Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Προστασία Ιδιωτικότητας Σημασιολογικών Δεδομένων Κίνησης με χρήση Μηχανισμού-Ελέγχου Ερωτημάτων** Privacy Preservation of Semantic Trajectory Data with Query Auditing-Mechanism
Ονοματεπώνυμο Φοιτητή	**Χρήστος Σκορδάς**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΣΠ/** 14081
Επιβλέπων	**Νικόλαος Πελέκης**, **Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Μάρτιος** 2018

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ιωάννης Θεοδωρίδης Καθηγητής	Νικόλαος Πελέκης Επίκουρος Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής

Ευχαριστίες

Η παρούσα εργασία αποτελεί διπλωματική εργασία στο πλαίσιο του μεταπτυχιακού προγράμματος «Προηγμένα Συστήματα Πληροφορικής» του τμήματος Πληροφορικής, του πανεπιστημίου Πειραιώς. Ταυτόχρονα σηματοδοτεί το τέλος σε μια διαδρομή γνώσης και διεύρυνσης του νου η οποία με επηρέασε βαθύτατα.

Αισθάνομαι την υποχρέωση να εκφράσω τις θερμές μου ευχαριστίες στους καθηγητές και τους επιβλέποντές μου Ιωάννη Θεοδωρίδη, Νικόλαο Πελέκη και Δέσποινα Κοπανάκη για την πολύτιμη γνώση που έλαβα από τα μαθήματά τους, την βοήθεια, την καθοδήγηση και τις συμβουλές τους κατά τη διάρκεια της εκπόνησης αυτής της εργασίας.

Κρίνω επίσης απαραίτητο να εκφράσω την εκ των προτέρων εκτίμησή μου προς τον επίκουρο καθηγητή Άγγελο Πικράκη που αποτελεί μέλος της επιτροπής για αυτή την εργασία.

Θα ήθελα στο σημείο αυτό να ευχαριστήσω ιδιαίτερα τον φίλο και καθοδηγητή μου στον κόσμο της πληροφορικής Αλέξανδρο Γιαβάρα για την πολύτιμη βοήθειά του, την υπομονή, την υποστήριξή του από την αρχή της εργασίας μου αλλά και τις γνώσεις που μου προσέφερε διαρκώς από την μέρα που τον γνώρισα.

Ένα μεγάλο ευχαριστώ μέσα από την καρδιά μου ανήκει στην Στελλίνα Μπάστα που μοιράστηκε τον ενθουσιασμό, τις χαρές αλλά και τις λύπες και απογοητεύσεις που μου επιφύλασσε η διαδικασία συγγραφής και υλοποίησης αυτής της εργασίας. Τα λόγια δεν μπορούν να περιγράψουν το πόσο σημαντική ήταν η στήριξη και συμπαράστασή της.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για την ηθική στήριξη και κατανόησή της σε όλη τη διαδρομή μου αυτή μέχρι και το τέλος της παρούσας εργασίας.

Χρωστάω ένα μεγάλο ειλικρινές ευχαριστώ σε όλους σας.

Περίληψη

Η χρήση κινητών τηλεφώνων (*smartphones*) και πολλών ακόμα συσκευών που υποστηρίζουν συστήματα και λογισμικά εντοπισμού θέσης είναι πλέον αναπόσπαστο κομμάτι της καθημερινότητάς μας. Η αύξηση της χρήσης και εφαρμογής τους είναι ραγδαία με συνέπεια να παρουσιάζεται ένας τεράστιος και διαρκώς αυξανόμενος όγκος δεδομένων που καταγράφονται ως τροχιές κινούμενων αντικειμένων. Η εκμετάλλευση των δεδομένων κίνησης που καταγράφονται, σε συνδυασμό με την άνθηση των σύγχρονων τεχνικών εξόρυξης γνώσης, ήταν φυσικό επακόλουθο τόσο από τους επιστήμονες στην διαδικασία ερευνών όσο και από τους οργανισμούς παρόχους για επιχειρηματικούς λόγους κυρίως. Μαζί με τις καινοτομίες και τη γνώση όμως που προσφέρει η αποθήκευση και ανάλυση χωροχρονικών δεδομένων, εισήγαγε και την επικινδυνότητα για την αποκάλυψη ιδιωτικών πληροφοριών της εκάστοτε καταγραφόμενης οντότητας από κακόβουλους χρήστες. Αυτό είχε ως συνέπεια την ανάπτυξη θεωριών και τεχνικών με στόχο την ανωνυμοποίηση των δεδομένων και την προστασία της ιδιωτικότητας.

Η παρούσα διατριβή αφορά βάσεις δεδομένων που αποθηκεύουν σημασιολογικά εμπλουτισμένες τροχιές οι οποίες παρέχουν περισσότερη πληροφορία και παραμένουν αποθηκευμένες στον οργανισμό που τις δημιούργησε και τις κατέγραψε. Βασιζόμενοι σε μελέτες και εργασίες που αφορούν απλές τροχιές αναπτύσσουμε έναν μηχανισμό ελέγχου ερωτημάτων προς τη βάση, με σκοπό να αποφεύγουμε την παραβίαση της ιδιωτικότητας αλλά ταυτόχρονα να υφίσταται η μικρότερη δυνατή αλλοίωση στα στοιχεία που πρέπει να επιστρέψουν ως απάντηση σε κάθε τέτοιο ερώτημα. Υλοποιούμε έναν νέο αλγόριθμο, ονομαζόμενο *Zoom_Out* [7] με την χρήση του οποίου μπορεί να τροποποιηθεί το αρχικό ερώτημα όταν δεν επιτρέπεται να απαντηθεί για λόγους ασφαλείας. Απώτερος στόχος του αλγορίθμου είναι να μετατρέψει το αρχικό ερώτημα στο πλησιέστερο δυνατό που μπορεί να απαντηθεί χωρίς να υπάρχει κίνδυνος παραβίασης της ιδιωτικότητας. Με αυτό τον τρόπο ο μηχανισμός γίνεται πιο ελαστικός και φιλικός προς τον χρήστη και ενισχύεται η λειτουργικότητά του.

Abstract

The use of smartphones and many other devices that support positioning systems and software is an integral part of our everyday life. The increase in their use and application is rapid, resulting in a huge and ever increasing volume of data recorded as moving objects trajectories. The exploitation of recorded traffic data, combined with the modern data mining techniques flourishing, was a natural consequence from both the researchers in the survey process and the providers' organizations mostly for business reasons. However storage and analysis of geospatial data not only offers innovations and knowledge, but also introduced the risk of disclosing private information of the recorded entities to malicious users. This fact has resulted the development of theories and techniques with the aim of anonymizing the data and offering privacy protection.

This thesis deals with databases containing semantically enriched trajectories that provide more information and remain stored in-house to the organization that created and recorded them. Based on studies and researches on simple trajectories, we develop a query auditing mechanism to avoid privacy violations, but at the same time suffer the least possible alteration in the data that must return as an answer to any such query. We are implementing a new algorithm, called *Zoom_Out* [7], which can be used to modify the original query when we are not allowed to give an answer for security reasons. The ultimate goal of the algorithm is to convert the original query to the closest possible query that cannot be compromised by privacy. In this way the mechanism becomes more flexible and user-friendly and enhances its functionality.

Περιεχόμενα

1 Εισαγωγή

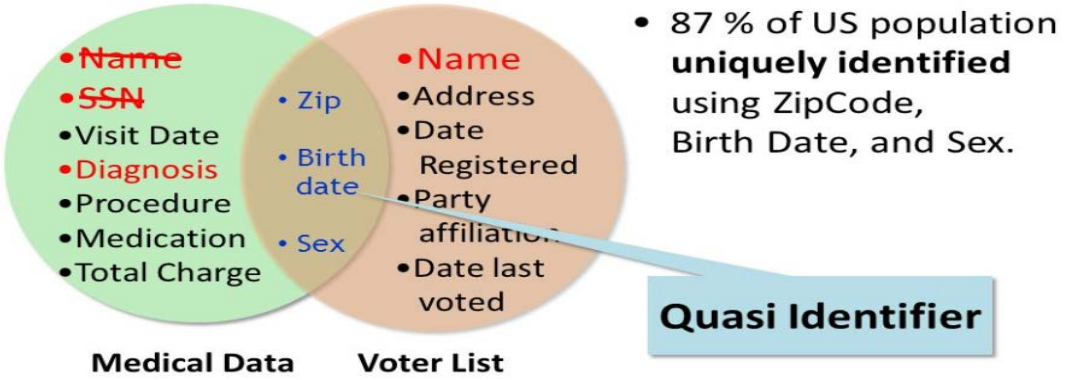
Η αυξανόμενη χρήση κινητών συσκευών τελευταίας τεχνολογίας και γενικότερα η διείσδυση στην καθημερινότητα του ανθρώπου των συσκευών που υποστηρίζουν συστήματα εντοπισμού θέσης (π.χ. GPS), έχει ως αποτέλεσμα τη συνεχή αύξηση ενός τεράστιου όγκου δεδομένων που εμφανίζονται με τη μορφή τροχιών. Η επιστημονική κοινότητα καθώς και οι οργανισμοί, πάροχοι διαφόρων υπηρεσιών οδηγήθηκαν στη συλλογή, αποθήκευση και επεξεργασία των χωροχρονικών δεδομένων που αποτυπώνονται ως ίχνη κινούμενων αντικειμένων, με σκοπό την δημιουργία καινοτόμων εφαρμογών που χρησιμοποιούν τεχνικές εξόρυξης γνώσης καθώς και τη βελτίωση παροχής των υπηρεσιών τους. Τα χωροχρονικά δεδομένα κάθε καταγραφόμενης οντότητας αποτελούν ένα πολύ χρήσιμο αλλά ταυτόχρονα και ευαίσθητο σύνολο πληροφοριών. Μπορεί λοιπόν για παράδειγμα στον τομέα της βιώσιμης διαχείρισης της κινητικότητας η χρήση αυτών των δεδομένων, για την εξαγωγή προτύπων συμπεριφοράς να είναι ένα κομβικό σκαλοπάτι, η αποκάλυψη και ανάλυσή τους όμως αυξάνει το ρίσκο παραβατικών συμπεριφορών διακινδυνεύοντας την ιδιωτικότητα των χρηστών, των οποίων οι κινήσεις έχουν καταγραφεί. Η διατήρηση της ιδιωτικότητας απασχόλησε τους επιστημονικούς κύκλους έντονα και τους ώθησε σε ανάπτυξη πολλών θεωριών και τεχνικών με στόχο την ανωνυμοποίηση των δεδομένων. Το ζήτημα της προστασίας της ιδιωτικότητας ωστόσο, έγινε ακόμα πιο απαιτητικό κατά την μετάβαση από τα σχεσιακά δεδομένα και τις απλές τροχιές σε ένα νέο τύπο δεδομένων τροχιών, τις *σημασιολογικά επαυξημένες τροχιές* (*semantic trajectories*) [15] οι οποίες παρέχουν περισσότερες πληροφορίες και αναλυτικότερη καταγραφή της τροχιάς.

Είναι κοινώς αποδεκτό ότι η χρήση δεδομένων τροχιών για την οποιαδήποτε υπηρεσία ή εξόρυξη γνώσης πρέπει να γίνεται μόνο έπειτα από την επεξεργασία και ανωνυμοποίηση τους. Αρκετές μεθοδολογίες αναπτύχθηκαν με κύριο άξονα τη δημιουργία και δημοσίευση ενός προστατευμένου συνόλου δεδομένων. Στις περιπτώσεις αυτές η αρχική βάση δεδομένων τροποποιείται ώστε να προστατευτεί η ιδιωτική ζωή των καταγεγραμμένων χρηστών και έπειτα η νέα βάση δεδομένων δημοσιοποιείται για οποιαδήποτε έρευνα και χρήση. Η εργασία αυτή έχει ως κύρια διαφορά ότι αφορά βάσεις δεδομένων οι οποίες παραμένουν στον οργανισμό που δημιούργησε τις σημασιολογικά επαυξημένες τροχιές που περιέχουν. Στόχος μας είναι η ανάπτυξη μιας *μηχανής ελέγχου ερωτημάτων* (*queries*) με σκοπό να αποφευχθεί η παραβίαση της ιδιωτικότητας κατά τη χρήση της βάσης και ταυτόχρονα να υπάρχει η μικρότερη δυνατή αλλοίωση στα στοιχεία που επιστρέφουν ως απαντήσεις των ερωτημάτων. Βασιζόμενοι σε προηγούμενη εργασία [7] υλοποιούμε δύο αλγορίθμους τους οποίους καλούμε *Auditor_check_overlapping* και *Zoom_out*. Ο πρώτος αλγόριθμος αποτελεί τον κύριο μηχανισμό ανίχνευσης πιθανής επίθεσης-παραβίασης και εκτελείται σε κάθε περίπτωση που θα τεθεί ένα ερώτημα προς τη βάση. Ο δεύτερος αλγόριθμος που υλοποιήσαμε τροποποιεί το αρχικό ερώτημα, αν δεν μπορεί να απαντηθεί για λόγους ασφαλείας, στο πλησιέστερο δυνατό ερώτημα που μπορεί να απαντηθεί χωρίς παραβίαση της ασφάλειας. Έτσι ενισχύεται η φιλικότητα προς τον χρήστη και κυρίως η λειτουργικότητα του μηχανισμού.

2 Σχετικές Εργασίες

Η προστασία της ιδιωτικότητας σε βάσεις δεδομένων απασχολεί την επιστημονική κοινότητα τα τελευταία τριάντα χρόνια. Για την αποφυγή της ταυτοποίησης και σύνδεσης στοιχείων με φυσικά πρόσωπα κρίθηκε σκόπιμο τα δεδομένα να τροποποιούνται με τεχνικές ανωνυμοποίησης. Η βασική τεχνική ανωνυμοποίησης η οποία προέβλεπε την αφαίρεση στοιχείων από κάθε εγγραφή, που προσδιόριζαν την ταυτότητα του φυσικού προσώπου, όπως για παράδειγμα ονοματεπώνυμο ή αριθμός ταυτότητας αποδείχθηκε μη επαρκής. Συγκεκριμένα η L.Sweeney [20] με την έρευνά της απέδειξε ότι κάποιες φορές συνδέοντας στοιχεία από διαφορετικές πηγές είναι πιθανό ένας *κακόβουλος χρήστης* να εξάγει συμπεράσματα για τον στόχο του, ακόμα και να ταυτοποιήσει τις εγγραφές που είναι ανωνυμοποιημένες. Ορίζοντας αρχικά τα πεδία της βάσης των οποίων ο συνδυασμός μπορεί να οδηγήσει στην ταυτοποίηση του φυσικού

προσώπου ως *Quasi-identifiers* και τα πεδία που περιέχουν ευαίσθητη πληροφορία ως *Sensitive-attributes*, πρότεινε ως λύση τη θεωρία του *k*-anonymity. Για να στηρίξει την έρευνά της παρουσίασε ένα παράδειγμα (Εικόνα 1) κατά το οποίο συνδυάζοντας δύο διαφορετικές βάσεις στοιχείων με κοινά χαρακτηριστικά τον ταχυδρομικό κώδικα, την ημερομηνία γέννησης και το φύλο (Quasi-identifiers) ένας επιτιθέμενος μπορούσε να εξάγει περαιτέρω γνώση και συμπεράσματα για τον στόχο του.



.

Εικόνα 1 Quasi identifiers - L. Sweeney [20]

Σύμφωνα με την πρότασή της το σύνολο των δεδομένων είναι *k*-anonymous όταν για οποιοδήποτε συνδυασμό τιμών των πεδίων quasi-identifiers, μια εγγραφή είναι δυσδιάκριτη από *k*-1 άλλες εγγραφές. Έτσι ο επιτιθέμενος δεν μπορεί να συσχετίσει μια εγγραφή με ένα συγκεκριμένο άτομο με πιθανότητα μεγαλύτερη από 1/*k*. Στον παρακάτω πίνακα (Εικόνα 2) παρουσιάζεται ένα παράδειγμα από 3-anonymity καθώς τα quasi-identifier πεδία που είναι το Zipcode, το Age και το Gender έχουν τροποποιηθεί με τέτοιο τρόπο ώστε κάθε εγγραφή είναι δυσδιάκριτη από άλλες δύο εγγραφές. Η ιδέα του *k*-anonymity ήταν επαναστατική και αποτελεί έναν σημαντικό άξονα στις περισσότερες σύγχρονες μεθόδους που έχουν αναπτυχθεί για την προστασία της ιδιωτικότητας.

Zipcode	Age	Gender	Disease
476**	2*	*	Ovarian Cancer
476**	2*	*	Ovarian Cancer
476**	2*	*	Skin Cancer
479**	[41,60]	*	Flu
479**	[41,60]	*	Arthritis
479**	[41,60]	*	Arthritis

Εικόνα 2 Παράδειγμα *3-anonymity*

Όμως παρόλα αυτά διαπιστώθηκε ότι ούτε αυτή η τεχνοτροπία ήταν επαρκής καθώς δεν αντιμετώπιζε ικανοποιητικά δύο είδη επιθέσεων. Τα *homogeneity attacks* [13] και τα *background knowledge attacks* [13]. Η πρώτη επίθεση αφορά περιπτώσεις όπου και οι *k* εγγραφές, όπως διαμορφώθηκαν, περιέχουν την ίδια τιμή στα ευαίσθητα πεδία. Έτσι μια εγγραφή μπορεί να είναι δυσδιάκριτη από *k*-1 άλλες εγγραφές αλλά η ευαίσθητη πληροφορία να αποκαλύπτεται. Η δεύτερη επίθεση στηρίζεται στο γεγονός ότι αν ο επιτιθέμενος έχει κάποιου είδους επιπλέον γνώση, αυξάνει την πιθανότητά του να εξάγει ευαίσθητες πληροφορίες για τον στόχο του. Ως λύση στους περιορισμούς του *k*-anonymity οι Machanavajjhala et al. [13] πρότειναν τον κανόνα του *l*-diversity. Ένα σύνολο δεδομένων ικανοποιεί την αρχή του *l*-diversity αν για κάθε ομάδα εγγραφών με κοινή τιμή των quasi-identifiers (ονομαζόμενη ως *equivalence*

class), υπάρχουν *l* διακριτές τιμές στα αντίστοιχα ευαίσθητα πεδία τους. Στην Εικόνα 3 βλέπουμε ένα παράδειγμα 3-diversity όπου έστω και με την έξτρα γνώση ότι ο στόχος του βρίσκεται στην πρώτη κατηγορία, οι τρείς διαφορετικές τιμές στα ευαίσθητα πεδία Salary και Disease δεν μπορούν να προσφέρουν στον επιτιθέμενο επιπλέον πληροφορία και καθιστούν τον στόχο δυσδιάκριτο.

Zipcode	Age	Salary	Disease
476**	2*	4K	Flu
476**	2*	6K	Gastritis
476**	2*	5K	Pneumonia
479**	>40	4K	Gastritis
479**	>40	30K	Cancer
479**	>40	15K	Flu

Εικόνα 3 Παράδειγμα *3-diversity*

Το επόμενο μεγάλο βήμα στην προστασία της ιδιωτικότητας των δεδομένων έγινε από τους Li et al. [12] οι οποίοι αντιμετώπισαν τα *similarity attacks* και *skewness attacks.* Η πρώτη μορφή αφορά περιπτώσεις όπου οι διακριτές τιμές ενός equivalence class απέχουν ελάχιστα μεταξύ τους και στην ουσία προσφέρουν τελικά μια γνώση στον επιτιθέμενο. Στο παράδειγμα (Εικόνα 3) οι τιμές του Salary της πρώτης κλάσης είναι διακριτές αλλά η διαφορά μεταξύ τους σημασιολογικά είναι πάρα πολύ μικρή. Η δεύτερη μορφή επιθέσεων αφορά περιπτώσεις όπου για παράδειγμα το 99% των εγγραφών της κλάσης έχει το ίδιο Disease οπότε είναι λογικό να συμπεράνει ο επιτιθέμενος ότι ο στόχος του έχει κατά πάσα πιθανότητα αυτή την τιμή στο Disease γνωρίζοντας ότι ανήκει σ αυτή την κλάση. Προς αντιμετώπιση λοιπόν αυτών των επιθέσεων εισήγαγαν τον κανόνα του *t*-closeness. Η αρχή αυτή ικανοποιείται αν για κάθε κλάση (equivalence class) η κατανομή της συχνότητας εμφάνισης των διακριτών τιμών του ευαίσθητου πεδίου, δεν αποκλίνει από την κατανομή που παρουσιάζουν αυτές οι τιμές σε όλο τον πίνακα παραπάνω από ένα κατώφλι *t*.

Οι έρευνες αυτές αποτέλεσαν το εφαλτήριο για τον τομέα της ιδιωτικότητας στις σχεσιακές βάσεις δεδομένων. Η ιδιομορφία των χωρικών και χωροχρονικών δεδομένων σε βάσεις κινούμενων αντικειμένων, απαιτεί ακόμα πιο περίπλοκες τεχνικές για την διασφάλιση της ιδιωτικότητας και δεν καλύπτεται μόνο από την εφαρμογή τεχνικών που εφαρμόζονται σε σχεσιακά δεδομένα. Μπορεί κανείς να αναλογιστεί την ιδιομορφία αυτή λαμβάνοντας υπόψη ότι ένα χωροχρονικό σημείο μπορεί να είναι quasi-identifier για έναν χρήστη, αλλά ταυτόχρονα sensitive για έναν άλλο. Θα μπορούσαμε να κατηγοριοποιήσουμε τις τεχνικές που αφορούν δεδομένα κινητικότητας και θέσης και αποτέλεσαν πηγή έμπνευσης για την παρούσα εργασία σε τρεις μεγάλες κατηγορίες:

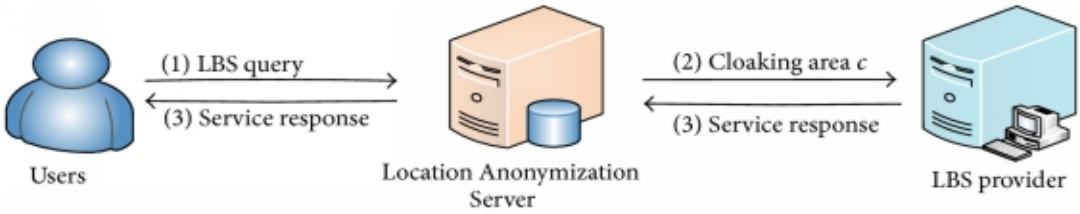
- Προστασία της ιδιωτικότητας σε υπηρεσίες γεωγραφικής θέσης (*LBS*)
- Προστασία της ιδιωτικότητας σε δημοσιοποιημένα δεδομένα κίνησης
- Προστασία της ιδιωτικότητας από ερωτήματα σε βάσεις κινούμενων αντικειμένων

Η πρώτη κατηγορία αναφέρεται σε υπηρεσίες που παρέχονται από τον πάροχο στον χρήστη με τη μορφή απάντησης σε ερώτημα που έθεσε ο τελευταίος, σύμφωνα με την τοποθεσία του. Η δεύτερη κατηγορία περιλαμβάνει τεχνικές με τις οποίες μια βάση τροχιών κινούμενων αντικειμένων (*trajectory database*) τροποποιείται με στόχο την ανωνυμοποίηση των δεδομένων της και έπειτα η νέα έκδοση δημοσιεύεται στο κοινό. Τεχνικές της τρίτης κατηγορίας προσεγγίζουν την ιδέα αυτής της εργασίας καθότι προϋποθέτουν ότι τα δεδομένα που συλλέγονται παραμένουν στον οργανισμό που τα δημιούργησε και τα συντηρεί και δίδεται η δυνατότητα ερωτήσεων (queries) προς τη βάση μόνο σε *εγκεκριμένους χρήστες*.

