται ο χρόνος εκτέλεσης του αλγορίθμου Getis – Ord σε συνάρτηση με το batch size, με τη παράμετρο r να λαμβάνει τις μέγιστες τιμές των γειτονικών ακμών για κάθε αρχείο. Σε αυτή τη περίπτωση είναι ξεκάθαρο πως όσο μεγαλύτερη είναι η ακτίνα των γειτόνων που λαμβάνει υπόψη του ο αλγόριθμος, τόσο μεγαλύτερος είναι και χρόνος εκτέλεσής του.

Κεφάλαιο 7: Επίλογος

Σύνοψη και Συμπεράσματα

Στη συγκεκριμένη εργασία αναπτύχθηκε ένα πρόγραμμα εύρεσης Hot Spot σε οδικό δίκτυο. Για να καταστεί αυτό εφικτό, έγινε χρήση της γλώσσας προγραμματισμού Python και του εργαλείου μεγάλων δεδομένων Spark. Επιπλέον υλοποιήθηκαν οι αλγόριθμοι Getis-Ord και Attribute Value, μέσω των οποίων υπολογίσαμε το στατιστικό Gi* για κάθε ακμή του δοθέντος συνόλου δεδομένων. Τα αποτελέσματα της υλοποίησης κρίνονται με βάση το χρόνο εκτέλεσης κάθε αλγορίθμου, για κάθε μέγεθος αρχείου που χρησιμοποιήθηκε. Τέλος, πειραματιστήκαμε και με τη παράμετρο του αλγορίθμου Getis-Ord για να δούμε πόσο επηρεάζει το χρόνο εκτέλεσης.

Μελλοντικές Επεκτάσεις

Στη παρούσα εργασία προσπαθήσαμε να βρούμε Hos Spot σημεία στο οδικό δίκτυο με τη χρήση του εργαλείου Spark Streaming. Η υλοποίηση έγινε σε περιβάλλον Windows 10 και με τη χρήση ενός μηχανήματος με μνήμη RAM των 12 GB, χωρίς την υποστήριξη cluster. Έτσι όλες οι λειτουργίες έγιναν από το εν λόγω μηχάνημα πράγμα που το καθιστά αρκετά αργό και περιορίζει τις δυνατότητες του εργαλείου. Με τη χρήση ενός cluster οι υπολογισμοί θα γίνονται σε πολύ λιγότερο χρόνο και τα δεδομένα εισαγωγής θα μπορούν να είναι πολύ περισσότερα, εμβαθύνοντας περισσότερο ως προς τους πειραματισμούς και τα αποτελέσματα. Επίσης καθότι η υλοποίηση είναι σε Steaming μορφή, μπορεί να προσαρμοστεί εύκολα σε δεδομένα που παράγονται live από κάποιο API, χρησιμοποιώντας έτσι και τη μέθοδο του Sliding Window.

Βιβλιογραφία

[1] Xun Tang, Emre Eftelioglu, Dev Oliver, Shashi Shekhar: Significant Linear Hotspot Discovery. IEEE Trans. Big Data 3(2): 140-153 (2017)

[2] David M. Blei, Andrew Y. Ng, Michael I. Jordan: Latent Dirichlet Allocation. J. Mach. Learn. Res. 3: 993-1022 (2003)

[3] Nikolaos Zygouras, Dimitrios Gunopulos: Discovering Corridors From GPS Trajectories. SIGSPATIAL/GIS 2017: 61:1-61:4

[4] Benjamin B. Krogh, Ove Andersen, Kristian Torp: Trajectories for novel and detailed traffic information. IWGS@SIGSPATIAL/GIS 2012: 32-39

[5] Lee Mendelowitz: Algorithms for the Alignment and Visualization of genome Mapping Data with Applications to Structural variant Detection. University of Maryland, College Park, MD, USA, 2015

[6] Sergey Kitaev: A Comprehensive Introduction to the Theory of Word-Representable Graphs. DLT 2017: 36-67

[7] Ting Liu, Andrew W. Moore, Alexander G. Gray: New Algorithms for Efficient High-Dimensional Nonparametric Classification. J. Mach. Learn. Res. 7: 1135-1158 (2006)

[8] C. F. F. Karney, «Algorithms for geodesics,» *Journal of Geodesy,* pp. 46-55, 2013.