2.1 Προστασία της ιδιωτικότητας σε υπηρεσίες γεωγραφικής θέσης

Με τη χρήση των υπηρεσιών γεωγραφικής θέσης (*Location Based Services*) ένα φυσικό πρόσωπο αποστέλλει μέσω μιας κινητής συσκευής την τοποθεσία του και την ταυτότητά του για να του παρασχεθεί μια υπηρεσία θέσης. Η ιδιωτικότητα των τοποθεσιών του χρήστη θα πρέπει να διαφυλαχτεί καθώς η αποκάλυψή τους μπορεί να οδηγήσει έναν επιτιθέμενο στο να εξάγει συμπεράσματα για τις συνήθειες του χρήστη, την επισκεψιμότητά του σε τοποθεσίες και γενικότερα να οδηγήσει στην διαρροή ευαίσθητων προσωπικών δεδομένων. Το πρόβλημα της *ιδιωτικότητας της θέσης* (*location privacy*) μελετήθηκε από την επιστημονική κοινότητα και πλήθος τεχνικών, στηριζόμενες σε μια τροποποιημένη μορφή του *k*-anonymity [6], έχουν προταθεί και δοκιμαστεί. Το ζητούμενο είναι να καταστεί ο αιτών την LBS δυσδιάκριτος από τουλάχιστον *k*-1 άλλους χρήστες, που επιθυμούν την ίδια υπηρεσία στον ίδιο χώρο. Η τοποθεσία του χρήστη παρουσιάζεται ως μια ολόκληρη περιοχή με αποτέλεσμα ο επιτιθέμενος να μην μπορεί να συμπεράνει την ακριβή θέση του στόχου του και να αδυνατεί να τον ταυτοποιήσει με πιθανότητα μεγαλύτερη από 1/*k*.

Η χρήση δυναμικών βάσεων δεδομένων από τους παρόχους LBS σε συνδυασμό με την επικοινωνία με τους χρήστες, τους καθιστούν κατά γενική ομολογία μη αξιόπιστους. Για το λόγο αυτό ένας έμπιστος διακομιστής (*anonymizer*) ως τρίτη ξεχωριστή οντότητα παρεμβάλλεται μεταξύ χρήστη και παρόχου. Ο διακομιστής αυτός αφού λάβει το αίτημα του χρήστη, αρχικά αφαιρεί τις εμφανείς πληροφορίες που μπορούν να προδώσουν την ταυτότητά του και έπειτα ανωνυμοποιεί την τοποθεσία του καθιστώντας την δυσδιάκριτη ανάμεσα σε *k* θέσεις. Στη συνέχεια αποστέλλει αυτά τα δεδομένα στον πάροχο, ο οποίος απαντά για τις *k* πιθανές θέσεις της περιοχής και επιστρέφει πλέον ο anonymizer στον πραγματικό αιτών της υπηρεσίας την σωστή απάντηση.



Εικόνα 4 LBS- Anonymization Server – Jin Wang et al. [11]

Υπάρχουν δύο κατηγορίες από LBSs και αντίστοιχα δύο κατηγορίες τεχνικών προστασίας της ιδιωτικότητας στις υπηρεσίες αυτές. Η πρώτη κατηγορία αντιστοιχεί σε υπηρεσίες θέσης όπου υπάρχει μια μόνο επικοινωνία μεταξύ χρήστη και παρόχου και ονομάζονται *Snapshot LBSs* [8]. Στην δεύτερη κατηγορία ο χρήστης καθώς μετακινείται ζητά συνεχώς την παροχή πληροφοριών με αποτέλεσμα την πολλαπλή αποστολή δεδομένων θέσης. Τέτοιες περιπτώσεις καλούνται *Continuous LBSs* [5]. Τεχνικές *απόκρυψης* (*cloaking*) και *διαταραχής* (*perturbation*) χαρακτηρίζουν τις λύσεις που έχουν προταθεί και στις δύο κατηγορίες. Με εφαρμογή των τεχνικών απόκρυψης είτε γύρω από τον χρήστη που θα ζητήσει υπηρεσία θα κατασκευαστεί μια περιοχή MBB (*Minimum Bounding Box*) η οποία θα περιλαμβάνει και τις υπόλοιπες *k*-1 θέσεις χρηστών όπως προαναφέραμε, είτε η συνολική περιοχή θα έχει εξ αρχής κατακερματιστεί σε τμήματα ενός πλέγματος και με αφετηρία το τμήμα που βρίσκεται ο αιτών θα προσθέτονται γειτονικά ακέραια τμήματα μέχρι να ικανοποιηθεί ο κανόνας του *k*-anonymity και στην ολική περιοχή που επιλέχθηκε να ανήκουν πλέον τουλάχιστον *k* χρήστες. Οι μέθοδοι διαταραχής αναγκάζουν τον πάροχο να απαντήσει δεχόμενος ως είσοδο μια ευρύτερη περιοχή όπου κάπου εντός των ορίων της βρίσκεται ο χρήστης. Είναι λογικό πως όσο μεγαλύτερη είναι αυτή η περιοχή τόσο πιο πιθανό είναι να έχουμε μη ικανοποιητική παρεχόμενη υπηρεσία.

2.2 Προστασία της ιδιωτικότητας κατά την δημοσιοποίηση δεδομένων κίνησης

Η προστασία της ιδιωτικότητας σε *βάσεις κινούμενων αντικειμένων* (*Moving Object Databases*) οι οποίες θέλουμε να δημοσιευτούν για στατιστικούς λόγους και έρευνα αποτέλεσαν τον κύριο τομέα ενασχόλησης των επιστημόνων για πολλά χρόνια, με πληθώρα προτάσεων και τεχνικών. Σε αντίθεση με τις δυναμικές βάσεις δεδομένων στις υπηρεσίες γεωγραφικής θέσης, στον τομέα αυτό αναφερόμαστε σε στατικές βάσεις των οποίων τα δεδομένα κίνησης αναλύονται για να εξαγχθούν πρότυπα. Η τεχνική του *k*-anonymity υιοθετείται και βρίσκει εφαρμογή σε συνδυασμό με επιπλέον τεχνικές-αντίμετρα για την ανωνυμοποίηση και τροποποίηση της αρχικής βάσης στην μορφή που θα δοθεί προς χρήση στο ευρύτερο κοινό.

Η πρώτη τεχνική χρησιμοποιεί μια μορφή διαταραχής της διαδρομής (*path perturbation*) που ακολουθεί μια καταγεγραμμένη οντότητα, ώστε ο επιτιθέμενος να μην μπορεί να ακολουθήσει με βεβαιότητα την πορεία της. Ο αλγόριθμος που πρότειναν οι Hoh et al. [10] εξετάζει τις τροχιές και στην περίπτωση που δύο μη διασταυρώμενες τροχιές βρίσκονται αρκετά κοντά δημιουργεί μια ψεύτικη διασταύρωση μεταξύ τους. Στην ανωνυμοποιημένη έκδοση της βάσης τροχιών η πιθανότητα ένας κακόβουλος χρήστης να ακολουθήσει μια διαδρομή και να ταυτοποιήσει τον χρήστη στον οποίο ανήκει αυτή μειώνεται όσο αυξάνεται ο αριθμός των διασταυρώσεων της τροχιάς. Το μειονέκτημα είναι ότι όσο μεγαλώνει η ακτίνα μέσα στην οποία μπορεί να δημιουργηθεί ένα ψεύτικο σημείο διασταύρωσης και αυξάνεται ο βαθμός προστασίας της ιδιωτικότητας, υφίσταται μεγαλύτερη αλλοίωση των αρχικών τροχιών, με αποτέλεσμα τη μείωση της χρησιμότητας των δεδομένων.

Με ένα διαφορετικό τρόπο σκέψης εργάστηκαν οι Terrovitis και Mamoulis [21], υποθέτοντας στην μελέτη τους πως κάθε τροχιά είναι μια ακολουθία σημείων, τα οποία επισκέφτηκε ο χρήστης και είχε μια μορφή συναλλαγής. Θεωρούμε πως η πρότερη γνώση που έχει ένας κακόβουλος χρήστης είναι τμήματα των τροχιών επίσης σε μορφή ακολουθίας σημείων. Ο αλγόριθμός τους χρησιμοποιώντας τεχνική εξάλειψης σημείων (*suppression*), με επαναληπτική λειτουργία δημιουργεί μια νέα βάση τροχιών που διακατέχεται από την ιδέα του *k*-anonymity. Σκοπός είναι για κάθε ακολουθία τοποθεσιών που γνωρίζει ο επιτιθέμενος να υπάρχουν τουλάχιστον *k* τροχιές που περιέχουν το τμήμα αυτό.

id	trajectory
t_1	$a_1 \rightarrow b_1 \rightarrow a_2$
t_2	$a_1 \rightarrow b_1 \rightarrow a_2 \rightarrow b_3$
t_3	$a_1 \rightarrow b_2 \rightarrow a_2$
t_4	$a_1 \rightarrow a_2 \rightarrow b_2$
t_5	$a_1 \rightarrow a_3 \rightarrow b_1$
t_6	$a_3 \rightarrow b_1$
t_7	$a_3 \rightarrow b_2$
t_8	$a_3 \rightarrow b_2 \rightarrow b_3$

(a) exact data (T)

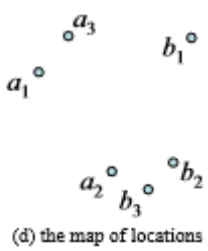
id	trajectory
t_1^A	$a_1 \rightarrow a_2$
t_2^A	$a_1 \rightarrow a_2$
t_3^A	$a_1 \rightarrow a_2$
t_4^A	$a_1 \rightarrow a_2$
t_5^A	$a_1 \rightarrow a_3$
t_6^A	a_3
t_7^A	a_3
t_8^A	a_3

(b) A's knowledge (T_A)

Εικόνα 5 Παράδειγμα data suppression - Terrovitis M, Mamoulis N [21]

id	trajectory
t_1'	$a_1 \rightarrow b_1 \rightarrow a_2$
t_2'	$a_1 \rightarrow b_1 \rightarrow a_2$
t_3'	$a_1 \rightarrow b_2 \rightarrow a_2$
t_4'	$a_1 \rightarrow a_2 \rightarrow b_2$
t_5'	$a_3 \rightarrow b_1$
t_6'	$a_3 \rightarrow b_1$
t_7'	$a_3 \rightarrow b_2$
t_8'	$a_3 \rightarrow b_2$

(c) transformed database (T')



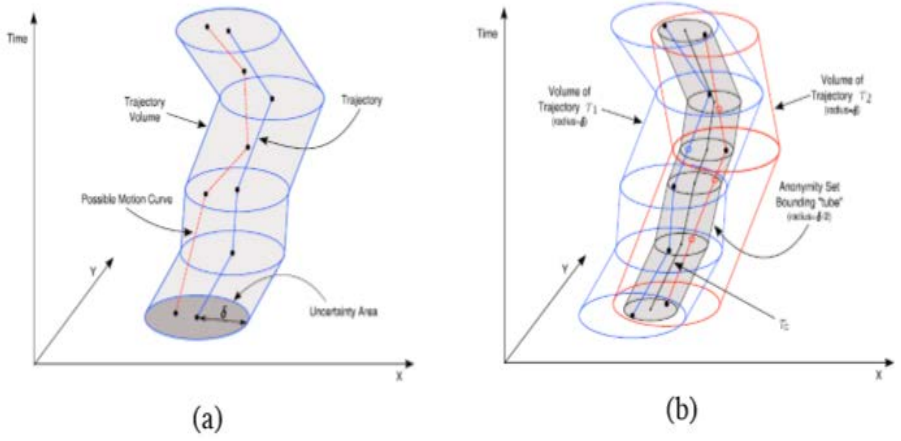
(d) the map of locations

Εικόνα 6 Παράδειγμα data suppression - Terrovitis M, Mamoulis N [21]

Για παράδειγμα η εικόνα (a) (Εικόνα 5) αντιπροσωπεύει την αρχική βάση δεδομένων και η εικόνα (b) τα τμήματα που γνωρίζει ο επιτιθέμενος. Επιδιώκοντας 2-anonymity οι τοποθεσίες b_3 και a_1 εξαλείφονται από τις τροχιές t_2, t_8 και t_5 αντίστοιχα καθώς βρίσκονται κοντά στα γειτονικά τους σημειά όπως παρουσιάζεται και στην εικόνα (d), και η νέα βάση προς δημοσίευση είναι ο πίνακας της εικόνας (c) (Εικόνα6).

Τα κύρια μειονεκτήματα αυτής της τεχνικής είναι ότι δεν λαμβάνει υπόψη τον παράγοντα του χρόνου και δεν είναι απόλυτα ρεαλιστική, καθώς προϋποθέτει να γνωρίζουμε όλες τις γνώσεις του επιτιθέμενου πριν την δημοσίευση των δεδομένων.

Βασιζόμενοι σε τεχνικές *συσταδοποίησης* (*clustering*) οι Abul et al. [1], [2] παρουσίασαν τους αλγορίθμους NWA (Never Walk Alone) και W4M (Wait for Me). Επικαλούμενοι την εγγενή αβεβαιότητα των κινούμενων αντικειμένων, λόγο του περιθωρίου σφάλματος που έχουν οι συσκευές καταγραφής, παρουσίασαν μια τροχιά ως ένα κύλινδρο (Εικόνα 7 (a)), όπου η ακτίνα του αντιπροσωπεύει το περιθώριο σφάλματος (αβεβαιότητα). Το αντικείμενο κινείται μέσα στον κύλινδρο χωρίς όμως να γνωρίζουμε τα ακριβή σημεία της τροχιάς. Αυτό έχει ως αποτέλεσμα η τροχιά να μην ξεχωρίζει από κάθε άλλη τροχιά μέσα στον ίδιο κύλινδρο (Εικόνα 7 (b)). Ο NWA προσπαθεί να ικανοποιήσει αυτό που οι συγγραφείς όρισαν ως (k,δ)-anonymity. Για να ικανοποιείται το (k,δ)-anonymity το αρχικό σύνολο τροχιών D μετασχηματίζεται στο D', έτσι ώστε κάθε τροχιά τ του D' να ανήκει πάντα σε ένα σύνολο τροχιών S του D'. Το σύνολο S περιέχει τουλάχιστον k τροχιές, οι οποίες τοποθετούνται χωροχρονικά σε ένα κυλινδρικό όγκο με ακτίνα δ/2. Αρχικά γίνεται μια επεξεργασία όπου οι τροχιές ομαδοποιούνται σε κλάσεις σύμφωνα με την χρονική στιγμή έναρξης και λήξης τους. Για να είναι εφικτό αυτό ορισμένες τροχιές τροποποιούνται με βάση μια χρονική παράμετρο/σταθερά (π.χ. ανά 15'), αφαιρώντας τμήματα για να έχουν κοινή αρχή και τέλος με την ομάδα στην οποία ανήκουν. Στη συνέχεια λαμβάνει μέρος η συσταδοποίηση κατά την οποία ως κέντρο της *συστάδας* επιλέγεται η πιο απομακρυσμένη τροχιά από την προηγούμενη επιλεγμένη. Η αρχική τροχιά αποτελεί την πιο απομακρυσμένη από το νοητό κέντρο των τροχιών. Κάθε τροχιά η οποία επιλέγεται ως κέντρο χρησιμοποιείται για να σχηματίσει συστάδα από ακριβώς k τροχιές επιλέγοντας τις k-1 κοντινότερες που ικανοποιούν τα κριτήρια.



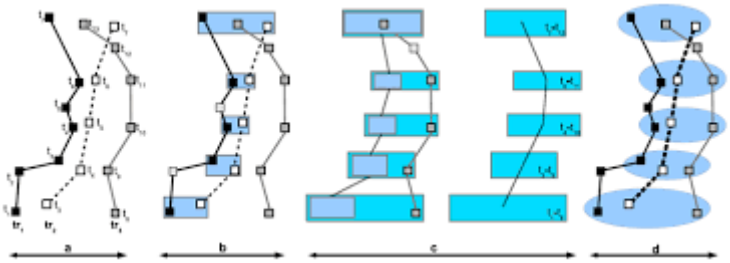
Εικόνα 7 NWA – Abul et al. [1]

Τέλος εφαρμόζεται μια τεχνική μετακίνησης χωρικών σημείων (*space translation*) ώστε τα σημεία κάθε τροχιάς σε κάθε συστάδα να απέχουν το πολύ δ/2 από τον κεντρικό άξονα του εικονικού κυλίνδρου.

Το μεγάλο μειονέκτημα του αλγορίθμου NWA ήταν η χρήση της συνάρτηση ευκλείδειας απόστασης κατά την φάση της συσταδοποίησης. Η μετρική αυτή μπορούσε να χρησιμοποιηθεί μόνο σε όμοιου μήκους τροχιές για αυτό το λόγο υπήρχε και το πρώτο στάδιο επεξεργασίας. Προς αντιμετώπιση αυτών των αδυναμιών οι συγγραφείς Abul et al. [2] προχώρησαν στην

μελέτη και δημοσίευση του αλγορίθμου W4M ο οποίος αποτελεί βελτίωση του προηγούμενου έργου τους. Πλέον το πρώτο στάδιο επεξεργασίας δεν χρειάζεται οπότε δεν εκτελείται, ο τρόπος υπολογισμού και σύγκρισης των αποστάσεων βασίζεται στην μετρική EDR (*Edit Distance Function*) [4] απόσταση και όχι στην ευκλείδεια ενώ η γενική φιλοσοφία της συσταδοποίησης παραμένει ίδια. Το σύνολο των τροχιών επεξεργάζεται αδιαίρετο κατά το στάδιο της συσταδοποίησης και η τροχιά η οποία αποτελεί το κέντρο της εκάστοτε συστάδας, σε κάθε επανάληψη του αλγορίθμου επιλέγεται τυχαία. Στο τελευταίο στάδιο όλες οι τροχιές σε κάθε συστάδα τροποποιούνται ώστε να "συσσωματωθούν" με την τροχιά, που έχει επιλεχθεί τυχαία στο προηγούμενο βήμα, ως το κέντρο της συστάδας. Επειδή υπάρχει χρονική ασυμφωνία μεταξύ των τροχιών του *cluster,* προσθαφαιρούνται σημεία όπου είναι απαραίτητο, ώστε να προσεγγίσουν όλες οι τροχιές την τυχαία επιλεγμένη και να έχουν το ίδιο πλήθος σημείων και κοινά χρονικά στιγμιότυπα.

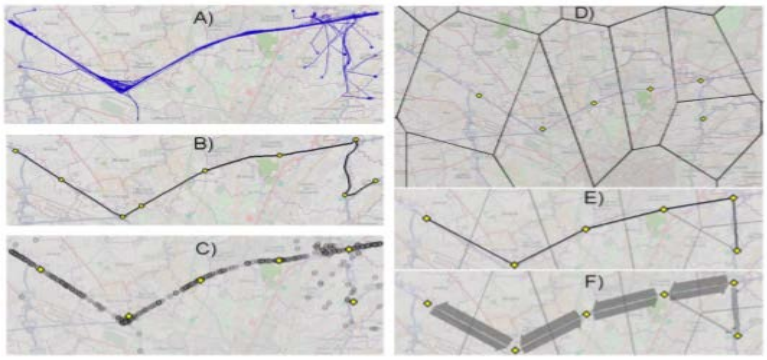
Οι Nergiz et al. [17] παρουσίασαν έναν αλγόριθμο που λειτουργεί με τεχνική *γενίκευσης* (*generalization*) για την προστασία της ιδιωτικότητας σε στατικές βάσεις δεδομένων τροχιών. Ο AWO (Always Walk with Others) χρησιμοποιώντας χωρικές και χρονικές γενικεύσεις δεδομένων και απαλείφοντας κάποια σημεία και τροχιές κατασκευάζει μια νέα *k*-anonymized βάση. Θεωρώντας ότι ο επιτιθέμενος μπορεί να έχει γνώση από ένα τμήμα ή και ολόκληρης της τροχιάς στην αρχική βάση, η λειτουργία του αλγορίθμου διαρθρώνεται σε δύο βήματα. Στο πρώτο βήμα σχηματίζονται ομάδες αποτελούμενες από *k* τροχιές.



Εικόνα 8 AWO – Nergiz et al. [17]

Σε κάθε ομάδα επιλέγεται τυχαία μια τροχιά (Εικόνα 8 (a)) και βρίσκοντας την πλησιέστερη ευθυγραμμίζονται όσα από τα σημεία τους είναι δυνατό και αντικαθίστανται με ένα πλαίσιο οριοθέτησης (MBB) όπως παρουσιάζεται γραφικά στην Εικόνα 8 (b) για τις πρώτες δύο τροχιές του γραφήματος από αριστερά. Τα σημεία που δεν μπορούν να ευθυγραμμιστούν απαλείφονται. Η διαδικασία επαναλαμβάνεται μέχρι να περιέχονται και οι *k* τροχιές και κατασκευάζεται μια ακολουθία πλέον από MBBs (Εικόνα 8 (c)). Στο δεύτερο βήμα του αλγορίθμου εκτελείται μια τυχαία ανακατασκευή των σημείων κάθε τροχιάς μέσα στα MBBs για να επιτευχθεί μεγαλύτερου βαθμού ανωνυμοποίηση (Εικόνα 8 (d)).

Μια τεχνική βασισμένη επίσης σε γενίκευση και συσταδοποίηση αλλά διαφορετική από όσες προαναφέρθηκαν αποτέλεσε ο αλγόριθμος PPSG των Monreale et al. [16]. Στην συγκεκριμένη μελέτη γίνεται η υπόθεση ότι ο επιτιθέμενος εκτός από τμήματα τροχιών, για τις εγγραφές που τον ενδιαφέρουν, μπορεί να γνωρίζει και τις τεχνικές ανωνυμοποίησης που έχουν εφαρμοστεί. Ο αλγόριθμος έχει τρία στάδια. Στο πρώτο στάδιο η περιοχή των τροχιών χωρίζεται σε τμήματα. Από κάθε τροχιά ανακτώνται χαρακτηριστικά σημεία όπως η αρχή, το τέλος της, σημεία στροφής, σημαντικά σημεία στάσης και αντιπροσωπευτικά σημεία μεγάλων ευθειών. Στην Εικόνα 9 (B) παρουσιάζεται μια τροχιά και τα χαρακτηριστικά της σημεία. Στη συνέχεια στα χαρακτηριστικά σημεία εφαρμόζεται χωρική συσταδοποίηση. Αυτό δίνει τη δυνατότητα να χωριστεί η περιοχή σε κελιά *Voronoi* (Εικόνα 9 (D)), τα οποία περιέχουν τα σημεία κάθε συστάδας και να υπολογιστεί το κεντροειδές τους. Έπειτα όλα τα σημεία της συστάδας ταυτίζονται με το κεντροειδές.



Εικόνα 9 Διαδικασία γενίκευσης PPSG – Monreale et al. [16]

Στο δεύτερο στάδιο αν υπάρχουν κελιά όπου το πλήθος σημείων δεν ικανοποιεί ένα κατώφλι *k* τότε γίνεται ένωση με γειτονικά κελιά και υπολογίζεται το νέο κεντροειδές. Οι γενικευμένες τροχιές με μια μορφή όπως στην Εικόνα 9 (E), δημιουργούν μια βάση δεδομένων καλούμενη *prefix tree*, στην οποία κάθε κόμβος αντιπροσωπεύει ένα κελί και τις κοινές τροχιές που εισέρχονται από τα γειτονικά κελιά. Ο αλγόριθμος αντιστοιχίζει κάθε σημείο μιας τροχιάς σε ένα κόμβο του prefix tree. Έτσι στο τελευταίο στάδιο γίνεται αποκοπή τμημάτων του δενδροειδούς εξετάζοντας σε ποιούς κόμβους δεν υποστηρίζεται ο κανόνας του *k*-anonymity.

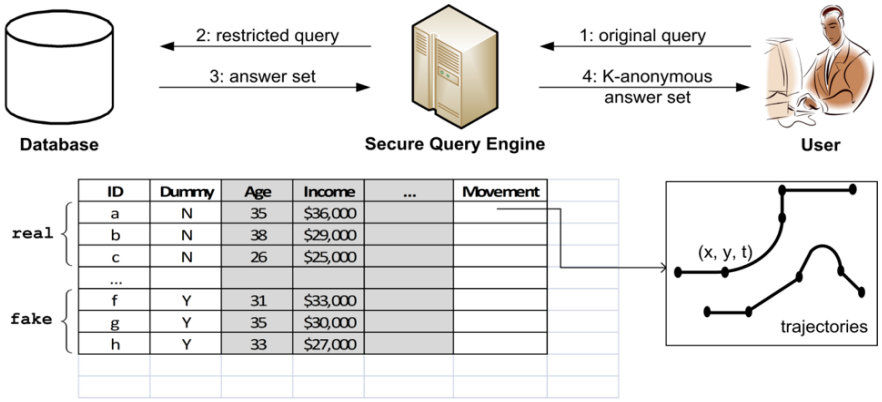
Ο βαθμός στον οποίο απαιτείται προστασία της ιδιωτικότητας σε μια τροχιά δεν είναι ο ίδιος για όλους τους χρήστες και τις τροχιές τους. Με αυτό το σκεπτικό εργάστηκαν οι Mahdavifar et al. [14]. Ο αλγόριθμος που πρότειναν είναι βασισμένος σε συσταδοποίηση και χρησιμοποιεί και αυτός την μετρική EDR για συγκρίσεις αποστάσεων των τροχιών από το κεντροειδές της συστάδας. Το πλήθος των τροχιών σε μια συστάδα καθορίζεται από το βαθμό ιδιωτικότητας με το οποίο χαρακτηρίζεται μια τροχιά. Τροχιές οι οποίες έχουν υψηλό βαθμό εξετάζονται με προτεραιότητα και θα βρίσκονται σε συστάδες με μεγαλύτερο πλήθος τροχιών σε σχέση με τροχιές χαμηλότερου βαθμού. Με την μετρική EDR ελέγχεται η απόσταση του κεντροειδούς της συστάδας από την εξεταζόμενη τροχιά. Η απόσταση αυτή έχει ένα άνω όριο το οποίο δεν πρέπει να ξεπερνιέται. Στη συνέχεια οι Mahdavifar et al. [14] χρησιμοποίησαν τον αλγόριθμο FLTP με χρήση του οποίου γίνεται πλέον σε δεύτερο στάδιο η ανωνυμοποίηση των τροχιών κάθε συστάδας.

2.3 Προστασία της ιδιωτικότητας από ερωτήματα σε βάσεις τροχιών κινούμενων αντικειμένων

Η δυνατότητα και οι τεχνικές παραβίασης της ιδιωτικότητας των εγγραφών μιας βάσης, με χρήση διαδοχικών στατιστικών ερωτημάτων, είναι ένα πεδίο που ως προς τις παραδοσιακές μορφές βάσεων δεδομένων έχει μελετηθεί εδώ και χρόνια, με μελέτες όπως αυτή των Adam and Wortmann [3]. Αυτό το πεδίο εργασιών αποτελεί σκαλοπάτι για νεότερες εργασίες που αφορούν ερωτήματα σε βάσεις κινούμενων αντικειμένων. Στη συγκεκριμένη μελέτη, οι Adam and Wortmann ανέλυσαν όλα τα καταγεγραμμένα προβλήματα ιδιωτικότητας στις στατιστικές βάσεις της εποχής, καταγράφοντας τέσσερεις κύριες κατηγορίες αντιμετώπισης. Οι κατηγορίες αυτές περιείχαν την έννοια του *A-population* το οποίο αποτελεί ένα πρώιμο 2-anonymity, καθώς και μια προσέγγιση προάγγελο του ευρέως πλέον χρησιμοποιούμενου κύβου δεδομένων. Επίσης η δεύτερη και τρίτη κατηγορία περιέγραφαν τεχνικές διατάραξης δεδομένων (*data perturbation*) και διατάραξης της απάντησης που δίνεται σε κάθε ερώτημα. Η τέταρτη κατηγορία παρουσίαζε τεχνικές ελέγχου ερωτημάτων σύμφωνα με το μέγεθος της απάντησης και τεχνικές ελέγχου επικάλυψης των ερωτημάτων. Αξίζει να αναφέρουμε ότι οι συγγραφείς καταλήγουν σε ένα σημαντικό και διαχρονικό συμπέρασμα, ότι εάν συνεργαστούν πολλοί κακόβουλοι χρήστες έχοντας ένα κοινό στόχο, είναι ιδιαίτερα δύσκολο να διαπιστωθεί και να αποτραπεί η επίθεσή τους.

Πολλοί οργανισμοί διατηρούν τα δεδομένα που συλλέγουν αποκλειστικά στην υποδομή τους χωρίς να τα δημοσιεύουν, αλλά δίνουν τη δυνατότητα σε εγκεκριμένους χρήστες να προβούν σε ερωτήματα προς τη βάση. Αυτό γίνεται για λόγους όπως ο έλεγχος πρόσβασης μόνο από εξουσιοδοτημένα πρόσωπα, η δυνατότητα ανανέωσης των δεδομένων με πιο πρόσφατα ανά διαστήματα, η δυνατότητα διορθωτικών κινήσεων σε περίπτωση παραβίασης της προστασίας της ιδιωτικότητας αλλά και λόγο νομικών περιορισμών.

Οι Gkoulalas-Divanis et al. [9] ήταν οι πρώτοι που εργάστηκαν πάνω σε ένα μηχανισμό προστασίας της ιδιωτικότητας, για δεδομένα τα οποία διατηρούνται στον οργανισμό που τα συλλέγει. Σεβόμενοι την αρχή του *k*-anonymity πρότειναν έναν μηχανισμό ο οποίος δημιουργεί ψεύτικες τροχιές και απώτερος στόχος του είναι να κάνει δυσδιάκριτες τις ψεύτικες από τις αληθινές τροχιές (Εικόνα 10), που λαμβάνει ως απαντήσεις στα ερωτήματά του ένας χρήστης. Μεγάλο μειονέκτημα του αλγορίθμου ήταν ότι δεν λάμβανε υπ' όψη του τη χρονική παράμετρο κατά την κατασκευή των ψεύτικων τροχιών.



Εικόνα 10 Privacy query engine - Gkoulalas-Divanis et al. [9]

Ως επέκταση της προηγούμενης δουλειάς παρουσιάστηκε ο μηχανισμός ελέγχου, για ερωτήματα σε βάσεις κινούμενων αντικειμένων, HERMES++ με δημιουργούς τους Pelekis et al. [19]. Ο μηχανισμός καλείται να αντιμετωπίσει τριών ειδών επιθέσεις.

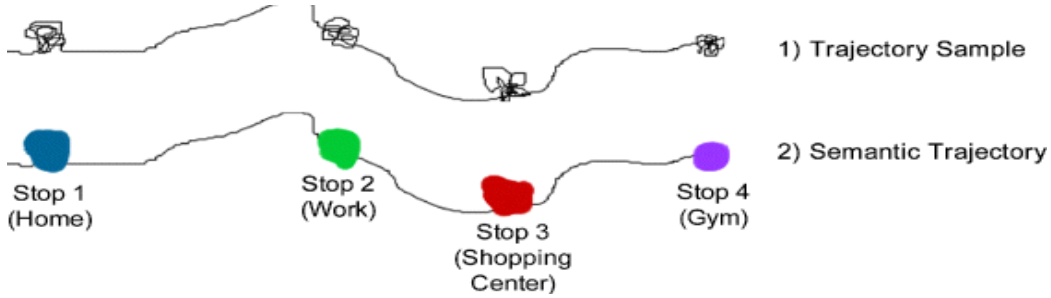
- *User identification attack* όπου ο επιτιθέμενος χρησιμοποιεί ολικώς επικαλυπτόμενα ερωτήματα για να ταυτοποιήσει έναν χρήστη.
- *Sensitive location tracking attack* όπου ο επιτιθέμενος προσπαθεί να εξάγει γνώση αντιστοιχίζοντας και συνδυάζοντας ευαίσθητα σημεία μιας τροχιάς με πρότερη γνώση που έχει για σημεία όπως η αρχή και το τέλος μιας τροχιάς.
- *Sequential tracking attack* κατά την οποία ο επιτιθέμενος προσπαθεί να ακολουθήσει τη διαδρομή του χρήστη κάνοντας στοχευόμενα ερωτήματα σε γειτονικές περιοχές.

Για την προστασία της ιδιωτικότητας και τον χειρισμό των παραπάνω επιθέσεων ο μηχανισμός δεν απαντά σε επικαλυπτόμενα ερωτήματα, προστατεύει την αρχή και το τέλος κάθε τροχιάς με τον ίδιο τρόπο που χειρίζεται τα ευαίσθητα σημεία και δημιουργεί ρεαλιστικές ψεύτικες τροχιές για να καλύπτει το κατώφλι ανωνυμίας που έχει τεθεί.

2.4 Προστασία της ιδιωτικότητας σε βάσεις με σημασιολογικά εμπλουτισμένες τροχιές κινούμενων αντικειμένων

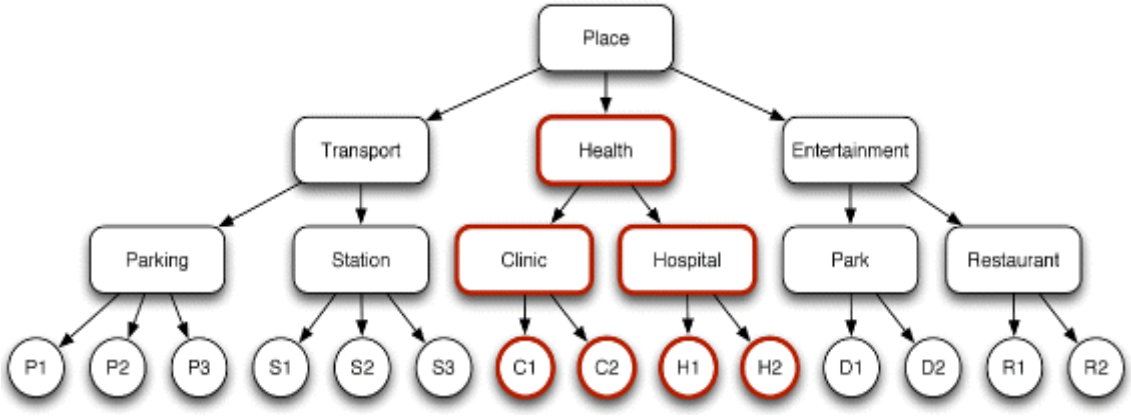
Η εξέλιξη της τεχνογνωσίας και των δεδομένων κίνησης εισήγαγε μια νέα μορφή τροχιών, τις εμπλουτισμένες τροχιές. Στην ουσία πρόκειται για το επόμενο στάδιο των δεδομένων κίνησης που συλλέγονταν μέχρι τώρα, το οποίο έχει εμπλουτιστεί με περισσότερη πληροφορία. Σκοπός πλέον δεν είναι να παρακολουθήσουμε απλά την διαδρομή ενός χρήστη αλλά να δούμε τα σημεία ενδιαφέροντός του μέσα από αυτή. Η κάθε τροχιά αυτού του τύπου [18] τμηματοποιείται

σε *επεισόδια* (*episodes*). Κάθε επεισόδιο χαρακτηρίζεται από μια *χωρική πληροφορία*, ένα *χρονικό διάστημα*, ένα *label* που χαρακτηρίζει το είδος του επεισοδίου (*Stop / Move*) και ένα σύνολο από *tags* τα οποία το εμπλουτίζουν σημασιολογικά με τις πληροφορίες που παρέχουν. Στην Εικόνα 11 παρουσιάζεται ένα παράδειγμα semantic trajectory το οποίο εξάγεται από το sample trajectory και εμφανίζει την επιπλέον πληροφορία ότι η καταγεγραμμένη οντότητα ήταν αρχικά σπίτι, στη συνέχεια πήγε στη δουλειά, έπειτα στο εμπορικό κέντρο και κατέληξε στο γυμναστήριο. Τα χαρακτηριστικά αυτά κάθε επεισοδίου αποτελούν τα δεδομένα επί των οποίων κατά την εκτέλεση ενός ερωτήματος μπορούν να εφαρμοστούν διάφορα κριτήρια. Το ερώτημα που αποστέλλεται στη βάση αποτελείται από ένα ή περισσότερα ανεξάρτητα υποερωτήματα, όπου το καθένα έχει ως κριτήρια τις τέσσερις κατηγορίες δεδομένων που αναφέρθηκαν πριν. Η απάντηση που επιστέφεται αποτελείται από τα κοινά μόνο επεισόδια, μεταξύ των επεισοδίων που θα επέστρεφαν ως αποτέλεσμα αν κάθε υποερώτημα εκτελούνταν ξεχωριστά.



Εικόνα 11 Παράδειγμα Trajectory Sample and Semantic Trajectory – Monreale et al. [15]

Οι επιπλέον πληροφορία που υπάρχει σε κάθε τροχιά βοηθάει την έρευνα και την εξαγωγή χρήσιμων συμπερασμάτων, αλλά όπως είναι φυσικό αυξάνει και την απειλή της ιδιωτικότητας. Οι Monreale et al. [15] πρότειναν για επαυξημένες τροχιές μια μεθοδολογία (*C*-safety) η οποία στηρίζεται στη γενίκευση των δεδομένων, ώστε να επιτευχθεί η απαραίτητη ανωνυμοποίηση. Έκαναν την υπόθεση ότι ο επιτιθέμενος γνωρίζει την διαδικασία ανωνυμοποίησης, την ύπαρξη τροχιάς του ατόμου-στόχου καθώς και ένα σύνολο/ακολουθία από μη ευαίσθητες τοποθεσίες στην τροχιά που τον ενδιαφέρει. Σκοπός της μεθόδου είναι να τροποποιεί την βάση έτσι ώστε να μην μπορεί ο επιτιθέμενος να συμπεράνει κάποια ευαίσθητη τοποθεσία που επισκέφτηκε ο στόχος με πιθανότητα μεγαλύτερη από *C*. Για να γίνει αυτό εφικτό χρησιμοποίησαν μια δενδροειδή ταξινόμηση (Εικόνα 12) των τοποθεσιών όπου τα φύλλα είναι οι τοποθεσίες και όσο ανεβαίνει κάποιος επίπεδο τόσο πιο γενική μορφή παίρνουν.



Εικόνα 12 Δενδροειδής ταξινόμηση τοποθεσιών – Monreale et al. [15]

Η μέθοδος χωρίζεται σε τρία στάδια. Στο πρώτο απαλείφονται τα sensitive σημεία τα οποία για ένα χρήστη είναι σημεία quasi-identifiers. Στην Εικόνα 12 οι σημειωμένες με κόκκινο περιοχές είναι sensitive και πρέπει να προστατευθούν. Έπειτα οι τροχιές ομαδοποιούνται

σύμφωνα με το μήκος τους. Τέλος δημιουργείται μια γενικευμένη μορφή κάθε τροχιάς τροποποιώντας τις quasi-identifier τοποθεσίες σύμφωνα με το δενδροδιάγραμμα ώστε να είναι κοινές σε όλη την ομάδα. Μεγάλο μειονέκτημα της μεθόδου αποτελεί το γεγονός ότι δεν λαμβάνονται υπόψη τα χρονικά δεδομένα των τροχιών, αλλά το μοντέλο στηρίζεται ουσιαστικά σε ένα παγκόσμιο σύνολο από quasi-identifiers και sensitive places το οποίο δίνει μια εξειδικευμένη πηγή η οποία παρέχει την ταξινόμηση των τοποθεσιών.

3 Ο αλγόριθμος Zoom_out

3.1 Σκοπός του αλγορίθμου Zoom_out

Σύμφωνα με την αρχή του *k*-anonymity ένα ερώτημα στη βάση, από έναν εξουσιοδοτημένο χρήστη, το οποίο δεν επιστρέφει ως απάντηση τουλάχιστον *k* τροχιές δεν πρέπει να απαντηθεί. Έχοντας ως στόχο την φιλικότερη προσέγγιση στο χρήστη και την βελτίωση της λειτουργικότητας υλοποιήσαμε έναν αλγόριθμο ο οποίος επιχειρεί να δώσει απάντηση σε τέτοιες περιπτώσεις τροποποιώντας το αρχικό ερώτημα σε ένα παρεμφερές, πάντα σεβόμενος την αρχή του *k*-anonymity [7].

Θα μπορούσαμε να φανταστούμε την διαδικασία αυτή ως μια γενίκευση (zoom out) σε ένα γεωγραφικό χάρτη, από την αρχική περιοχή που έθεσε ο χρήστης σε μια ευρύτερη περιοχή, ικανή όμως να επιφέρει ασφαλή απάντηση στο ερώτημά του. Ουσιαστικά βοηθάμε τον χρήστη να λάβει την πληροφορία που θέλει χωρίς να διενεργεί συνεχόμενα, διευρυμένα ως προς τα κριτήρια, ερωτήματα μέχρι να λάβει απάντηση. Αυτό άλλωστε δημιουργεί την άσκοπη επιβάρυνση στη χρήση της βάσης δεδομένων και την πιθανότητα να μην λάβει ο χρήστης την καλύτερη δυνατή απάντηση.

Ο μηχανισμός λειτουργεί διευρύνοντας ένα ή δύο κριτήρια του κάθε υποερωτήματος του αρχικού ερωτήματος. Η διεύρυνση καθορίζεται από το πώς έχουμε επιλέξει εμείς να χειρίζεται ο αλγόριθμος τέτοιες περιπτώσεις και μπορεί να είναι μόνο χωρική, μόνο χρονική ή χωρική και χρονική ταυτόχρονα δηλαδή χωροχρονική.

3.2 Στοιχεία εισόδου και εξόδου του αλγορίθμου Zoom_out

Ο αλγόριθμος δέχεται ως είσοδο το ερώτημα του χρήστη *Q={SQ1, SQ2,…,SQn}* όπου *SQi* τα υποερωτήματά του, το οποίο γνωρίζουμε πως αν εκτελεστεί επιστρέφει ένα πλήθος τροχιών μικρότερο από το κατώφλι *k* που έχουμε θέσει. Συμπληρωματικά ως είσοδο δέχεται το κατώφλι *k* που θέτουμε ως *k*-anonymity, το βήμα ως αριθμό με το οποίο μεγαλώνουμε σε κάθε επανάληψη τις συντεταγμένες του χώρου (*area_step*), το βήμα ως αριθμό αντίστοιχα για τη χρονική διάρκεια (*time_step*) καθώς και δύο επιπλέον δεκαδικούς αριθμούς *Rmax*, *Rmin* που θα εξηγήσουμε στη συνέχεια τον ρόλο τους. Επίσης λαμβάνει ως είσοδο την πληροφορία μιας συνάρτησης *distortionData* η οποία υποδεικνύει αν κατά την εκτέλεσή του θα τροποποιεί μόνο τις συντεταγμένες του MBB, μόνο το χρόνο ή και τα δύο. Η *distortionData* περιλαμβάνει πιο συγκεκριμένα τρεις Boolean μεταβλητές *distort_area_only*, *distort_time_only*, *distort_area_time*. Πάντα μια από αυτές θα έχει τιμή True και οι άλλες δυο τιμή False. Αν η πρώτη έχει την τιμή True τότε τροποποιείται μόνο το MBB, αντίστοιχα αν η τιμή True είναι στη δεύτερη τροποποιείται μόνο ο χρόνος, ενώ αν True είναι η *distort_area_time* μεταβάλουμε και το χώρο και το χρόνο.

Μετά το τέλος της εκτέλεσής του ο αλγόριθμος έχει ως έξοδο, όχι την απάντηση στο ερώτημα του χρήστη, αλλά ένα νέα ερώτημα, έστω *FQ*. Το νέο ερώτημα έχει ίδιο πλήθος υποερωτημάτων με το αρχικό ερώτημα *Q* και αν η διαδικασία ήταν επιτυχής επιστρέφει ως απάντηση όταν εκτελεστεί τουλάχιστον *k* τροχιές. Κάθε υποερώτημα του *FQ* είναι είτε το ίδιο είτε διευρυμένο συγκριτικά με το αντίστοιχο του ερωτήματος *Q*. Αν κάποιο υποερώτημα έχει

διευρυνθεί αυτό σημαίνει ότι πλέον κάποια κριτήριά του έχουν τροποποιηθεί και αποτελούν υπερσύνολο των αρχικών.