[9] Panagiotis Nikitopoulos, Aris-Iakovos Paraskevopoulos, Christos Doulkeridis, Nikos Pelekis, Yannis Theodoridis: Hot Spot Analysis over Big Trajectory Data. BigData 2018: 761-770

[10] Geoff Boeing:OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Comput. Environ. Urban Syst. 65: 126-139 (2017)

[11] Fatima Zahra El Mazouri, Mohammed Chaouki Abounaima, Khalid Zenkouar: Data mining combined to the multicriteria decision analysis for the improvement of road safety: case of France. J. Big Data 6: 5 (2019)

[12] Diansheng Guo, Jeremy L. Mennis: Erratum to "Spatial data mining and geographic knowledge discovery - An introduction" [Computers, Environment and Urban Systems 33 (2009) 403-408]. Comput. Environ. Urban Syst. 34(2): 175 (2010)

[13] Maik Häsner, Conny Junghans, Christian Sengstock, Michael Gertz: Online Hot Spot Prediction in Road Networks. BTW 2011: 187-206

[14] Shashi Shekhar, Zhe Jiang, Reem Y. Ali, Emre Eftelioglu, Xun Tang, Venkata M. V. Gunturi, Xun Zhou: Spatiotemporal Data Mining: A Computational Perspective. ISPRS Int. J. Geo-Information 4(4): 2306-2338 (2015)

[15] Zhe Jiang, Arpan Man Sainju, Yan Li, Shashi Shekhar, Joseph K. Knight: Spatial Ensemble Learning for Heterogeneous Geographic Data with Class Ambiguity. ACM TIST 10(4): 43:1-43:25 (2019)

[16] Shashi Shekhar, Zhe Jiang, James M. Kang, Vijay Gandhi:Spatial Data Mining. Encyclopedia of Database Systems (2nd ed.) 2018

[17] Aanchal Mongia, Venkata M. V. Gunturi, Vinayak Naik: Detecting activities at metro stations using smartphone sensors. COMSNETS 2018: 57-65

[18] Venkata M. V. Gunturi, Shashi Shekhar: Spatio-Temporal Graph Data Analytics. Springer 2017, ISBN 978-3-319-67770-5, pp. 1-100

Παράρτημα

Υποδομή Υλοποίησης

Για την υλοποίηση της παρούσας εργασίας χρησιμοποιήθηκαν:

- Microsoft Windows 10 Pro
- Python v.3.7 με τις κύριες βιβλιοθήκες που χρησιμοποιήθηκαν:
  - Numpy
  - NetworkX
  - Geopy
  - pyspark
- Anaconda Spyder v.4.8.2
- Apache Spark v.2.4.3
- Apache Hadoop v.2.7
- Oracle Java v.1.8.0_161
- Scala v.2.13.1