Το αποτέλεσμα λοιπόν του *Zoom_out* είναι ένα νέο ερώτημα το οποίο, αν είναι επιτυχής η διαδικασία και απαντηθεί επιστρέφει πλήθος τροχιών μεγαλύτερο ή ίσο με *k* .

3.3 Distortion Unit και Distortion Limit στον αλγόριθμο Zoom_out

Ο μηχανισμός *Zoom_out* τροποποιεί ένα ή περισσότερα από τα υποερωτήματα του αρχικού *query* του χρήστη μεταβάλλοντας τα κριτήρια χώρου ή χρόνου όπως ήδη αναφέραμε. Όλη η διαδικασία αποσκοπεί στο να συμπεριληφθούν περισσότερα επεισόδια τροχιών σε κάθε υποερώτημα και ως εκ τούτου να ικανοποιηθεί ο κανόνας του *k*-anonymity στην συνολική απάντηση. Για να μπορέσει όμως ο αλγόριθμος να επιλέξει ποιο ή ποια υποερωτήματα θα μεταβάλει και κατά πόσο, καθώς εξετάζει ποιο επεισόδιο είναι προτιμότερο να συμπεριληφθεί, θα πρέπει να έχει τη δυνατότητα να συγκρίνει και να μετράει την αλλοίωση που δύναται να προκύψει για κάθε ένα από τα υποψήφια προς ένταξη επεισόδια. Για το λόγο αυτό χρησιμοποιούνται δύο μονάδες μέτρησης που καλούνται *distortion unit* και *distortion limit*.

Ας υποθέσουμε ότι ένα υποερώτημα πρέπει να τροποποιηθεί για να περιλαμβάνει επεισόδιο μιας τροχιάς. Έστω ότι:

- *SQold*: το υποερώτημα προς τροποποίηση
- *SQnew*: το υποερώτημα όπως θα διαμορφωθεί
- $SQoldx1, SQoldy1$: οι συντεταγμένες του κάτω αριστερού άκρου του MBB του SQold
- $SQoldx2, SQoldy2$: οι συντεταγμένες του πάνω δεξιά άκρου του MBB του SQold
- $SQnewx1, SQnewy1$: οι συντεταγμένες του κάτω αριστερού άκρου του MBB του SQnew
- $SQnewx2, SQnewy2$: οι συντεταγμένες του πάνω δεξιά άκρου του MBB του SQnew
- *Duration()*: η χρονική διάρκεια σε δευτερόλεπτα που έχει ως κριτήριο ένα υποερώτημα

Το MBB του SQold υπολογίζεται ως:

$$OldArea = (SQoldx2 - SQoldx1) * (SQoldy2 - SQoldy1)$$

Αντίστοιχα το MBB του SQnew:

$$NewArea = (SQnewx2 - SQnewx1) * (SQnewy2 - SQnewy1)$$

Η χωρική αλλοίωση υπολογίζεται ως:

$$AreaDistortionUnit = (NewArea - OldArea)/OldArea$$

Η χρονική αλλοίωση υπολογίζεται ως:

$$TimeDistortionUnit = (Duration(SQnew) - Duration(SQold))/Duration(SQold)$$

Η χωροχρονική αλλοίωση υπολογίζεται ως:

$$AreaTimeDistortionUnit = (AreaDistortionUnit + TimeDistortionUnit)/2$$

Ο *Zoom_out* για όλα τα επεισόδια που μπορούν να ενταχθούν σε ένα τροποποιημένο υποερώτημα υπολογίζει το *distortion unit*. Αν έχουμε επιλέξει να κάνουμε μόνο χωρική τροποποίηση τότε *distortion unit* = *AreaDistortionUnit*. Αν έχουμε επιλέξει μόνο χρονική τροποποίηση τότε *distortion unit* = *TimeDistortionUnit* ενώ αν έχουμε ταυτόχρονη μεταβολή χώρου και χρόνου τότε *distortion unit* = *AreaTimeDistortionUnit*. Ο αλγόριθμος επιλέγει το επεισόδιο που θα ενταχθεί με βάση τη λογική του κοντινότερου γείτονα. Κοντινότερος γείτονας ως προς τα κριτήρια ενός υποερωτήματος είναι εκείνο το επεισόδιο που έχει τη μικρότερη *distortion unit* βαθμολογία μεταξύ των υπολοίπων.

Είναι προφανές ότι χρειάζεται ένα συγκεκριμένο όριο στο μέγεθος της αλλοίωσης που μπορούμε να δεχτούμε. Διαφορετικά θα υπήρχαν περιπτώσεις κατά τις οποίες μπορεί ο χώρος ή ο χρόνος να μεγάλωναν τόσο που πλέον να απείχαν κατά πολύ από τα κριτήρια που έθεσε

αρχικά ο χρήστης με το ερώτημά του. Επίσης χωρίς κάποιο όριο θα αντιμετωπίζαμε περιπτώσεις στις οποίες ο αλγόριθμος θα έκανε ατελείωτες επαναλήψεις προσπαθώντας να εντάξει κάποιο επεισόδιο που ίσως να μην δύναται να ενταχθεί. Για αυτούς τους λόγους χρησιμοποιείται μια μεταβλητή που ονομάζουμε *distortion limit*. Όπως θα εξηγήσουμε και στη συνέχεια αναλυτικά η *distortion unit* για κάθε επεισόδιο συγκρίνεται και δεν θα πρέπει να ξεπερνάει την τιμή του *distortion limit*. Σε περίπτωση που ξεπεραστεί η τιμή του ορίου αυτού, το επεισόδιο και δεν υπολογίζεται πλέον ως υποψήφιο για ένταξη.

3.4 **Περιγραφή του αλγορίθμου** Zoom_out

Ο αλγόριθμος *Zoom_out* αποτελείται από δύο διακριτά στάδια. Το πρώτο περιέχει διαδικασίες αρχικοποίησης μεταβλητών. Το δεύτερο στάδιο αποτελεί το κύριο κομμάτι του αλγορίθμου και είναι μια μεγάλη επαναληπτική διαδικασία η οποία χωρίζεται σε δύο επιμέρους τμήματα. Οι επαναλήψεις τερματίζονται όταν ικανοποιείται πλέον το *k*-anonymity ή αν εξετασθούν όλες οι περιπτώσεις και δεν μπορεί να αλλάξει κάτι άλλο ώστε να λάβουμε απάντηση.

Τμήμα 1: Στην αρχή κάθε επανάληψης δημιουργείται ένας πίνακας *hmat* και εκτελείται μια μικρή επανάληψη κατά την οποία κάθε υποερώτημα (*subquery*) εκτελείται μεμονωμένα. Οι τροχιές που εντοπίζονται ως απάντηση σε κάθε subquery εισάγονται στον πίνακα *hmat* μέσω μιας συνάρτησης (*fill*). Κάθε εγγραφή στον πίνακα περιέχει το *id* της τροχιά (*tr_id*), τη συχνότητα εμφάνισής της (*freq*) στο σύνολο των υποερωτημάτων καθώς και μια λίστα (*SQueries*) που περιέχει τα subqueries στα οποία αποτελεί μέρος της απάντησης η τροχιά. Την πρώτη φορά που εισάγεται μια τροχιά ο δείκτης *freq* παίρνει την τιμή 1 και στη λίστα *SQueries* εισάγεται το υποερώτημα που εξετάζεται εκείνη τη στιγμή. Την επόμενη φορά που θα βρεθεί η ίδια τροχιά ως απάντηση σε άλλο υποερώτημα δεν θα εισαχθεί εκ νέου στον πίνακα, αλλά στην ήδη υπάρχουσα εγγραφή θα αυξηθεί ο δείκτης *freq* κατά 1 και στη λίστα της *SQueries* θα προστεθεί το εξεταζόμενο υποερώτημα. Είναι προφανές ότι το πλήθος των στοιχείων της λίστας *SQueries* θα είναι όσο και το *freq*. Όταν συμπληρωθεί ο πίνακας, τουλάχιστον μια εγγραφή δηλαδή μια τροχιά θα έχει *freq* ίσο με το πλήθος των υποερωτημάτων, η οποία είναι και η μέγιστη τιμή που μπορεί να πάρει ο δείκτης της συχνότητας. Τροχιές που έχουν συχνότητα ίση με το πλήθος των υποερωτημάτων είναι κοινές για όλα τα subqueries και αποτελούν μέρος της απάντησης του συνολικού ερωτήματος. Έχουν εντοπιστεί δηλαδή επεισόδιά τους σε όλα τα subqueries.

Στη συνέχεια αυτού του τμήματος ο αλγόριθμος βρίσκει με χρήση της συνάρτησης *get_init_max_freq* την συχνότητα στον πίνακα η οποία δεν έχει τη μέγιστη δυνατή τιμή (*freq*= πλήθος subqueries), αλλά είναι η μεγαλύτερη μεταξύ όλων των υπολοίπων. Μέσω αυτής της συχνότητας και χρησιμοποιώντας τη συνάρτηση *find_next_max_freq_rows* εντοπίζει τις τροχιές του πίνακα που έχουν αυτή τη συχνότητα εμφάνισης. Αυτές οι τροχιές είναι οι πρώτες, επεισόδια των οποίων επιχειρούμε να εντάξουμε στα υποερωτήματα που δεν περιέχονται.

Τμήμα 2: Το τμήμα 2 του δεύτερου σταδίου του αλγορίθμου είναι το κυριότερο κομμάτι και αποτελείται και αυτό από μια επαναληπτική διαδικασία. Αν δεν βρεθεί επεισόδιο κατάλληλο για ένταξη στο ερώτημα από τις αρχικά εξεταζόμενες τροχιές με την επανάληψη αυτή εξετάζονται οι αμέσως επόμενες σε βαθμό *freq* τροχιές. Αυτό γίνεται με τις συναρτήσεις *get_next_max_freq* και *find_next_max_freq_rows*. Η επανάληψη ολοκληρώνεται είτε όταν βρεθεί κατάλληλο επεισόδιο τροχιάς, είτε αν ερευνηθούν όλες οι εναπομείνασες τροχιές του πίνακα και δεν έχει βρεθεί κατάλληλο επεισόδιο προς ένταξη.

Στην επανάληψη αυτή, για τις εκάστοτε εξεταζόμενες τροχιές καλείται η συνάρτηση *compute_rows_distortion*. Μέσω της συνάρτησης αυτής ξεκινάει μια διαδικασία υπολογισμού των *distortion units* για τις τροχιές που επεξεργαζόμαστε. Υπολογίζεται για την εκάστοτε εξεταζόμενη τροχιά τι distortion θα υπάρξει για να συμπεριληφθεί επεισόδιό της σε κάθε υποερώτημα στο οποίο δεν εμφανίζεται. Κάθε φορά ελέγχεται να μην ξεπεραστεί το όριο που θέτουμε από την αρχή ως *distortion limit*. Αν κάποιο επεισόδιο για να συμπεριληφθεί ξεπερνάει το όριο αυτό σταματά η διερεύνηση του και εξετάζουμε την αμέσως επόμενη τροχιά. Όταν έχουν γίνει οι υπολογισμοί όλων των *distortion_unit* επιλέγεται το επεισόδιο με το μικρότερο *distortion*

unit αν έχουν βρεθεί επιτυχώς παραπάνω από ένα υποψήφια προς ένταξη επεισόδια. Τέλος τροποποιείται κατάλληλα το υποερώτημα που θα πρέπει να περιλαμβάνει το επεισόδιο που επιλέχθηκε, αν φυσικά δύναται να γίνει αυτό σύμφωνα με το *distortion_limit* όπως αναφέραμε. Ελέγχοντας ποιο είδος τροποποίησης έχουμε επιλέξει (χωρικό, χρονικό, χωροχρονικό) με την βοήθεια της συνάρτησης *check_by_distortion*, καλείται η αντίστοιχη επαναληπτική διαδικασία (*check_by_distort_area*, *check_by_distort_time*, *check_by_distort_area_time*) στην οποία για κάθε εξεταζόμενη τροχιά, για κάθε subquery που δεν έχει επεισόδιο της τροχιάς, υπολογίζεται ποια θα είναι η μεταβολή των κριτηρίων του και ποιο το *distortion unit* αν τροποποιηθεί έτσι ώστε να περιλαμβάνει πλέον επεισόδιο. Συγκρίνοντας σε κάθε επανάληψη το *distortion unit* με το *distortion limit* που έχει τεθεί, τροχιές που ξεπερνούν το όριο απορρίπτονται και δεν εξετάζονται επιπλέον. Συνεχίζεται η εξέταση της επόμενης τροχιάς. Αυτό συμβαίνει διότι σκοπός είναι να φτάσει η εκάστοτε τροχιά να έχει επεισόδιο σε όλα τα υποερωτήματα. Αν λοιπόν σε κάποιο υποερώτημα ξεπερνάμε το *distortion limit* χωρίς να βρίσκουμε επεισόδιο είναι ανώφελο να συνεχίσουμε τη διερεύνηση της τροχιάς, διότι ανεξάρτητα από τα άλλα υποερωτήματα σε αυτό το συγκεκριμένο δεν θα συμπεριληφθεί ποτέ επεισόδιό της.

Με το πέρας όλων των επαναλήψεων, για κάθε τροχιά που δεν απορρίφθηκε λόγο υπέρβασης του *distortion limit*, έχει δημιουργηθεί, αναλόγως τα κριτήρια που επιλέξαμε εξ αρχής να μεταβάλουμε, από την αντίστοιχη συνάρτηση (*set_subquery_dist_area_unit*, *set_subquery_dist_time_unit*, *set_subquery_dist_area_time_unit*) μια λίστα την οποία ονομάζουμε *SQ_dis_units*. Κάθε εγγραφή αυτής περιέχει το είδος τροποποίησης, το id του subquery που εξετάσθηκε, το *distortion unit* και μια μεταβλητή μετρητή για τις επαναλήψεις που έγιναν όσον αφορά το συγκεκριμένο subquery.

Με βάση αυτή τη λίστα η *check_by_distortion* επιλέγει για κάθε εξεταζόμενη τροχιά την εγγραφή με το μικρότερο *distortion unit*. Αν η τροχιά σε κάποιο έλεγχο ξεπέρασε το *distortion limit* τότε δεν έχει συμπληρωμένη τη λίστα *SQ_dis_units* και σημειώνεται με ένα αρνητικό αριθμό -1. Η *compute_rows_distortion* μη λαμβάνοντας υπόψη της τις εγγραφές μαρκαρισμένες με -1 επιλέγει αυτή με το μικρότερο *distortion unit* και μπορεί πλέον να τροποποιήσει το subquery που δηλώνεται με το id του στη εγγραφή χρησιμοποιώντας την κατάλληλη συνάρτηση μετατροπής (*distort_area*, *distort_time*, *distort_area_time*).

Εμπνεόμενοι από τεχνικές που έχουν εφαρμοστεί για την προστασία της ιδιωτικότητας στα LBS [8] προσπαθούμε να γενικεύσουμε το κάθε MBB που έχει δημιουργήσει ο αλγόριθμος *Zoom_out*, προεκτείνοντάς το προς κάθε κατεύθυνση. Αυτό δημιουργεί μια «ζώνη αβεβαιότητας» περιμετρικά του MBB. Το ποσοστό που επεκτείνεται κάθε συντεταγμένη είναι ένας αριθμός που επιλέγεται κάθε φορά τυχαία, ανάμεσα σε δύο αριθμούς που θέτουμε εξ αρχής εμείς ως διαχειριστές του μηχανισμού. Αυτός ο αριθμός μεταξύ των *Rmin* και *Rmax* που θέτουμε, παράγεται από τη συνάρτηση *random.uniform(Rmin, Rmax)* η οποία εκτελείται στο τμήμα 2 του δεύτερου σταδίου του αλγορίθμου και χρησιμοποιείται από τις *distort_area* και *distort_area_time*. Θεωρούμε πως ο αλγόριθμος είναι γνωστός και εφόσον οι πλευρές του MBB τροποποιούνται κατά το ίδιο βήμα, χρησιμοποιούμε αυτή την τεχνική ώστε να γενικεύσουμε το τελικό τροποποιημένο MBB και να κάνουμε δυσδιάκριτη αυτή την τροποποίηση στον χρήστη. Με αυτό τον τρόπο ενισχύουμε την ασφάλεια και την προστασία της ιδιωτικότητας.

Algorithm – Zoom_Out

Input: k // *k*-anonymity threshold
Q = <SQ_1, SQ2, SQ_3, … ,SQ_n> // αρχικό ερώτημα με τα υπο-ερωτήματά του (n: αρ. υπο-ερωτ)
D = < > // DataBase
H[tr_id, freq, SQueries] // ο βοηθητικός πίνακας H

Output: F_Q = < F_SQ_1, F_SQ_2, F_SQ_3, …, F_SQ_n > // τελικό ερώτημα με τα υπο-ερωτήματά του

1: F_Q ← Q // αρχικοποίηση του final query ως το πραγματικό query Q
2: H ← 0 // αρχικοποίηση του βοηθητικού πίνακα. Αρχικά είναι κενός
3: Ntr ← Count(F_Q) // συνολικός αριθμός αρχικών τροχιών που επιστρέφει ως απάντηση το F_Q
4: **repeat**
5: Something_Changed ← False // αρχικοποίηση
6: **for** i=1 **to** n **do**
7: **begin**
8: **Execute_Query(in** F_SQ_i **out** tr_ids**)**
// επιστρέφει όλες τις τροχιές (tr_ids) που αντιστοιχούν στην εκτέλεση του κάθε υπο-ερωτήματος
9: **Fill_Help_Table(in** H, tr_ids **out** H**)**
// προσθέτει στον πίνακα H τα tr_ids & ανανεώνει το πλήθος των συχνοτήτων
10: **end**
11: **Find_max_freq(in** H, n **out** freq**)**
// βρίσκει το max frequency στον πίνακα H όπου είναι ταυτόχρονα freq < n (όπου n: αρ. υπο-ερωτ)
12: **Find_max_freq_rows(in** H, n, freq **out** rows**)**
// βρίσκει τα tr_ids του πίνακα που έχουν frequency= freq αυτά δηλαδή που έχουν το max frequency
13: episode_found ← False // αρχικοποίηση
14: **repeat**
15: **Compute_Random_Number**(**in** R_{min}, R_{max} **out** R) // υπολογισμός ενός τυχαίου αριθμού R όπου
// R_{min}<R< R_{max}
16: **Compute_Distortion_Units(in** rows **out** tr_id, F_SQ_i, distortion unit, steps**)**
// υπολογίζονται τα distortion units
// και επιλέγεται για κάθε τροχιά το επεισόδιο (και το υπο-ερώτημα) με το μικρότερο distortion
17: **Select_best_candidate_episode(in** tr_id, F_SQ_i, distortion unit, steps **out** F_SQ_i, steps, episode_found)
// γίνεται η επιλογή του πλέον συμφέροντος επεισοδίου
18: **if** episode_found is False **then**
19: **Find_next_max_freq(in** H, n, freq **out** freq**)** // υπολογισμός του επόμενου μεγαλύτερου frequency
20: **Find_max_freq_rows(in** H, n, freq **out** rows**)**
21: **until** episode_found **or** EOF // End Of File
// η επανάληψη τελειώνει αν βρεθεί κατάλληλο επεισόδιο από την προηγ. διαδικασία ή
// αν τελειώσουν τα δεδομένα (τροχιές) του πίνακα
22: **if** episode_found **then**
23: **Embed_New_Episode(in** steps, R **out** F_SQ_i**,** Something_Changed)
// τροποποίηση των κριτηρίων του υπο-ερωτήματος ως προς χώρο και/ή χρόνο,
// βασιζόμενοι στο επεισόδιο που επιλέχθηκε
24: Ntr ← Count(F_Q)
25: **until (not** Something_Changed) **or** (Ntr>=k)
26: **return** F_Q

3.5 Θεωρητικό παράδειγμα λειτουργίας του Zoom_out

Με στόχο την καλύτερη κατανόηση του αλγορίθμου παρουσιάζουμε ένα θεωρητικό παράδειγμα λειτουργίας του με χρήση τυχαίων τιμών. Έστω ότι *k*-anonymity=3, έχουμε αποφασίσει να τροποποιούμε χωρικά τα subqueries (*distort_area_only=True*), *distortion limit*=1.5 και ένας χρήστης θέτει ένα query *Q* το οποίο αποτελείται από 5 subqueries. Η απάντηση στο ερώτημά του περιλαμβάνει μόνο μια τροχιά και επειδή παραβιάζεται ο κανόνας του *k*-anonymity ενεργοποιείται ο μηχανισμός *Zoom_out*. Έστω ότι γεμίζοντας για πρώτη φορά τον πίνακα *hmat* έχουμε το παρακάτω στιγμιότυπο (Πίνακας 1):

tr_id	freq	SQueries
A	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
B	3	[SQ1,SQ3,SQ4]
C	3	[SQ1,SQ2,SQ3]
D	2	[SQ1,SQ4]
E	1	[SQ1]

Πίνακας 1

Η τροχιά A η οποία έχει *freq*=5 είναι προφανώς η κοινή για όλα τα υποερωτήματα τροχιά και είναι η μια τροχιά που αποτελεί την αρχική απάντηση. Αφού *k*=3 αναζητούμε άλλες δύο τροχιές που θα έχουν επεισόδιο σε κάθε subquery. Οι τροχιές B και C είναι οι αμέσως μεγαλύτερες σε συχνότητα και θα είναι οι πρώτες που θα εξετασθούν. Κάθε μια από αυτές δεν έχει επεισόδιό της σε δύο subqueries. Η B στα SQ2, SQ5 και η C στα SQ4, SQ5. Στον επόμενο πίνακα (Πίνακας 2) προσθέτουμε μια στήλη με τα *distortion unit* που υπολογίζονται στην πρώτη επανάληψη μαζί με το υποερώτημα για το οποίο υπολογίστηκαν (οι τιμές είναι τυχαίες για το παράδειγμα):

tr_id	freq	distortion unit
A	5	-
B	3	1.7 SQ2 , -
C	3	0.5 SQ4 , 0.6 SQ5
D	2	-
E	1	-

Πίνακας 2

Παρατηρούμε ότι για την τροχιά B το *distortion unit* που υπολογίστηκε για το SQ2 ξεπερνάει το *distortion limit*. Για το λόγο αυτό η τροχιά απορρίφθηκε και ο αλγόριθμος δεν συνέχισε την διερεύνηση για το *distortion unit* του SQ5 της B τροχιάς. Για την τροχιά C που υπολογίστηκαν επιτυχώς τα *distortion unit* καθώς δεν ξεπερνούν το όριο, το μικρότερο αφορά το SQ4. Ο αλγόριθμος θα τροποποιήσει το SQ4 έτσι ώστε να περιλαμβάνει πλέον επεισόδιο της C τροχιάς. Στη συνέχεια παρουσιάζουμε ένα πιθανό πίνακα (Πίνακας 3) για το πώς έχει διαμορφωθεί πλέον ο πίνακας *hmat*:

tr_id	freq	SQueries
A	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
B	3	[SQ1,SQ3,SQ4]
C	4	[SQ1,SQ2,SQ3,SQ4]
D	2	[SQ1,SQ4]
E	2	[SQ1,SQ4]

Πίνακας 3

Η τροχιά C έχει πλέον *freq*=4. Παρατηρούμε ότι η τροχιά E ενώ είχε *freq*=1 πλέον έχει *freq*=2 και επεισόδιό της εμφανίζεται και στο SQ4. Αυτό συμβαίνει διότι τροποποιώντας και στην ουσία επεκτείνοντας το SQ4 για να εισαχθεί επεισόδιο της C, το νέο subquery εντόπισε και επεισόδιο της τροχιάς E εντός της νέας του περιοχής. Υποθέτοντας ότι στην επόμενη επανάληψη εντάσσεται επιτυχώς επεισόδιο της C και στο SQ5, το οποίο τροποποιείται βλέπουμε τον αντίστοιχο πιθανό πίνακα (Πίνακας 4):

tr_id	freq	SQueries
A	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
B	3	[SQ1,SQ3,SQ4]
C	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
D	2	[SQ1,SQ4]
E	2	[SQ1,SQ4]
F	1	[SQ5]

Πίνακας 4

Έχοντας δύο τροχιές στην απάντηση του ερωτήματος αναζητούμε τουλάχιστον μια ακόμα. Αξίζει να σημειώσουμε ότι στον πίνακα έχει εμφανιστεί μια νέα τροχιά η F μετά την τελευταία επανάληψη και την διεύρυνση του MBB του SQ5. Εκτελώντας την επόμενη επανάληψη ο αλγόριθμος θα προσπαθήσει να εξετάσει την τροχιά με τη μεγαλύτερη συχνότητα (μικρότερη πάντα του *k*=5), η οποία είναι η B. Επειδή όπως και πριν το *distortion unit* της B για το SQ2 ξεπερνάει το *distortion limit*, ο *Zoom_out* θα συνεχίσει με τις επόμενες σε συχνότητα τροχιές οι οποίες είναι οι D και E. Η συνέχεια θα μπορούσε να είναι όπως απεικονίζεται στον Πίνακα 5:

tr_id	freq	distortion unit
A	5	-
B	3	-
C	5	-
D	2	0.8 SQ2, 0.3 SQ3, 0.9 SQ5
E	2	0.5 SQ2, 0.3 SQ3, 0.7 SQ5
F	1	-

Πίνακας 5

Επιλέγοντας ο αλγόριθμος να διευρύνει το SQ3 καθώς και οι δύο τροχιές έχουν για αυτό το μικρότερο *distortion unit* και μάλιστα ίσο μεταξύ τους, το *freq* τους γίνεται 3 και πλέον περιλαμβάνεται επεισόδιό τους και στην απάντηση του τρίτου υποερωτήματος. Συνεχίζοντας την διαδικασία ο αλγόριθμος θα εξετάσει τις τροχιές B, D και E όπου η B γνωρίζουμε πώς θα απορριφθεί και θα διευρυνθεί το SQ2 περιλαμβάνοντας πλέον στην απάντησή του επεισόδιο της τροχιάς E. Στην τελευταία επανάληψη θα τροποποιηθεί το SQ5 και η τροχιά E θα αποκτήσει *freq*=5, δηλαδή θα αποτελεί απάντηση στο συνολικό ερώτημα. Ο Πίνακας 6 θα έχει την εξής μορφή:

tr_id	freq	SQueries
A	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
B	3	[SQ1,SQ3,SQ4]
C	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
D	3	[SQ1,SQ3,SQ4]
E	5	[SQ1,SQ2,SQ3,SQ4,SQ5]
F	1	[SQ5]

Πίνακας 6

Ελέγχοντας ο αλγόριθμος αν ικανοποιείται το *k*-anonymity ή αν πρέπει να προβεί σε επιπλέον επαναλήψεις θα διαπιστώσει ότι ως απάντηση στο συνολικό ερώτημα έχουμε 3 τροχιές οπότε θα σταματήσει επιστρέφοντας το τελικό ερώτημα. Το ερώτημα πλέον περιλαμβάνει το SQ1 όπως το έθεσε ο χρήστης και τα υπόλοιπα subqueries διευρυμένα όσον αφορά το MBB τους από τη συνολική διαδικασία.

4 **Ο αλγόριθμος** Auditor_check_overlapping

4.1 Σκοπός του αλγορίθμου Auditor_check_overlapping

Ο αλγόριθμος *Auditor_check_overlapping* καλείται κάθε φορά που η απάντηση σε ένα ερώτημα του χρήστη καλύπτει το *k*-anonymity (από την αρχή ή μετά τη χρήση του αλγορίθμου *Zoom_out*). Πριν επιστραφεί η απάντηση του τελικού ερωτήματος σε ένα χρήστη, σε μια από τις παραπάνω δύο περιπτώσεις, ο μηχανισμός αυτός ελέγχει συμβουλευόμενος μια βάση δεδομένων όπου αποθηκεύονται τα ερωτήματα κάθε χρήστη, αν για λόγους ασφαλείας πρέπει να του δοθεί η απάντηση ή αν θα πρέπει να υποστεί άρνηση της υπηρεσίας.

Ένας κακόβουλος χρήστης δύναται να εκτελέσει μια επίθεση διενεργώντας μια σειρά από δύο ή περισσότερα ερωτήματα με κάποια κοινά χαρακτηριστικά μεταξύ τους. Στόχος του είναι η απόκτηση επιπρόσθετης γνώσης για μια κατάσταση ή μια καταγεγραμμένη οντότητα στη βάση για ιδιοτελείς σκοπούς.

Η κατηγορία επιθέσεων που ο μηχανισμός που υλοποιήσαμε αποτρέπει είναι τα ολικώς επικαλυπτόμενα ερωτήματα (*overlapping queries*). Μια κατηγορία που έχει απασχολήσει την επιστημονική κοινότητα στο παρελθόν [9], [19] αλλά αφορούσε μη σημασιολογικά επαυξημένες τροχιές. Τα overlapping queries είναι μια ακολουθία τουλάχιστον δύο ερωτημάτων με χαρακτηριστικό γνώρισμα την επικάλυψη των κριτηρίων αυτών. Πιο συγκεκριμένα είτε τα κριτήρια των ερωτημάτων διαφέρουν κατά ένα μόνο είδος (χώρο, χρόνο ή tag) είτε διαφέρουν αριθμητικά κατά ένα ή περισσότερα υποερωτήματα με τα υπόλοιπα υποερωτήματα απολύτως όμοια.

Στόχος μας ήταν να κατασκευάσουμε έναν μηχανισμό που θα ελέγχει το ερώτημα του χρήστη, θα το συγκρίνει με όλα τα προηγούμενα ερωτήματα που έχει θέσει στη βάση δεδομένων και θα εντοπίζει αν υπάρχει:

- Χωρικά ολικώς επικαλυπτόμενο ερώτημα
- Χρονικά ολικώς επικαλυπτόμενο ερώτημα
- Ολικώς επικαλυπτόμενο ερώτημα ως προς τα tags
- Ολικώς επικαλυπτόμενο ερώτημα ως προς το πλήθος των υποερωτημάτων

4.2 Ελαστικοποίηση του αλγορίθμου

Είναι κοινώς αποδεκτό το συμπέρασμα, από προηγούμενες έρευνες της επιστημονικής κοινότητας, ότι είναι επικίνδυνο να επιτρέπονται ολικώς επικαλυπτόμενα ερωτήματα και καλό

είναι να μην δίνεται απάντηση στον χρήστη σε τέτοιες περιπτώσεις. Στην προσπάθειά μας όμως να είμαστε φιλικότεροι προς τον χρήστη και να μην αρνούμαστε την παροχή απάντησης σε ερωτήματα τα οποία, επικαλύπτονται μεν αλλά χωρίς να υπάρχει κίνδυνος παραβίασης της ιδιωτικότητας, υλοποιήσαμε μια ελαστικοποιημένη μορφή αλγορίθμου. Αντλώντας την ιδέα από την εργασία των Adam and Wortmann [3] που αφορούσε στατιστικές βάσεις δεδομένων και μόνο για την περίπτωση των ολικώς επικαλυπτόμενων ερωτημάτων ως προς το πλήθος των υποερωτημάτων, θεωρούμε πως ο χρήστης μπορεί να λάβει απάντηση για ένα ερώτημα το οποίο χαρακτηρίζεται ως overlapping query, αρκεί η αριθμητική διαφορά της απάντησης σε σχέση με το συγκρινόμενο παρελθοντικό query να μην είναι μικρότερη από το κατώφλι *k* που έχει τεθεί για το anonymity. Συγκεκριμένα υπολογίζουμε την απόλυτη διαφορά μεταξύ των τροχιών που δίνεται ως απάντηση στα δύο ερωτήματα και αν αυτή είναι μεγαλύτερη η ίση με το *k*-anonymity τότε ο χρήστης μπορεί να λάβει απάντηση στο ερώτημά του παρά το γεγονός ότι χαρακτηρίζεται ως overlapping query. Στις περιπτώσεις των χωρικών, χρονικών ή ως προς τα tags overlapping queries δεν έχουμε τη δυνατότητα να είμαστε ανεκτικοί και αν ανιχνευτούν ο χρήστης δεν λαμβάνει απάντηση.

4.3 Στοιχεία εισόδου και εξόδου του αλγορίθμου Auditor_check_overlapping

Ο μηχανισμός αυτός δέχεται ως είσοδο το k-anonymity που έχει τεθεί από τους διαχειριστές, το id του χρήστη (*userid*) το οποίο είναι μοναδικό για κάθε ξεχωριστό χρήστη καθώς και το ερώτημα του χρήστη όπως είναι αν δεν χρειάστηκε να το τροποποιήσει ο αλγόριθμος *Zoom_out* ή το τελικό ερώτημα όπως διαμορφώθηκε μετά τη χρήση του *Zoom_out*. Ως έξοδο λαμβάνουμε μια Boolean μεταβλητή η οποία είναι True αν έχουμε ολικώς επικαλυπτόμενο ερώτημα ή False αν δεν υπάρχει ολικώς επικαλυπτόμενο ερώτημα. Επίσης τυπώνεται ένα μήνυμα για το είδος του overlapping query που εντοπίστηκε.

4.4 Περιγραφή του αλγορίθμου Auditor_check_overlapping

Αρχικά ο αλγόριθμος ενεργοποιείται όταν τον καλέσουμε μέσω του *driver* κώδικα με χρήση της συνάρτησης *check_for_overlap*. Στο πρώτο στάδιο εξετάζεται η βάση δεδομένων στην οποία αποθηκεύονται τα ερωτήματα κάθε χρήστη και δημιουργείται μια λίστα (*list_queries_to_exam*) με όλα τα ερωτήματα που έχει κάνει στο παρελθόν ο χρήστης με το συγκεκριμένο *userid*. Τα στοιχεία αυτής της λίστας θα αντιπαραβάλουμε ένα προς ένα με το νέο ερώτημα που επιχειρεί ο χρήστης για να ανιχνεύσουμε πιθανή ολική επικάλυψη. Για κάθε ερώτημα της λίστας μέσω μιας επαναληπτικής διαδικασίας καλούμε τέσσερεις συναρτήσεις, τις *check_spatial_overlap*, *check_time_overlap*, *check_tag_overlap* και *check_overlap_by_subqueries_number*. Κάθε μια από αυτές ελέγχει αν υπάρχει η αντίστοιχη περίπτωση ολικώς επικαλυπτόμενου ερωτήματος. Στην περίπτωση που μια από τις τρεις πρώτες επιστρέψει θετική απάντηση τότε τυπώνεται ένα μήνυμα με το είδος overlapping που έχουμε και ο χρήστης θα υποστεί άρνηση της παροχής της υπηρεσίας. Κρίνεται μη ασφαλές να του δοθεί απάντηση και το ερώτημά του δεν αποθηκεύεται στη βάση των ερωτημάτων. Στην περίπτωση που λάβουμε θετική απάντηση από την τέταρτη συνάρτηση ελέγχου ο μηχανισμός υπολογίζει την απόλυτη διαφορά μεταξύ του πλήθος των τροχιών που έχουν ως απάντηση το νέο ερώτημα και το ερώτημα από την λίστα *list_queries_to_exam*. Αυτό γίνεται λόγω της ελαστικοποίησης του μηχανισμού που περιγράψαμε παραπάνω. Εάν η απόλυτη διαφορά είναι μικρότερη του *k* τότε θα τυπωθεί ένα μήνυμα με το είδος ολικής επικάλυψης που ανιχνεύθηκε και όπως και στις άλλες τρεις περιπτώσεις κρίνεται μη ασφαλές να δοθεί απάντηση στον χρήστη. Αντιθέτως αν η διαφορά δεν είναι μικρότερη του *k* το ερώτημα θα απαντηθεί και το νέο ερώτημα του χρήστη θα αποθηκευτεί στην βάση των ερωτημάτων. Ξεχωριστή θα πρέπει να αναφέρουμε, είναι η περίπτωση στην οποία η λίστα *list_queries_to_exam* είναι άδεια. Σε αυτή την περίπτωση ο συγκεκριμένος χρήστης δεν έχει κανένα αποθηκευμένο query στη βάση ερωτημάτων με το id του (πιθανότατα να είναι η πρώτη φορά που θέτει κάποιο ερώτημα). Σε καταστάσεις όπως αυτή η συνάρτηση

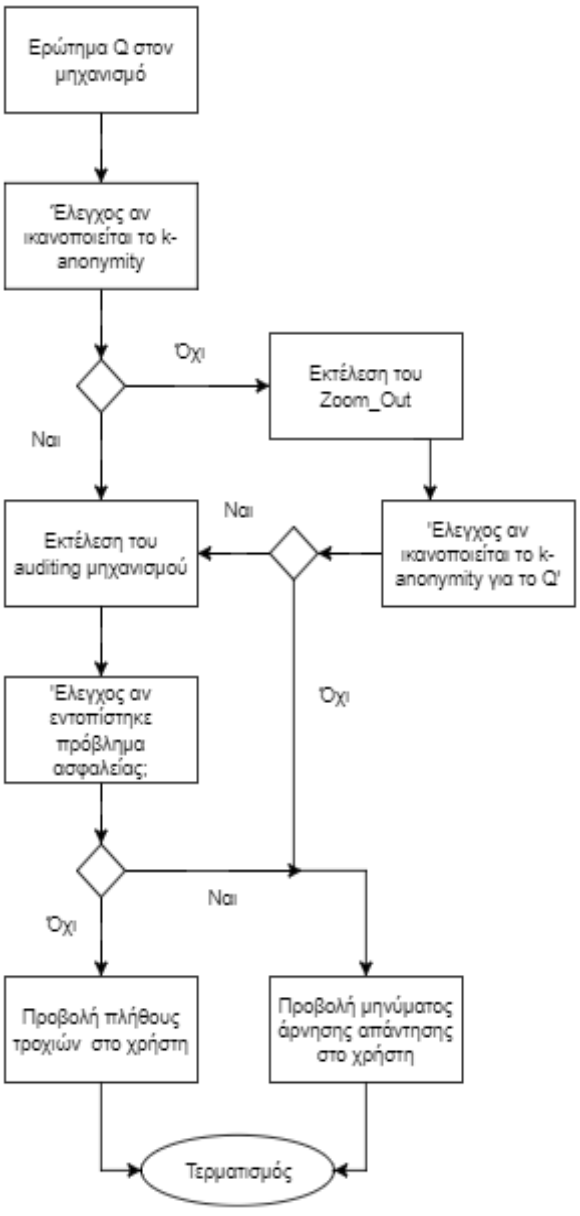
check_for_overlap δεν καλεί τις συναρτήσεις ελέγχου. Επιστρέφει κατευθείαν τη Boolean μεταβλητή ως False, δίδεται απάντηση στον χρήστη και αποθηκεύεται το ερώτημά του.

Algorithm 2 – Auditor

Input: Db = < > // Η βάση δεδομένων που είναι καταχωρημένο το ιστορικό των ερωτημάτων όλων των χρηστών
k // *k*-anonymity threshold
Q = <SQ_1,SQ_2, SQ_3,…, SQ_n> // ερώτημα με τα υποερωτήματά του (n: αρ. υπο-ερωτημάτων)
uid // User identification number στη βάση
Output: Db = < > // Η βάση με ενημερωμένο (αν χρειαστεί) το ιστορικό των ερωτημάτων όλων των χρηστών
is_ok // Boolean μεταβλητή που απαντά αν κρίνεται «αθώο» το ερώτημα ή όχι

```
1:  list_queries_to_exam[] ← 0; check_for_overlap ← False
2:  for each Qi ∈ Db(where user_id=uid) do  // εξέταση όλης της βάσης με τα ερωτήματα
3:  begin
4:    list_queries_to_exam[] ← Qi  // εισαγωγή των ερωτημάτων με το uid στη λίστα list_queries_to_exam
5:  end
6:  for each Qi ∈  list_queries_to_exam[] do  // για κάθε ερώτημα της λίστας κάνε τον έλεγχο σε σχέση με το Q
7:  begin
8:    if sq_num_ Qi = n then  // αν το πλήθος των υποερωτημάτων του Qi είναι ίσο με το πλήθος αυτών του Q
9:      if check_spatial_overlap(in Q, i out True/False) is True then  // αν είναι χωρικά ολικώς επικαλυπτόμενο
10:         print("spatial overlapping")                                // ερώτημα τύπωσε το αντίστοιχο μήνυμα
11:         check_for_overlap = True                                   // και κάνε True το check_for_overlap
12:     else if check_time_overlap(in Q, i out True/False) is True then  // αν είναι χρονικά ολικώς
        //επικαλυπτόμενο ερώτημα τύπωσε το αντίστοιχο μήνυμα και κάνε True το check_for_overlap
13:         print("time overlapping")
14:         check_for_overlap = True
15:     else if check_tag_overlap(in Q, i out True/False) is True then  // αν είναι ολικώς επικαλυπτόμενο
        // ερώτημα ως προς τα tags τύπωσε το αντίστοιχο μήνυμα και κάνε True το check_for_overlap
16:         print("tag overlapping")
17:         check_for_overlap = True
18:     end if
19:   else
20:     if check_overlap_by_subqueries_number(in Q, i out True/False) is True then
        // αν υπάρχει ολική επικάλυψη ως προς το πλήθος τον υποερωτημάτων
21:         if |Count(Q) - Count(Qi)| < k then
22:             print("overlapping by subqueries numbers")
23:             check_for_overlap = True
24:         end if
25:     end if
26:   end if
27: end
28: if check_for_overlap is True then
29:   print("We have overlap so the user must not get an answer")
30:   is_ok ← False
31: else
32:   is_ok ← True
33:   Db ← Q  //η βάση με τα ερωτήματα ενημερώνεται. Το ερώτημα Q αποθηκεύεται.
34: end if
35: return Db, is_ok
```

Γραφική απεικόνιση - Ροή μηχανισμού



5 Παραδείγματα

Στη συνέχεια και με τη βοήθεια στιγμιότυπων που καταγράψαμε παραθέτουμε μερικά παραδείγματα από τη λειτουργία του συνολικού μηχανισμού. Στο σημείο αυτό θα πρέπει να αναφέρουμε ότι χρησιμοποιήθηκε ένα dataset 378131 μη πραγματικών καταγεγραμμένων επεισοδίων τα οποία αποθηκεύσαμε σε μια βάση δεδομένων, με κάθε ένα να παρέχει πληροφορία id τροχιάς, συντεταγμένων, χρονικής διάρκειας του επεισοδίου και tags που το συνοδεύουν. Σε κάθε παράδειγμα θα παρουσιάζουμε το query που θέσαμε εμείς στη θέση του χρήστη, το *userid*, το *k*, αν χρησιμοποιήθηκε ο *Zoom_out*, το είδος τροποποίησης που εκτελεί ο *Zoom_out*, το βήμα ως αριθμό που τροποποιείται κάθε κριτήριο σε κάθε επανάληψη από τον *Zoom_out*, το εκάστοτε *distortion limit*, το αρχικό πλήθος τροχιών που ανιχνεύεται ως απάντηση στο query, το τελικό πλήθος τροχιών αν χρησιμοποιήθηκε επιτυχώς ο *Zoom_out*, αν ανιχνεύθηκε overlapping query και τέλος αν ο χρήστης λαμβάνει ή όχι απάντηση.

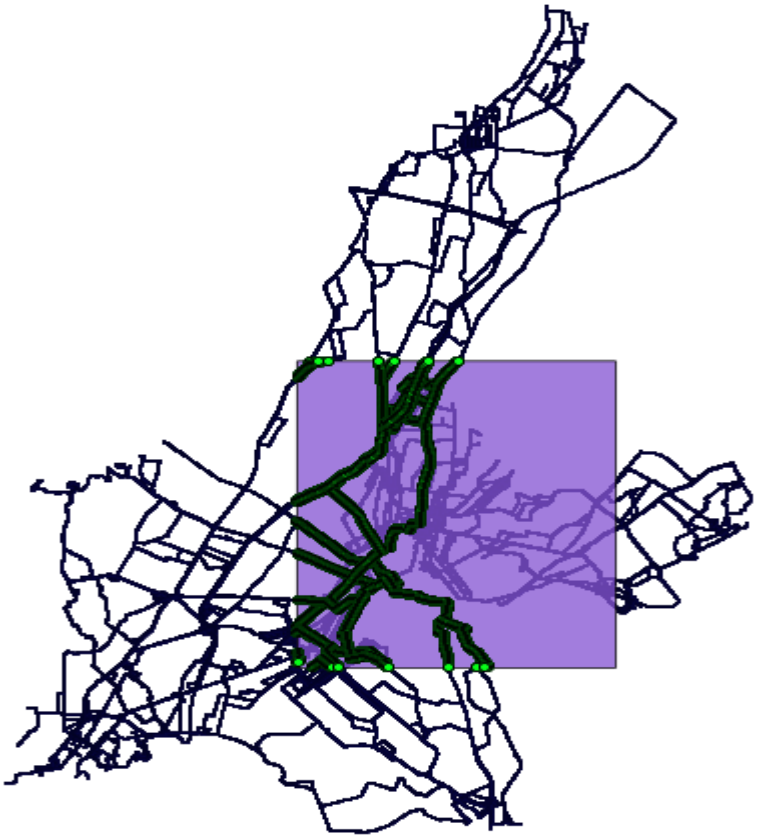
1. userid = 1

   k = 20
   Trajectories as an answer: 64
   Use of Zoom_out: No
   The user can safely get the answer to his query

```
K = 20
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 40000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'CAR'
Q.append(Q_1)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
64
473318.0 4201118.0 480000.0 4209000.0
The user can safely get the answer to his query


Process finished with exit code 0
```

Στην εικόνα 13 παρουσιάζουμε με τη βοήθεια του λογισμικού *QGIS* το σύνολο των στοιχείων ως απεικόνιση στο επίπεδο καθώς και το τμήμα που περιλαμβάνεται ως *Bounding Box* και στο οποίο εκτελείται το ερώτημά μας. Να σημειώσουμε εδώ ότι το σύνολο των σημείων εγγραφών βρίσκονται στο οδικό δίκτυο μιας πόλης.Το *Bounding Box* σημειώνεται με μωβ χρώμα και έχουμε τονίσει τα στοιχεία που αποτελούν απάντηση με πράσινο χρώμα και μεγαλύτερο μέγεθος.