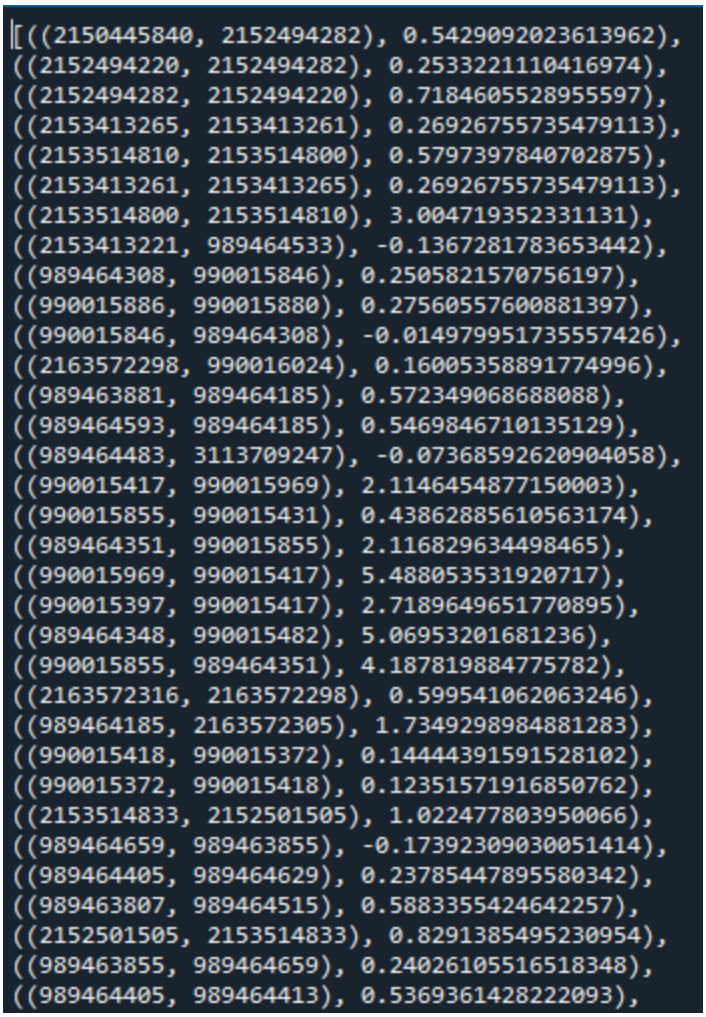
Μορφή Δεδομένων

```
Vehicle,Traj_ID,LocalDate,Latitude,Longitude,Road_Speed,Length,Node1,Node1_Lat,Node1_Lon,Node2,Node2_Lat,Node2_Lon
3130_36240,,2018-07-02 08:34:56.6200000,37.965471999999998,23.770309000000001,50,,,,,,
3130_36240,,2018-07-02 08:34:57.7500000,37.965488000000001,23.770247999999999,50,,,,,,
3130_36240,,2018-07-02 08:34:59.2970000,37.965507000000002,23.770184,50,,,,,,
3130_36240,,2018-07-02 08:35:01.0100000,37.965513999999999,23.770160000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:04.4000000,37.965549000000003,23.770015999999998,50,,,,,,
3130_36240,,2018-07-02 08:35:06.4700000,37.965584,23.769960000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:08.6000000,37.965642000000003,23.769908999999998,50,,,,,,
3130_36240,,2018-07-02 08:35:09.6700000,37.965685999999998,23.769880000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:11.8100000,37.965744000000001,23.769846000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:14.9400000,37.965817999999999,23.769801999999999,50,,,,,,
3130_36240,,2018-07-02 08:35:16.0100000,37.965850000000003,23.769787000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:17.0800000,37.965877999999996,23.769770000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:19.2100000,37.965919999999997,23.769746000000001,50,,,,,,
3130_36240,,2018-07-02 08:35:20.2800000,37.965935999999999,23.769738,50,,,,,,
3130_36240,,2018-07-02 08:35:25.4800000,37.965964999999997,23.769736000000002,50,,,,,,
3130_36240,,2018-07-02 08:35:26.6100000,37.966003000000001,23.769757999999999,50,,,,,,
3130_36240,,2018-07-02 08:35:27.7400000,37.966064000000003,23.769791999999999,50,,,,,,
3130_36240,,2018-07-02 08:35:29.8800000,37.966217999999998,23.769826999999999,50,,,,,,
3130_36240,,2018-07-02 08:35:31.0100000,37.966275000000003,23.769832000000001,50,,,,,,
```

Εικόνα 6 : Raw Data

```
Vehicle,Traj_ID,LocalDate,Latitude,Longitude,Road_Speed,Length,Node1,Node1_Lat,Node1_Lon,Node2,Node2_Lat,Node2_Lon
3130_36240,3130_36240-1,2018-07-02 08:34:56.620,37.965472,23.770309,50,0.056861402622604l2,359623576,37.9654465,23.7706487,359623591,37.9655722,23.7700214
3130_36240,3130_36240-1,2018-07-02 08:34:57.750,37.965488,23.770248,50,0.056861402622604l2,359623591,37.9655722,23.7700214,359623576,37.9654465,23.7706487
3130_36240,3130_36240-1,2018-07-02 08:34:59.297,37.965507,23.770184,50,0.056861402622604l2,359623591,37.9655722,23.7700214,359623576,37.9654465,23.7706487
3130_36240,3130_36240-1,2018-07-02 08:35:01.010,37.965514,23.77016,50,0.056861402622604l2,359623591,37.9655722,23.7700214,359623576,37.9654465,23.7706487
3130_36240,3130_36240-1,2018-07-02 08:35:04.400,37.965549,23.770016,50,0.050884588831049S4,359623591,37.9655722,23.7700214,359623373,37.965357,23.7695101
3130_36240,3130_36240-1,2018-07-02 08:35:06.470,37.965584,23.76996,50,0.050884588831049S4,359623591,37.9655722,23.7700214,359623373,37.965357,23.7695101
3130_36240,3130_36240-1,2018-07-02 08:35:08.600,37.965642,23.769909,50,0.037231947722920S4,359623608,37.9658665,23.7698181,359623591,37.9655722,23.7700214
3130_36240,3130_36240-1,2018-07-02 08:35:09.670,37.965686,23.76988,50,0.037231947722920S4,359623608,37.9658665,23.7698181,359623591,37.9655722,23.7700214
3130_36240,3130_36240-1,2018-07-02 08:35:11.810,37.965744,23.769846,50,0.037231947722920S4,359623608,37.9658665,23.7698181,359623591,37.9655722,23.7700214
3130_36240,3130_36240-1,2018-07-02 08:35:14.940,37.965818,23.769802,50,0.016872392744014284,359623538,37.9660038,23.7697357,359623608,37.9658665,23.7698181
3130_36240,3130_36240-1,2018-07-02 08:35:16.010,37.96585,23.769787,50,0.016872392744014284,359623538,37.9660038,23.7697357,359623608,37.9658665,23.7698181
3130_36240,3130_36240-1,2018-07-02 08:35:17.080,37.965878,23.76977,50,0.016872392744014284,359623538,37.9660038,23.7697357,359623608,37.9658665,23.7698181
3130_36240,3130_36240-1,2018-07-02 08:35:19.210,37.96592,23.769746,50,0.016872392744014284,359623538,37.9660038,23.7697357,359623608,37.9658665,23.7698181
3130_36240,3130_36240-1,2018-07-02 08:35:20.280,37.965936,23.769738,50,0.016872392744014284,359623538,37.9660038,23.7697357,359623608,37.9658665,23.7698181
3130_36240,3130_36240-1,2018-07-02 08:35:25.480,37.965965,23.769736,50,0.005902121407312365,359623538,37.9660038,23.7697357,952669314,37.9660461,23.7697764
3130_36240,3130_36240-1,2018-07-02 08:35:26.610,37.966003,23.769758,50,0.005902121407312365,359623538,37.9660038,23.7697357,952669314,37.9660461,23.7697764
3130_36240,3130_36240-1,2018-07-02 08:35:27.740,37.966064,23.769792,50,0.005902121407312365,952669314,37.9660461,23.7697764,359623538,37.9660038,23.7697357
3130_36240,3130_36240-1,2018-07-02 08:35:29.880,37.966218,23.769827,50,0.054013030083723l8,952669314,37.9660461,23.7697764,359620088,37.9665235,23.7698955
3130_36240,3130_36240-1,2018-07-02 08:35:31.010,37.966275,23.769832,50,0.054013030083723l8,952669314,37.9660461,23.7697764,359620088,37.9665235,23.7698955
```

Εικόνα 7 : Preprocessed Data



Εικόνα 8 Result Data

Εκτέλεση Spark Streaming εφαρμογής

Για την υλοποίηση της παρούσας εργασίας, δε χρησιμοποιήθηκε κάποιο API που να διαβάζει δεδομένα live, αλλά ένα batch σύνολο δεδομένων. Έτσι, για να μπορέσουμε να εκτελέσουμε τη Streaming εφαρμογή, δημιουργήσαμε ένα script που χωρίζει το αρχείο δεδομένων μας σε υπό- αρχεία ώστε να μπορεί η εφαρμογή να κοιτάζει ένα directory που συνέχεια εμπλουτίζεται με δεδομένα. Για την εκτέλεση της εφαρμογής χρειάζονται δύο γραμμές εντολών, εκτελώντας τις εντολές όπως περιγράφονται παρακάτω.

cmd1: Εκκίνηση εφαρμογής Spark Streaming

spark-submit “Script Directory”\hot_spot_v03.py “Read_Streaming_Files Directory”

cmd2: Δημιουργία αρχείων που θα διαβάζει η εφαρμογή

cd “Script Directory”

python create_logs.py

Αρχικά εκτελούμε την cmd1 ώστε να εκκινήσει την εφαρμογή του Spark Streaming και στη συνέχεια τη cmd2 που θα αρχίσει να δημιουργεί υπό – αρχεία από τα οποία θα διαβάζει η εφαρμογή.