Εικόνα 13 Παράδειγμα 1

2. userid = 1

   k = 20
   Trajectories as an answer: 50
   Use of Zoom_out: No
   Tag overlapping
   We have overlap so the user must not get an answer

```
K = 20
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 40000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'BUS'
Q.append(Q_1)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
50
473318.0 4201118.0 480000.0 4209000.0
tag overlapping
We have overlap the user must not get an answer

Process finished with exit code 0
```

Έχοντας αποθηκεύσει το πρώτο query στη βάση και εκτελώντας με το ίδιο id χρήστη το δεύτερο αλλάζοντας μόνο το *tag* ο μηχανισμός εντοπίζει *tag overlapping* και δεν δίνεται απάντηση στον χρήστη. Επίσης το query δεν αποθηκεύεται αφού κρίθηκε μη ασφαλές. Η εικόνα 14 είναι όμοια με την απεικόνιση του προηγούμενου παραδείγματος και μπορούμε να διακρίνουμε με ροζ χρώμα τις εγγραφές που απαντούν στο ερώτημα καθώς και ότι βρίσκονται στο ίδιο *Bounding Box* με αυτό του ερωτήματος ένα. Αυτό σε συνδυασμό με το ίδιο χρονικό πλαίσιο αναγκάζει τον μηχανισμό να απορρίψει το ερώτημα.

Εικόνα 14 Παράδειγμα 2

3. userid = 1

   k = 20
   Trajectories as an answer: 64
   Use of Zoom_out: No
   Time overlapping
   We have overlap so the user must not get an answer

```
K = 20
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 50000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'CAR'
Q.append(Q_1)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
64
473318.0 4201118.0 480000.0 4209000.0
time overlapping
We have overlap the user must not get an answer

Process finished with exit code 0
```

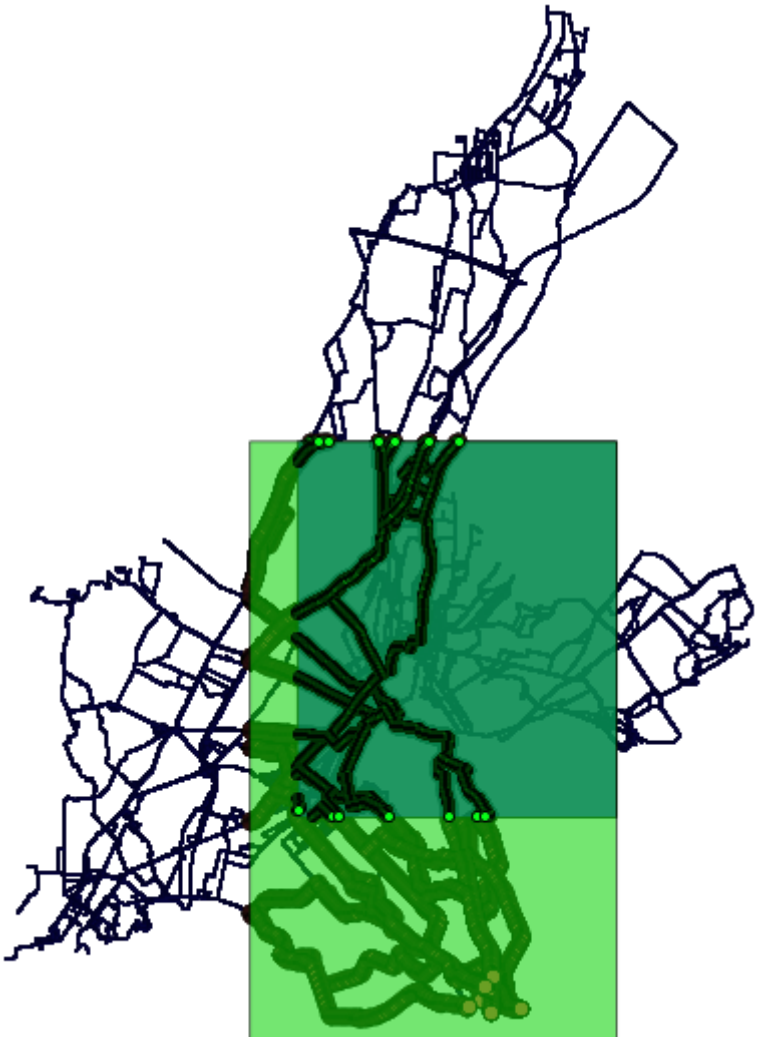
4. userid = 1

   k = 20
   Trajectories as an answer: 68
   Use of Zoom_out: No
   Spatial overlapping
   We have overlap so the user must not get an answer

```
K = 20
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 40000, 472318.0, 4100000.0, 480000.0, 4209000.0)
Q_1.episodes = 'CAR'
Q.append(Q_1)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
68
472318.0 4100000.0 480000.0 4209000.0
spatial overlapping
We have overlap the user must not get an answer

Process finished with exit code 0
```

Στην περίπτωση του παραδείγματος 4 ο ίδιος χρήστης που έθεσε το ερώτημα του παραδείγματος 1 εκτελεί ένα νέο ερώτημα που όπως φαίνεται και από την αναπαράσταση της εικόνας 15 περιέχει χωρικά το πεδίο του πρώτου ερωτήματος. Το ανοιχτό πράσινο είναι το *Bounding Box* του νέου ερωτήματος ενώ το σκούρο πράσινο συμβολίζει το *Bounding Box* του πρώτου ερωτήματος που έχει αποθηκευτεί στην βάση με τα ερωτήματα των χρηστών. Σε αυτή την περίπτωση έχουμε ολική χωρική επικάλυψη και ο μηχανισμός κρίνει ότι δεν είναι ασφαλές να δοθεί απάντηση στον χρήστη.



Εικόνα 15 Παράδειγμα 4

5. userid = 1

   k = 40
   Trajectories as an answer: 64
   Use of Zoom_out: No
   Overlapping by subqueries number
   We have overlap so the user must not get an answer

```
K = 40
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 40000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'CAR'
Q.append(Q_1)

Q_2 = SQuery(2, 500000, 462518.0, 4201118.0, 472000.0, 4206000.0)
Q_2.episodes = 'CAR'
Q.append(Q_2)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
64
473318.0 4201118.0 480000.0 4209000.0 462518.0 4201118.0 472000.0 4206000.0
overlapping by subqueries number
We have overlap the user must not get an answer

Process finished with exit code 0
```

Το query έχει ένα υποερώτημα που είναι ακριβώς ίδιο με το πρώτο query το οποίο εκτελέστηκε από τον ίδιο χρήστη. Η χωρική περιοχή συμπίπτει με την καφέ σκιασμένη περιοχή της εικόνας 16. Το μοβ *Bounding Box* αποτελεί το δεύτερο υποερώτημα. Η απόλυτη διαφορά των απαντήσεων δεν ήταν μεγαλύτερη από το όριο και ο χρήστης δεν θα λάβει ως αποτέλεσμα απάντηση.

Εικόνα 16 Παράδειγμα 5

6. userid = 1

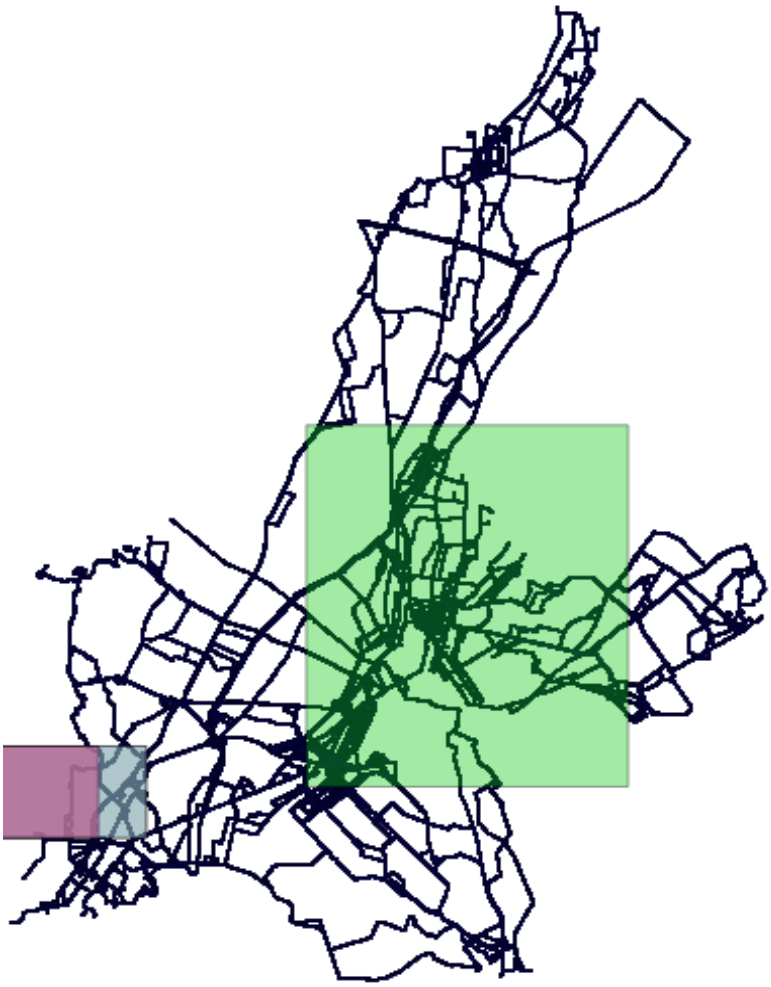
   k = 23
   Trajectories as an answer: 22
   Use of Zoom_out: Yes
   distort_area_only = True
   area_step = 2500.0
   area_distortion_limit = 4.0
   Trajectories as an answer after Zoom_Out = 23
   The user can safely get the answer to his query

```
K = 23
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 40000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'MOVE'
Q.append(Q_1)

Q_2 = SQuery(2, 70000, 460000.0, 4200000.0, 470000.0, 4202000.0)
Q_2.episodes = 'MOVE'
Q.append(Q_2)

Q_3 = SQuery(3, 70000, 450000.0, 4200000.0, 469000.0, 4202000.0)
Q_3.episodes = 'MOVE'
Q.append(Q_3)

# declare the object that handles the distortions

distortionData = DistortionData(2500.0, 10000.0, True, False, False)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
22
[80, 81, 82, 89, 90, 91, 92, 93, 98, 105, 107, 109, 110, 111, 112, 113, 116, 117, 119, 120, 121, 122]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 127,
 128, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177]
[131, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
 118, 119, 120, 121, 122, 123, 126]
[80, 81, 82, 89, 90, 91, 92, 93, 98, 105, 107, 109, 110, 111, 112, 113, 116, 117, 119, 120, 121, 122, 126]
23
[80, 81, 82, 89, 90, 91, 92, 93, 98, 105, 107, 109, 110, 111, 112, 113, 116, 117, 119, 120, 121, 122, 126]
470408.5412523345 4198208.541252335 482909.4587476655 4211909.458747665 460000.0 4200000.0 470000.0 4202000.0 450000.0 4200000.0 469000.0 4202000.0 40000 70000 70000
The user can safely get the answer to his query


Process finished with exit code 0
```

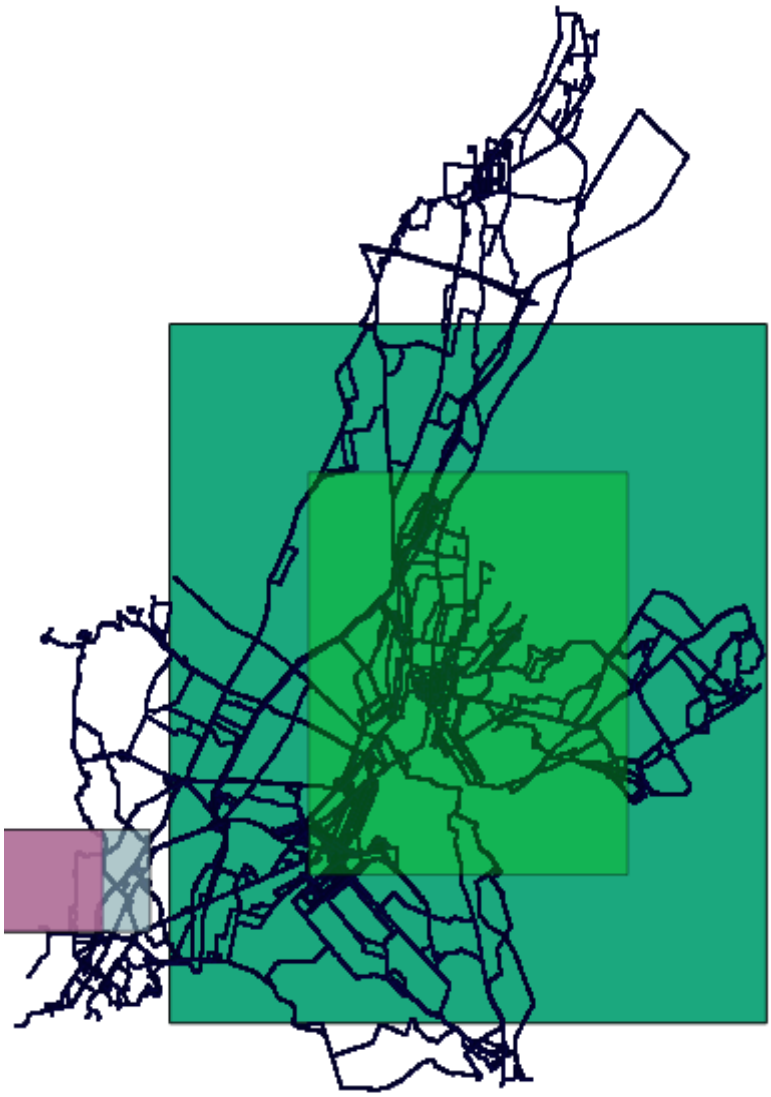
Στο δεύτερο στιγμιότυπο του παραδείγματος βλέπουμε ότι ο αλγόριθμος αρχικά εντοπίζει 22 τροχιές ως απάντηση. Στη συνέχεια βλέπουμε τις 22 αυτές κοινές τροχιές των υποερωτημάτων καθώς και τις τροχιές ξεχωριστά κάθε υποερωτήματος. Μετά τη χρήση του Zoom_Out ο αλγόριθμος βρίσκει 23 τροχιές κοινές. Μπορούμε να δούμε τις συντεταγμένες και το χρόνο από

κάθε *subquery* όπως διαμορφώνεται στο τελικό ερώτημα. Το πρώτο subquery αλλάζει χωρικά και δίνεται απάντηση στον χρήστη.



Εικόνα 17 Παράδειγμα 6

Στην εικόνα 17 παρουσιάζεται ο χάρτης των επεισοδίων και σκιασμένες οι τρεις περιοχές που αντιστοιχούν στα υποερωτήματα. Το πρώτο υποερώτημα χωρικά καλύπτει την πράσινη περιοχή ενώ το δεύτερο τη γαλάζια και το τρίτο την μοβ. Στην συνέχεια παραθέτουμε μια εικόνα έπειτα από την τροποποίηση που έγινε από τον μηχανισμό *Zoom_Out*. Το πρώτο υποερώτημα τροποποιήθηκε και πλέον έχει μεγαλώσει η περιοχή στην οποία κάνει αναζήτηση. Ενδεικτικά έχουμε αφήσει το αρχικό πράσινο *Bounding Box* το οποίο περικλείεται από ένα μεγαλύτερο το οποίο είναι αυτό που προκύπτει με την αλλαγή που υπέστη το αρχικό *sub-query*. Τα άλλα δύο υποερωτήματα δεν άλλαξαν και γι αυτό οι σκιασμένες περιοχές τους παρέμειναν αναλλοίωτες.



Εικόνα 18 Παράδειγμα 6

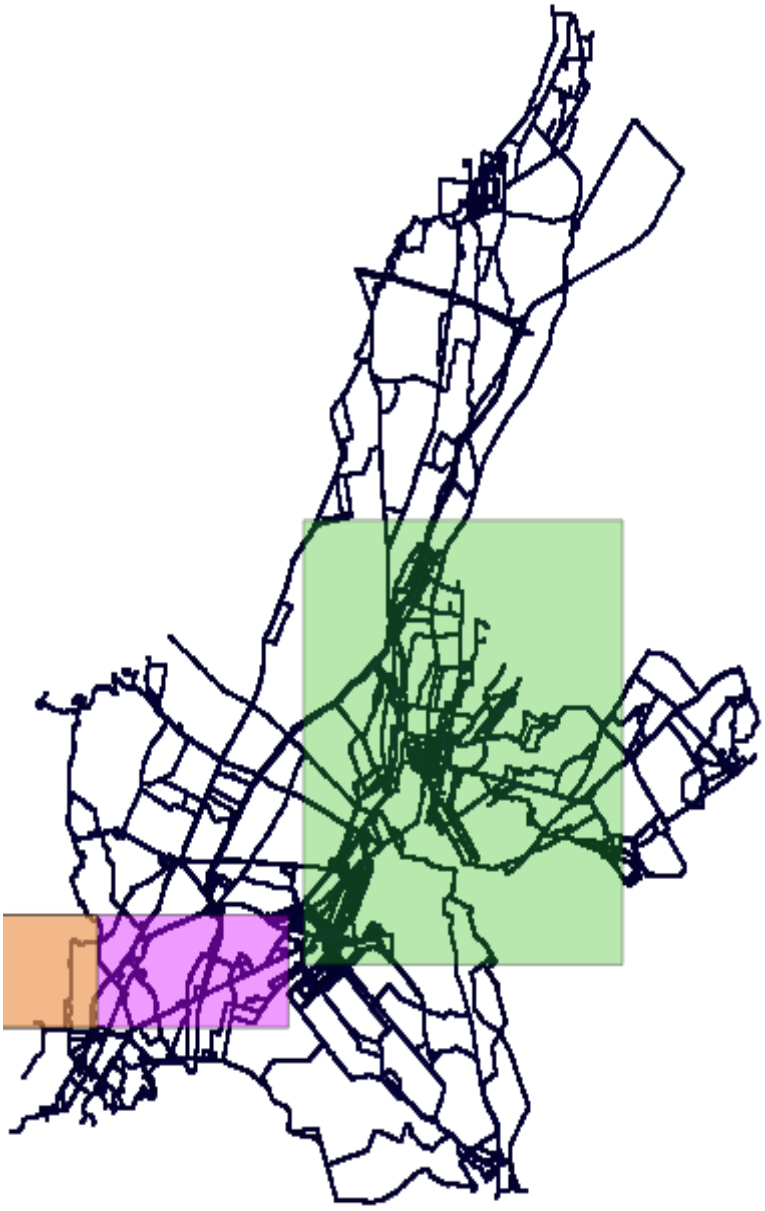
7. userid = 1

   k = 25
   Trajectories as an answer: 17
   Use of Zoom_out: Yes
   distort_area_time = True
   area_step = 2500.0
   time_step = 10000
   area_time_distortion_limit = 2.0
   Trajectories as an answer after Zoom_Out = 51
   The user can safely get the answer to his query

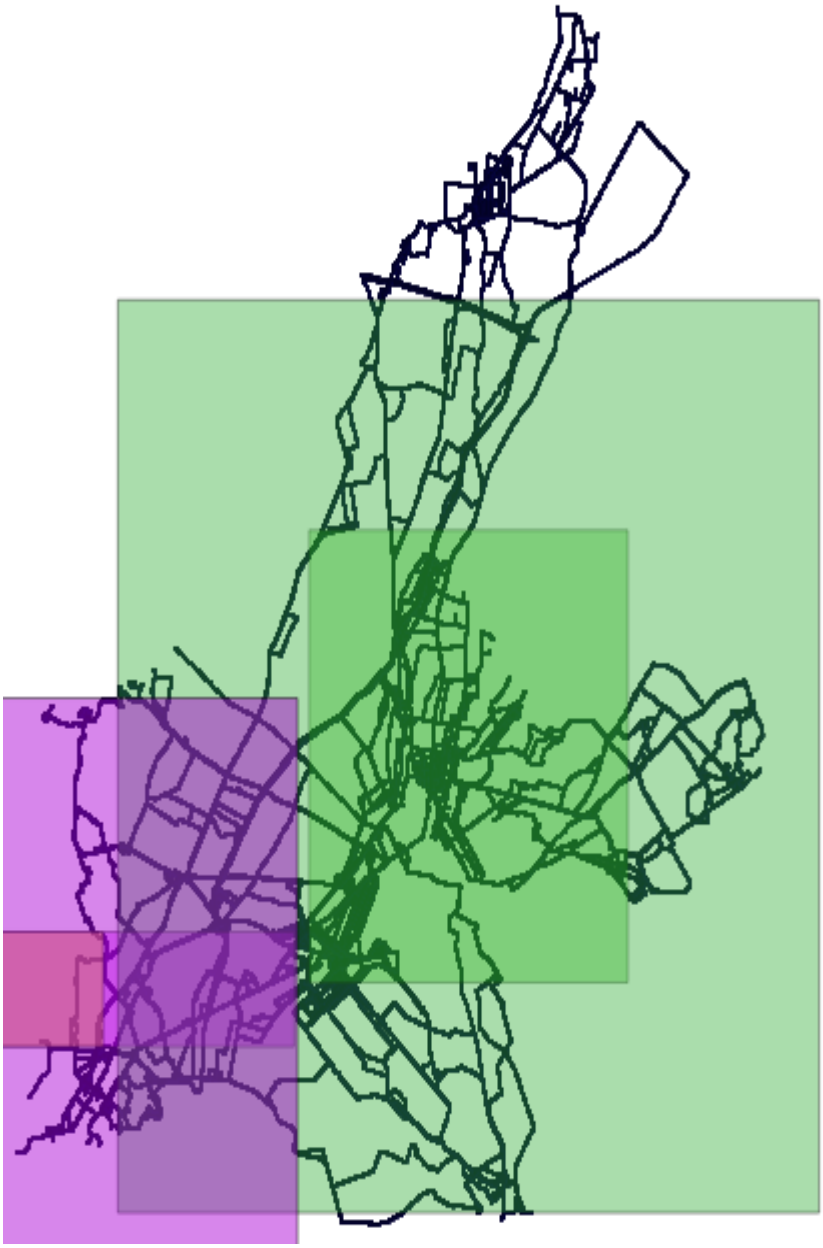
```
K = 25
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 35000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'MOVE'
Q.append(Q_1)

Q_2 = SQuery(2, 33000, 460000.0, 4200000.0, 473000.0, 4202000.0)
Q_2.episodes = 'MOVE'
Q.append(Q_2)

Q_3 = SQuery(3, 46000, 450000.0, 4200000.0, 469000.0, 4202000.0)
Q_3.episodes = 'MOVE'
Q.append(Q_3)

# declare the object that handles the distortions

distortionData = DistortionData(2500.0, 10000, False, False, True)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
17
[121, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 89, 122, 92, 93]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 127,
 128, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 144, 145, 146, 147]
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
[89, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 121, 122, 92, 93, 126]
18
[121, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 89, 122, 92, 93, 126]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177]
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
[89, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 121, 122, 92, 93, 126]
51
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
469305.06366895046 4197105.06366895 484012.93633104954 4213012.93633105 460000.0 4200000.0 473000.0 4202000.0 445927.6148064351 4195927.614806435 473072.3851935649 4206072
 .385193565 45000.0 33000 56000.0
The user can safely get the answer to his query


Process finished with exit code 0
```

Όπως και πριν παραθέτουμε δύο γραφικές απεικονίσεις (εικόνες 19, 20) που δείχνουν τις περιοχές αναζήτησης του αρχικού ερωτήματος καθώς και αυτές όπως τροποποιήθηκαν μετά τη χρήση του αλγορίθμου.



Εικόνα 19 Παράδειγμα 7



Εικόνα 20 Παράδειγμα 7

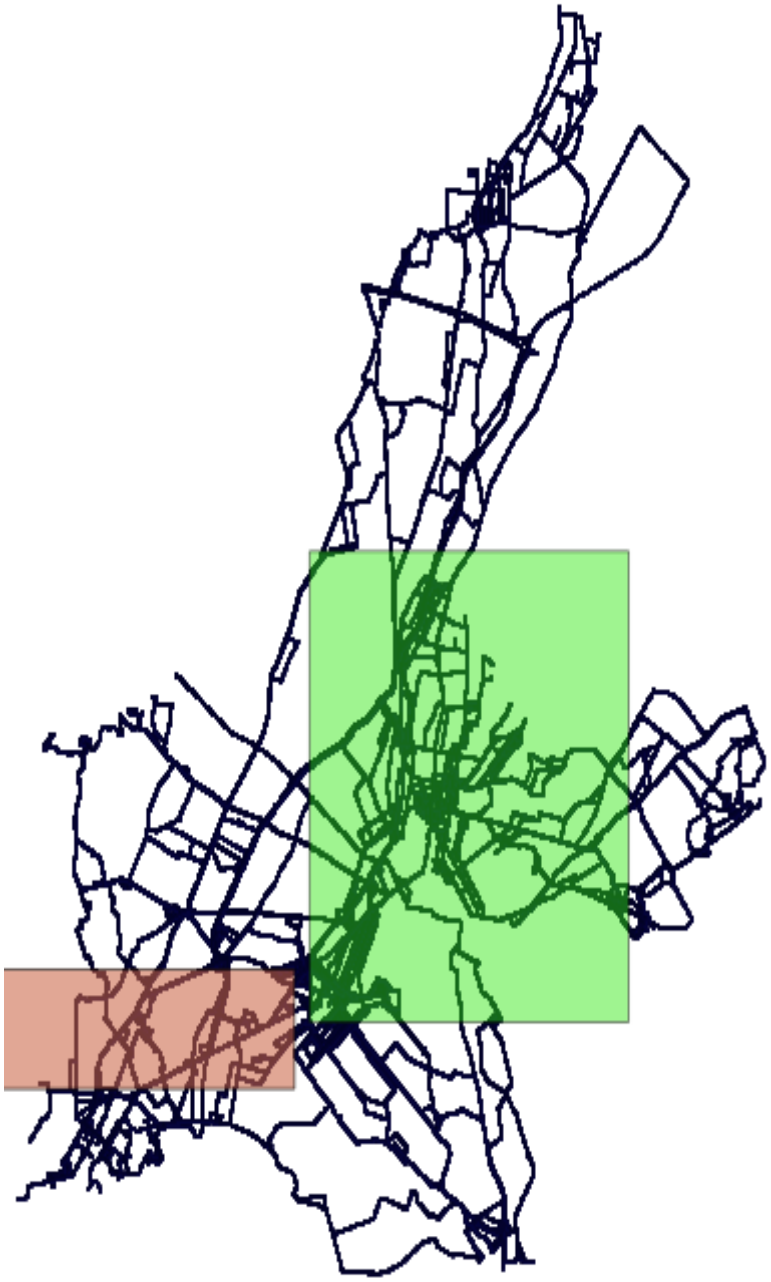
8. userid = 1

   k = 30
   Trajectories as an answer: 1
   Use of Zoom_out: Yes
   distort_time_only = True
   time_step = 10000
   time_distortion_limit = 1.0
   Trajectories as an answer after Zoom_Out = 47
   The user can safely get the answer to his query

```
K = 30
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 31500, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'MOVE'
Q.append(Q_1)

Q_2 = SQuery(2, 33000, 460000.0, 4200000.0, 473000.0, 4202000.0)
Q_2.episodes = 'MOVE'
Q.append(Q_2)

# declare the object that handles the distortions

distortionData = DistortionData(2500.0, 2500.0, False, True, False)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
1
[94]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 137, 129, 60, 61, 62, 63,
 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 144, 94, 146]
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
47
[134, 136, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
 116, 117, 118, 119, 120, 121, 122, 123]
473318.0 4201118.0 480000.0 4209000.0 460000.0 4200000.0 473000.0 4202000.0 41500.0 33000
The user can safely get the answer to his query


Process finished with exit code 0
```

Η εικόνα 21 παρουσιάζει τις περιοχές που εξετάζονται από το ερώτημα. Επειδή σε αυτό το παράδειγμα έχουμε επιλέξει μόνο χρονική τροποποίηση ακόμα και μετά τη χρήση του αλγορίθμου χωρικά οι περιοχές έχουν μείνει αναλλοίωτες.



Εικόνα 21 Παράδειγμα 8

9. userid = 1

   k = 25
   Trajectories as an answer = 17
   Use of Zoom_Out = Yes
   distort_area_time = True
   area_step = 2500.0
   time_step = 10000
   area_time_distortion_limit = 1.5
   Trajectories as an answer after Zoom_Out = -
   Zoom_Out cannot give an answer. The user must not get an answer

```
K = 25
userid = 1
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 35000, 473318.0, 4201118.0, 480000.0, 4209000.0)
Q_1.episodes = 'MOVE'
Q.append(Q_1)

Q_2 = SQuery(2, 33000, 460000.0, 4200000.0, 473000.0, 4202000.0)
Q_2.episodes = 'MOVE'
Q.append(Q_2)

Q_3 = SQuery(3, 46000, 450000.0, 4200000.0, 469000.0, 4202000.0)
Q_3.episodes = 'MOVE'
Q.append(Q_3)

# declare the object that handles the distortions

distortionData = DistortionData(2500.0, 10000, False, False, True)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
17
[121, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 89, 122, 92, 93]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 127,
 128, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 144, 145, 146, 147]
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
[89, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 121, 122, 92, 93, 126]
18
[121, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 89, 122, 92, 93, 126]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177]
[131, 134, 136, 140, 143, 145, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 126]
[89, 98, 105, 109, 110, 111, 112, 113, 82, 80, 116, 117, 119, 121, 122, 92, 93, 126]
Traceback (most recent call last):
  File "C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py", line 43, in <module>
    raise ValueError("zoom out can not give an answer. The user must not get an answer as a result.")
ValueError: zoom out can not give an answer. The user must not get an answer as a result.

Process finished with exit code 1
```

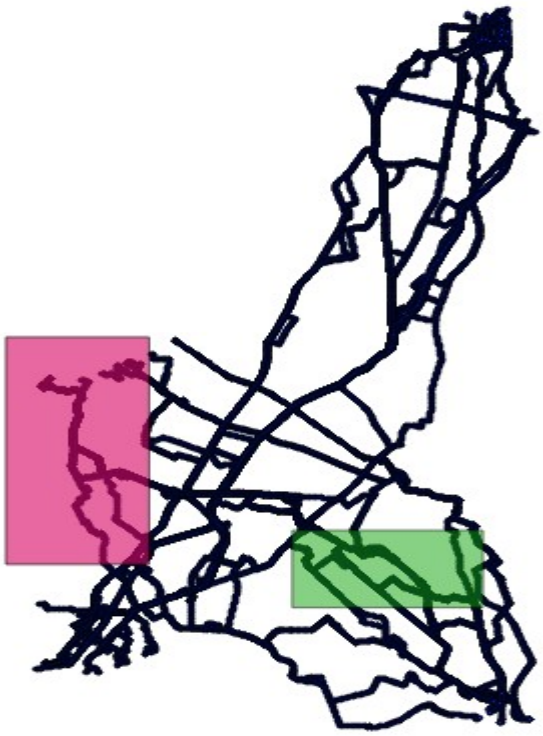
10. userid = 5676

    k = 14
    Trajectories as an answer = 13
    Use of Zoom_Out = Yes
    distort_area_time = True
    area_step = 800.0
    time_step = 2500
    area_time_distortion_limit = 1.0
    Trajectories as an answer after Zoom_Out = 22
    The user can safely get the answer to his query

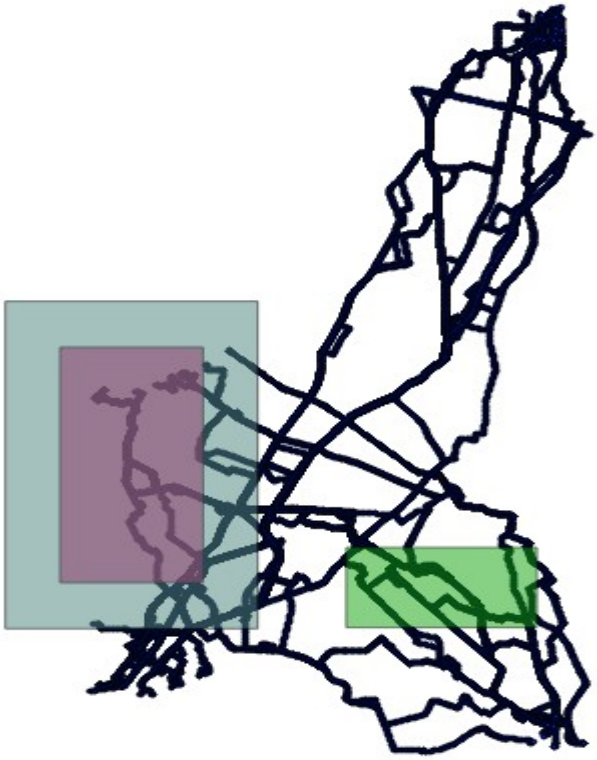
```
K = 14
userid = 5676
Rmax = 1.7
Rmin = 1.0

Q = []

Q_1 = SQuery(1, 50000, 467000.0, 4201118.0, 470000.0, 4207000.0)
Q_1.episodes = 'MOVE'
Q.append(Q_1)

Q_2 = SQuery(2, 50000, 473000.0, 4200000.0, 477000.0, 4202000.0)
Q_2.episodes = 'Null'
Q.append(Q_2)

# declare the object that handles the distortions

distortionData = DistortionData(800.0, 2500.0, False, False, True)
```

```
"C:\Program Files\Python 3.5\python.exe" C:/Users/Chris/PycharmProjects/DiplomaThesis/driver.py
13
[128, 130, 132, 134, 135, 136, 138, 139, 145, 147, 124, 125, 127]
[128, 130, 131, 132, 134, 135, 136, 138, 139, 140, 143, 145, 147, 80, 85, 86, 89, 92, 93, 98, 101, 105, 107, 109, 110, 111, 112, 113, 114, 115, 116, 117, 119, 121, 122, 123, 124,
 125, 126, 127]
[128, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 144, 145, 146, 147, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 82, 96, 124,
 125, 127]
22
[128, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 144, 145, 146, 147, 82, 96, 124, 125, 127]
465853.4699608564 4199971.469960856 471146.5300391436 4208146.530039144 473000.0 4200000.0 477000.0 4202000.0 52500.0 50000
The user can safely get the answer to his query


Process finished with exit code 0
```

Στο συγκεκριμένο παράδειγμα μπορεί να διαπιστώσει κανείς από το πρώτο στιγμιότυπο ότι το δεύτερο subquery έχει ως *tag* την ένδειξη *Null*. Αυτό συμβαίνει όταν ο χρήστης δεν θέσει κάποια λέξη κλειδί. Το υποερώτημα επιστρέφει όλες τις τροχιές που ικανοποιούν τα υπόλοιπα κριτήρια ανεξάρτητα του *tag*. Στις εικόνες 22 και 23 μπορούμε να δούμε τις περιοχές που εξετάζοντας στο αρχικό ερώτημα με πράσινο και μοβ χρώμα και έπειτα την αλλαγή χωρικά που υπέστη το ένα υποερώτημα τονίζοντας με μπλέ χρώμα την νέα περιοχή που καλύπτει ώστε να δοθεί απάντηση στον χρήστη έχοντας τροποποιήσει το ερώτημα. Για το υποερώτημα 1 φυσικά αλλάζει και το χρονικό πλαίσιο αφού τροποποιούμε χώρο και χρόνο.



Εικόνα 22 Παράδειγμα 10

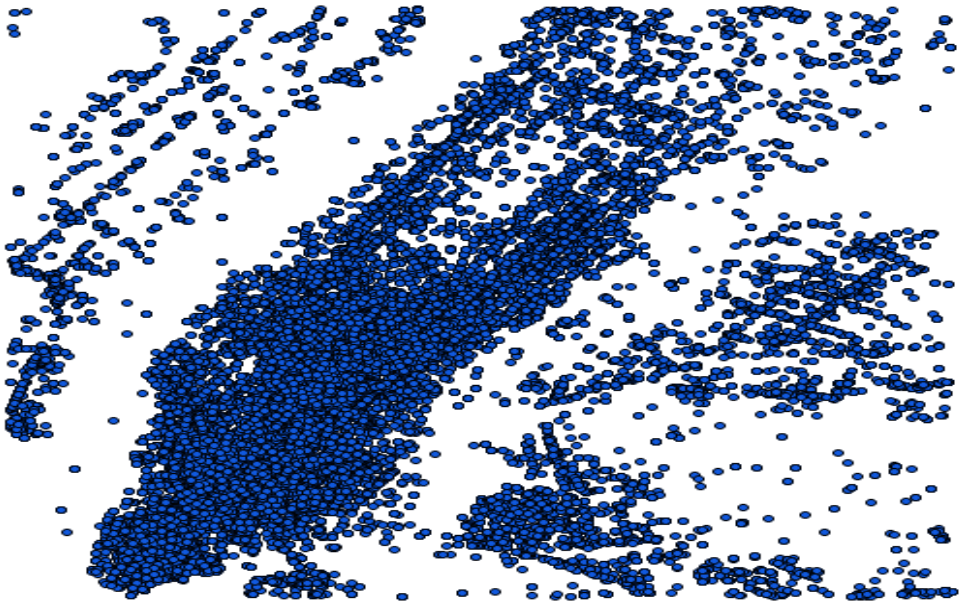


Εικόνα 23 Παράδειγμα 10

6 Πειραματική αξιολόγηση

6.1 Περιγραφή δεδομένων αξιολόγησης

Για την πειραματική αυτή αξιολόγηση χρησιμοποιήθηκε ένα πραγματικό dataset 126509 επεισοδίων που καταγράφηκαν στην πόλη της Νέας Υόρκης και συλλέχθηκαν κατά την περίοδο από 12 Απριλίου 2012 έως 16 Φεβρουαρίου 2013 από την εφαρμογή Foursquare. Κάθε επεισόδιο αποτελεί μια εγγραφή στη βάση δεδομένων και τα επεισόδια εκτελούνται από 1083 ξεχωριστούς χρήστες, οι οποίοι κινούνται σε ένα *Bounding Box* που ορίζεται από τις συντεταγμένες *min latitude* = 40.6995, *max latitude* = 40.8275, *min longitude* = -74.0301 και *max longitude* = -73.8989. Ένα πλήθος 116.8 επεισοδίων κατά μέσο όρο αντιστοιχούν σε κάθε χρήστη που έχει καταγραφεί. Στην εικόνα 13 παρουσιάζεται μια γραφική αναπαράσταση των σημείων στο χώρο όπως αποτυπώνεται με τη βοήθεια του λογισμικού *QGIS*. Κάθε εγγραφή έχει δώδεκα πεδία που προσδίδουν πληροφορίες όπως γεωγραφικές συντεταγμένες, χρονικό πλαίσιο, id χρήστη του επεισοδίου καθώς και πληροφορία με την μορφή *tag* για την τοποθεσία που συλλέχθηκε το επεισόδιο.



Εικόνα 24 Αναπαράσταση επεισοδίων με χρήση του QGIS

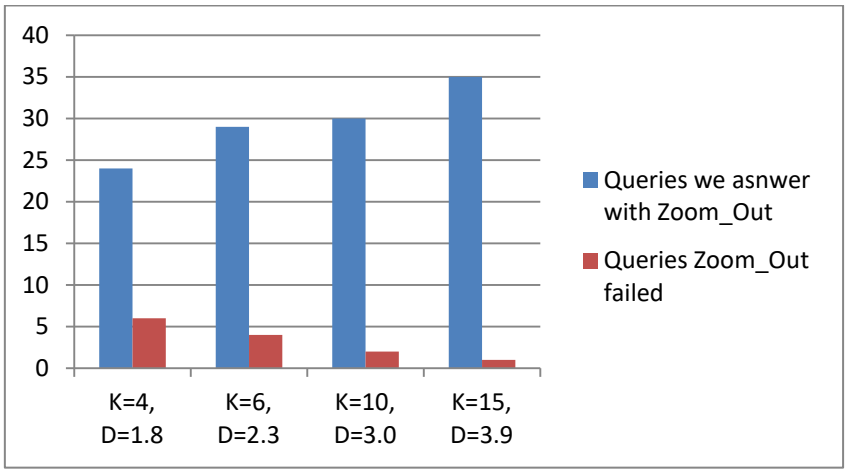
6.2 Εκτέλεση-Ανάλυση πειραμάτων

Με σκοπό να κάνουμε την πειραματική αξιολόγηση εκτελέσαμε ένα πλήθος τυχαίων ερωτημάτων τροποποιώντας δύο βασικά κριτήρια που επηρεάζουν τα αποτελέσματα κατά την εκτέλεση των αλγορίθμων. Αρχικά θα πρέπει να αναφέρουμε πως πρόσβαση στη βάση και δυνατότητα ερωτημάτων προς αυτή θεωρούμε ότι έχουν μόνο εξουσιοδοτημένοι χρήστες. Επίσης θεωρούμε πως το κατώφλι *k* που αποτελεί τον ελάχιστο αριθμό τροχιών που πρέπει να περιέχει μια απάντηση σε οποιοδήποτε ερώτημα καθορίζεται από τον οργανισμό που κατέχει τη βάση δεδομένων και αποτελεί ένα από τα κριτήρια που τροποποιούμε. Το δεύτερο κριτήριο που τροποποιούμε σε συνάρτηση με το κατώφλι *k* είναι το *distortion_limit* που ορίζεται από τον οργανισμό και επηρεάζει την εκτέλεση του αλγορίθμου *Zoom_Out*.

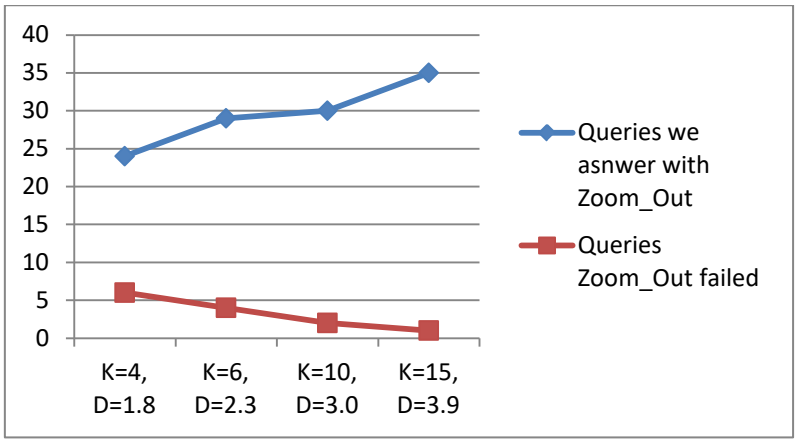
Πιο συγκεκριμένα για να αξιολογήσουμε την αποδοτικότητα του μηχανισμού εκτελούμε τέσσερα πειράματα που περιλαμβάνουν εκατό τυχαία ερωτήματα το καθένα. Σε κάθε πείραμα αυξάνουμε το προκαθορισμένο κατώφλι *k* απαιτώντας ένα μεγαλύτερο πλήθος τροχιών ως

απάντηση για να θεωρηθεί ασφαλές το ερώτημα, αλλά ταυτόχρονα αυξάνουμε και το *distortion_limit* επιτρέποντας στον *Zoom_Out* μεγαλύτερο αριθμό επαναλήψεων. Για κάθε ερώτημα το εκάστοτε υποερώτημά του λαμβάνει με τυχαίο τρόπο το χωρικό διάστημα που θα εξετάσει και ένα χρονικό κατώφλι. Οι πλευρές του MBB κάθε υποερωτήματος υπολογίζονται ως $0.1 * L$ όπου *L* είναι το μήκος της μεγαλύτερης πλευράς του συνολικού MBB του dataset που εξετάζουμε. Το *area_step* με το οποίο αυξάνουμε κάθε φορά τις συντεταγμένες στις επαναλήψεις του αλγορίθμου *Zoom_Out* υπολογίζεται ως $0.001 * L$ και το *time_step* αυξάνεται σε κάθε επανάληψη κατά 900 δευτερόλεπτα. Επίσης με τυχαία επιλογή λαμβάνει ένα *tag* μεταξύ ενός πλήθους που καθορίζουμε εμείς για τα πειράματα. Το σύνολο των *tags* από τα οποία ο αλγόριθμος των πειραμάτων διαλέγει τυχαία τροποποιήθηκε από εμάς. Εδώ θα πρέπει να εξηγήσουμε ότι η τροποποίηση που κάναμε όσο αφορά την επιλογή των *tags* βασίζεται στις ανθρώπινες καθημερινές συνήθειες και στη μορφή των δεδομένων μας με στόχο να έχουμε καλύτερα και πιο αξιόπιστα αποτελέσματα. Επιλέξαμε λοιπόν το πρώτο tag να επιλέγεται από μια λίστα τεσσάρων χαρακτηριστικών και το επόμενο να επιλέγεται από μια δεύτερη λίστα έξι χαρακτηριστικών. Σκοπός μας ήταν να έχουμε επιλογές οι οποίες έχουν αυξημένη εμφάνιση μέσα στο συνολικό dataset και επίσης να αντικατοπτρίζουν μια λογική αλληλουχία στο ερώτημα. Αν για παράδειγμα ένα ερώτημα έχει δύο υποερωτήματα στα οποία το *tag* του πρώτου είναι "*Bar*" δεν δίνουμε την επιλογή το *tag* του δεύτερου να είναι "*Office*". Αυτό γίνεται διότι υπό φυσιολογικές συνθήκες ο μέσος άνθρωπος δεν θα πήγαινε σε ένα μπαρ πριν πάει στη δουλειά του, ενώ το αντίστροφο έχει μεγαλύτερη πιθανότητα να συμβεί και θα το δεχόμασταν. Στη συνέχεια εκτελούμε ξανά τέσσερα πειράματα εκατό τυχαίων ερωτημάτων με τις ίδιες επιλογές και ρυθμίσεις, με την διαφορά ότι το *distortion_limit* παραμένει σταθερό και η μόνη μεταβολή γίνεται στο κατώφλι *k*.

Στα σχεδιαγράμματα των εικόνων 25 και 26 παρουσιάζονται γραφικά τα αποτελέσματα του πρώτου σετ των τεσσάρων πειραμάτων. Με K χαρακτηρίζουμε το κατώφλι του *k*-anonymity ενώ για λόγους συντομογραφίας στα γραφήματα με D χαρακτηρίζουμε το *area_time_distortion_limit*. Όπως αναφέρθηκε το *area_time_distortion_limit* αποτελεί το όριο που ο μηχανισμός εξετάζει και δεν ξεπερνάει όταν είναι αναγκασμένος να μεγαλώσει τα κριτήρια του χώρου και του χρόνου για να βρει επεισόδια τροχιών που θα του επιτρέψουν να τροποποιήσει το αρχικό ερώτημα του χρήστη και να δοθεί στο τέλος ασφαλής απάντηση. Το μπλε χρώμα αντιπροσωπεύει τα ερωτήματα που κρίθηκε ότι δύναται να απαντηθούν μετά από κατάλληλη επεξεργασία από τον μηχανισμό *Zoom_Out*. Το κόκκινο χρώμα αντίστροφα αντιπροσωπεύει τα ερωτήματα που τελικά ο μηχανισμός δεν κατάφερε να τροποποιήσει ώστε να μπορεί να δοθεί απάντηση. Αυτό πρακτικά σημαίνει ότι το όριο *distortion_limit* που είχε τεθεί ξεπεράστηκε σε κάθε περίπτωση πριν μπορέσει ο αλγόριθμος να συμπεριλάβει τα απαραίτητα επεισόδια και ως συνέπεια να καλύψει το όριο του *k*-anonymity για να μπορεί να απαντηθεί το ερώτημα.



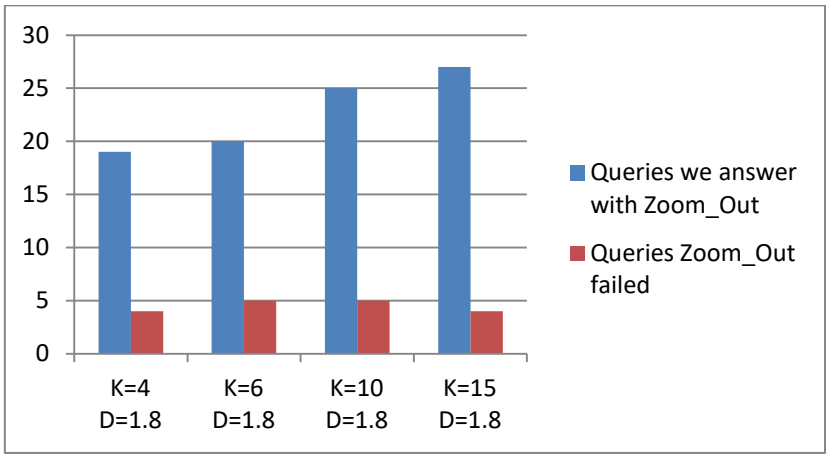
Εικόνα 25



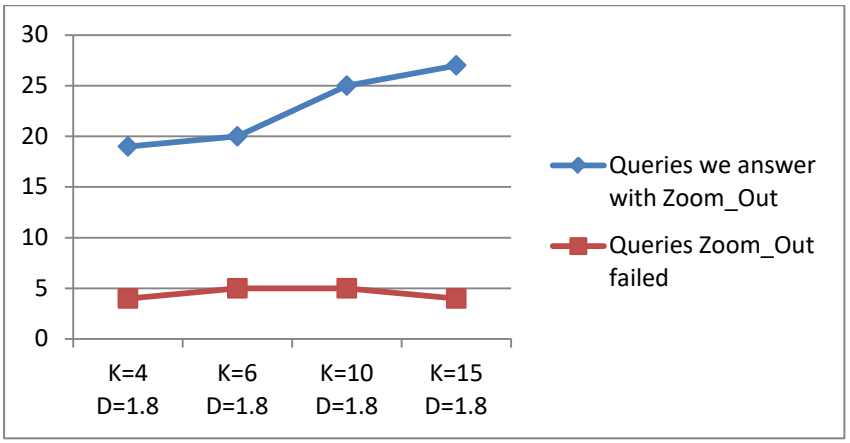
Εικόνα 26

Παρατηρούμε ότι παρόλο που το κατώφλι *k* αυξάνεται, έχοντας τη δυνατότητα περισσότερων επαναλήψεων ο αλγόριθμος γίνεται πιο αποδοτικός, απαντώντας περισσότερα ερωτήματα με μια σχετικά σταθερή αύξηση. Ταυτόχρονα ο αριθμός των ερωτημάτων στα οποία τελικά ο *Zoom_Out* αποτυγχάνει και δεν μπορεί να δώσει απάντηση παρέχοντας ένα τροποποιημένο κατάλληλα ερώτημα μειώνεται διαρκώς. Πιο συγκεκριμένα εκτελώντας 100 ερωτήματα με *k* = 4 και *area_time_distortion_limit* = 1.8 ο μηχανισμός *Zoom_Out* κατάφερε να τροποποιήσει 24 ερωτήματα που η αρχική τους μορφή θα απαγόρευε να δοθεί απάντηση στον χρήστη ενώ παρουσίασε και 6 περιπτώσεις που απέτυχε να καταλήξει σε κατάλληλο ερώτημα. Στη συνέχεια με *k* = 6 και *area_time_distortion_limit* = 2.3 τροποποιήθηκαν 29 ερωτήματα και απέτυχαν να τροποποιηθούν 4. Στο τρίτο πείραμα 100 ερωτημάτων και ενώ θέσαμε τα κριτήρια *k* = 10 και *area_time_distortion_limit* = 3.0 τα ερωτήματα που τροποποιήθηκαν ήταν 30 και 2 μόνο απέτυχαν να αλλάξουν σε μορφή που θα έδινε δυνατότητα απάντησης. Τέλος στο τελευταίο πείραμα με *k* = 15 και *area_time_distortion_limit* = 3.9 τα ερωτήματα που επεξεργάστηκε ο αλγόριθμος επιτυχώς αυξήθηκαν σε 35 ενώ ταυτόχρονα μόλις 1 ήταν αυτό το οποίο δεν κατάφερε να τροποποιήσει. Βλέποντας τα αποτελέσματα συμπεραίνουμε ότι ο αλγόριθμος είναι αποδοτικότερος όταν έχει μεγαλύτερο *distortion_limit* ακόμα και αν το *k*-anonymity έχει αυξηθεί.

Στις εικόνες 27 και 28 παρουσιάζονται γραφικά τα αποτελέσματα του δεύτερου σετ πειραμάτων που εκτελέσθηκαν με σταθερό *area_time_distortion_limit* = 1.8 σε κάθε μια από τις τέσσερεις ομάδες των ερωτημάτων και το *k*-anonymity παίρνει αντίστοιχα τιμές 4, 6, 10 και 15.



Εικόνα 27



Εικόνα 28

Στα συγκεκριμένα πειράματα διαπιστώνουμε ότι για K=4 ο μηχανισμός *Zoom_Out* χρησιμοποιήθηκε και τελικώς δόθηκε απάντηση 19 φορές. Δηλαδή 19 ερωτήματα που αρχικά δεν θα έπρεπε να απαντηθούν τελικά τροποποιούνται επιτυχώς και μπορεί να λάβει απάντηση ο χρήστης. Επίσης 4 είναι οι περιπτώσεις που ο μηχανισμός δεν μπόρεσε να κάνει τροποποίηση που να μην ξεπερνάει το *area_time_distortion_limit* και ουσιαστικά απέτυχε. Στο δεύτερο πείραμα με K=6 τα ερωτήματα που τροποποιεί με επιτυχία ο μηχανισμός μας είναι 20 και αποτυγχάνει μόνο σε 5 περιπτώσεις. Για K=10 ο *Zoom_Out* καταφέρνει να τροποποιήσει και να δοθεί απάντηση σε 25 ερωτήματα ενώ 5 ερωτήματα κρίνονται μη ασφαλή καθώς ο μηχανισμός δεν μπόρεσε να τα τροποποιήσει με τρόπο που να μην ξεπερνάει το *area_time_distortion_limit*. Στο τελευταίο πείραμα με K=15 το σύνολο των επιτυχώς τροποποιημένων ερωτημάτων είναι 27 και των αποτυχιών 4. Στο δεύτερο αυτό σετ των τεσσάρων πειραμάτων παρατηρείται μια αύξηση των περιπτώσεων επιτυχίας χρήσης του μηχανισμού όσο το κατώφλι *k* αυξάνεται και μια σταθερότητα στις περιπτώσεις αποτυχίας του. Αυτό το γεγονός της αύξησης το οποίο μπορεί αρχικά να φαντάζει περίεργο μπορεί να εξηγηθεί. Όσο αυξάνουμε το κατώφλι *k* είναι αναμενόμενο οι περιπτώσεις στις οποίες ο χρήστης θα έπαιρνε απάντηση με το αρχικό του ερώτημα να μειώνονται καθώς απαιτούνται περισσότερες κοινές τροχιές για τα υποερωτήματα. Το γεγονός αυτό άμεσα σημαίνει ότι ο Zoom_Out θα καλείται αντίστοιχα πολύ περισσότερες φορές για να επεξεργαστεί ερωτήματα. Συνέπεια αυτής της αλληλουχίας είναι να κρίνεται αποδοτικός ο αλγόριθμος και να παρουσιάζει αύξηση των περιπτώσεων που επεξεργάζεται και δίνει τη δυνατότητα απάντησης όσο αυξάνεται η τιμή του *k*-anonymity.

7 **Η βάση δεδομένων** MongoDB

Στο σημείο αυτό θα θέλαμε να αναφερθούμε στη βάση δεδομένων που χρησιμοποιήσαμε στην υλοποίησή μας και στα παραδείγματα που εκτελέσαμε. Όπως αναφέραμε και νωρίτερα μας δόθηκε ένα συνθετικό (κατασκευασμένο) και ένα πραγματικό dataset. Εμείς τα αρχεία αυτά για να τα χειριστούμε χρειαζόμασταν δύο αντίγραφα μια βάση δεδομένων, στα οποία περάστηκαν όλες οι εγγραφές τους. Επιπροσθέτως χρειαζόμασταν μια βάση για να αποθηκεύονται τα queries που κάνει ένας υποτιθέμενος χρήστης και να μπορεί να λειτουργήσει ο αλγόριθμος *Auditor_check_overlapping*. Επιλέξαμε να χρησιμοποιήσουμε την MongoDB η οποία είναι μια εύχρηστη, *open-source* βάση δεδομένων που προσφέρει υψηλή απόδοση, υψηλή διαθεσιμότητα και αυτόματη κλιμάκωση (*scaling*). Κάθε εγγραφή στη MongoDB είναι ένα

document, το οποίο είναι μια δομή δεδομένων αποτελούμενη από ζεύγη πεδίων και τιμών. Τα document είναι παρόμοια με JSON objects. Οι τιμές των πεδίων μπορούν να περιέχουν άλλα documents, *arrays* ή arrays από documents. Ένας επιπλέον λόγος που επιλέξαμε τη MongoDB ήταν τα πολύ χρήσιμα modules που έχει ώστε να διευκολύνει τα queries που αφορούν εγγραφές με *geospatial* δεδομένα σε συγκεκριμένο σχήμα. Αναφέρουμε συγκεκριμένα τα *$geoWithin* και *$box* ο συνδυασμός των οποίων μας επέτρεψε να προσδιορίζουμε τα MBB που θέλαμε. Η βάση δεδομένων αποθηκεύει τα documents σε *collections*. Εμείς δημιουργήσαμε δύο collections στη βάση μας (*FirstCollection*, *QueriesCollection*) αποθηκεύοντας στην πρώτη τα επεισόδια των τροχιών και στην δεύτερη τα queries που εκτελεί ο χρήστης.

8 **Κώδικας υλοποίησης και** modules

Η υλοποίηση της παρούσας εργασίας έγινε σε γλώσσα python. Δημιουργήσαμε έξι αρχεία και ένα επιπρόσθετο script αρχείο το οποίο εκτελεί την εισαγωγή των δεδομένων από csv αρχείο στη βάση MongoDB. Το script αρχείο καλείται *create_mongo_db.py* και χρειάζεται να εκτελεστεί αυτόνομα μόνο μία φορά, όταν δημιουργείται η βάση δεδομένων. Το αρχείο *driver.py* είναι αυτό το οποίο εκτελείται κάθε φορά που θέτουμε ένα query και χρησιμοποιεί τα υπόλοιπα όπου χρειάζεται να τα καλέσει. Τα αρχεία *query.py*, *zoom_out.py*, *distortion.py* και *constants.py* περιέχουν τον κώδικα υλοποίησης του μηχανισμού *Zoom_out*, ενώ το αρχείο *auditor_check_overlapping.py* παρουσιάζει την υλοποίηση του auditor μηχανισμού. Θα θέλαμε τέλος να αναφέρουμε ότι στον κώδικα έχουν χρησιμοποιηθεί τα *modules random* και *pymongo* τα οποία έχουν γίνει import στην αρχή των κατάλληλων αρχείων.

9 **Συμπεράσματα**

Η τεχνολογική πρόοδος στις συσκευές, στα λειτουργικά συστήματα, στις εφαρμογές καθώς και η εκτεταμένη χρήση δεδομένων καταγραφής κίνησης οδήγησε στην εμφάνιση μιας νέας μορφής τροχιών κίνησης οι οποίες εκτός από χωρική και χρονική πληροφορία εμπλουτίζονται σημασιολογικά με πληροφορίες, καθιστώντας τις πολύ χρήσιμες αλλά ταυτόχρονα ιδιαίτερα ευαίσθητες. Η αποφυγή παραβίασης της ιδιωτικότητας των καταγεγραμμένων οντοτήτων, κατά τη χρήση βάσεων με τροχιές αυτού του τύπου, αποτελεί πολύ σημαντικό και δύσκολο στόχο, ιδιαίτερα υπό το πρίσμα της χρήσης των δεδομένων από πολλούς χρήστες για διαφορετικούς σκοπούς.

Με κύριους άξονες την προστασία της ιδιωτικότητας και την αύξηση της λειτουργικότητας στο μέγιστο δυνατό υλοποιήσαμε μια μηχανή ελέγχου και επεξεργασίας ερωτημάτων. Ο μηχανισμός προσπαθεί να αυξάνει τη φιλικότητα στον χρήστη τροποποιώντας και απαντώντας τα ερωτήματα στα οποία δεν θα έπρεπε αρχικά να λάβει απάντηση, σεβόμενοι ωστόσο τις αρχές διατήρησης της ιδιωτικότητας και εκτελώντας τους απαραίτητους ελέγχους.

Αναπτύχθηκαν δύο βασικοί αλγόριθμοι ο *Zoom_Out* ο οποίος φαντάζει ως ένα ζουμάρισμα-απομάκρυνση στα κριτήρια που πλαισιώνουν το εκάστοτε ερώτημα και ο αλγόριθμος *Auditor_check_overlapping* ο οποίος ενισχύει την ασφάλεια ανιχνεύοντας και αποτρέποντας πιθανές περιπτώσεις επίθεσης με τη μορφή overlapping ερωτημάτων.

Μελλοντικές εργασίες θα μπορούσαν να εστιάσουν στην βελτίωση της απόδοσης των αλγορίθμων του μηχανισμού, κυρίως επεξεργαζόμενοι και θέτοντας ένα πλαίσιο στην παράμετρο του χρόνου εκτέλεσης που απαιτεί ο μηχανισμός. Επίσης ένας τομέας ο οποίος δύναται να εμπλουτιστεί είναι τα είδη των επιθέσεων που ανιχνεύει-καλύπτει ο μηχανισμός που υλοποιήθηκε. Τέλος θα αποτελούσε ενδιαφέρουσα επέκταση η αντικατάσταση των MBB που χρησιμοποιεί και εξετάζει ως γεωγραφικές περιοχές ο *Zoom_Out* από κάποιο άλλο είδος γεωγραφικής περιοχής όπως πολύγωνα ή κελιά με μη καθορισμένο γεωμετρικό σχήμα.

Πίνακες συναρτήσεων σε κάθε αρχείο του κώδικα

query.py	zoom_out.py	distortion.py
distort_area	get_trace_id	find_subqueries_not_in_row
distort_time	subquery_exists	set_subqueries
distort_area_time	set_subquery_dist_area_unit	check_by_distort_area
get_results	set_subquery_dist_time_unit	check_by_distort_time
combine	set_subquery_dist_area_time_unit	check_by_distort_area_time
conversion	get_min_area_dist_unit	check_by_distortion
	get_min_time_dist_unit	compute_rows_distortions
	get_min_area_time_dist_unit	
	get_row_with_trace_id	
	fill	
	rows	
	get_init_max_freq	
	get_next_max_freq	
	find_next_max_freq_rows	
	zoom	

auditor_check_overlapping.py
check_spatial_overlap
check_time_overlap
check_tag_overlap
check_overlap_by_queries_numbers
check_for_overlap

9 Βιβλιογραφία

[1] Abul O, Bonchi F, and Nanni M. Never walk alone: Uncertainty for anonymity in moving objects databases. In Proceedings of ICDE, pages 376-385, 2008
[2] Abul O, Bonchi F, and Nanni M. Anonymization of moving objects databases by clustering and perturbation. Information Systems, 35(8):884-910, 2010
[3] Adam N.R and Wortmann J.C. Security-control methods for statistical databases: A comparative study. ACM Computing Surveys, 21(4):515-556, 1989
[4] Chen L, Ozsu M.T and Oria V. Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05), 2005
[5] Chow C.Y and Mokbel M.F. Trajectory privacy in location-based services and data publication. ACM SIGKDD Explorations Newsletter 13(1), pp.19-29. 2011
[6] Gedik B and Liu L. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In Proceeding of the 25th International Conference on Distributed Computing Systems ICDCS '05
[7] Giotakis Spyridon Privacy preservation for semantically enriched trajectory databases using query auditing techniques, Piraeus, July 2015
[8] Gkoulalas-Divanis A, Kalnis P, Verykios V.S. Providing k-anonymity in location based service. ACM SIGKDD Explorations Newsletter 12(1), pp.3-10. 2010
[9] Gkoulalas-Divanis A, Verykios V.S. A privacy-aware trajectory tracking query engine. SIGKDD Explorations, 10(1):40-49, 2008
[10] Hoh B and Gruteser M. Protecting location privacy through path confusion. In SECURECOMM, pages 194-205, 2005
[11] Jin Wang, Hao Wu, Yudian Liu A New Distributed User-Demand-Driven Location Privacy Protection Scheme for Mobile Communication Network, 20 September 2015
[12] Li N, Li T, Venkitasubramanian S. t-closeness: Privacy beyond k-anonymity and l-diversity. Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, 2007
[13] Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramanian M. l-diversity: Privacy Beyond k-anonymity. In Proceedings of ICDE, pp.24, 2006
[14] Mahdavifar S, Abadi M, Kahami M, Mahdikhani H. A clustering-based approach for personalized privacy preserving publication of moving object trajectory data. Network and System Security. Springer Berlin Heidelberg, 149-165, 2012
[15] Monreale A, Trasarti R, Pedreschi D, Renso C, Bogomy V. C-safety: a framework for the anonymization of semantic trajectories. Transactions on Data Privacy, vol. 4, no. 2, pp. 73-101, 2011
[16] Monreale A, Andrienko G, Andrienko N, Giannotti F, Pedreschi D, Rinzivillo S, Wrobel S. Movement Data Anonymity through Generalization. Transactions on Data Privacy 3(2), 91-121, 2010
[17] Nergiz M. E, Atzori M and Saygin Y. Towards trajectory anonymization: A generalization-based approach. Transactions on Data Privacy 2(1): 47-75, 2009
[18] Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., and Yan, Z. Semantic trajectories modeling and analysis. ACM Comput. Surv. 45, 4, Article 42, August 2013
[19] Pelekis N, Gkoulalas-Divanis A, Vodas M, Kopanaki D, Theodoridis Y. Privacy Aware Querying over Sensitive Trajectory Data. In Proceedings of CIKM, pp.895-904, 2011
[20] Sweeney L. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10.05: 557-570, 2002
[21] Terrovitis M, Mamoulis N. Privacy preservation in the publication of trajectories. In MDM, pp.65-72, 2008