ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία χωρικών ρευμάτων δεδομένων με διαγράμματα Voronoi

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΗΝΟΓΙΑΝΝΗ ΘΕΟΦΑΝΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία χωρικών ρευμάτων δεδομένων με διαγράμματα Voronoi

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΗΝΟΓΙΑΝΝΗ ΘΕΟΦΑΝΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6η Ιουνίου 2007.

.....	.....	.....
Τ. Σελλής	Α-Γ. Σταφυλοπάτης	Α. Παγουρτζής
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούνιος 2007

ΘΕΟΦΑΝΗΣ Γ. ΜΗΝΟΓΙΑΝΝΗΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

στους γονείς μου

Πρόλογος

Η ευρεία διάδοση των τεχνολογιών δικτύωσης δημιούργησε πρόσφορο έδαφος για υπηρεσίες που προσφέρουν χρήσιμες και άμεσα αξιοποιήσιμες πληροφορίες στον χρήστη. Η εξέλιξη των φορητών συσκευών σε ισχυρά υπολογιστικά μηχανήματα αναίρεσε την απαίτηση ύπαρξης ενός προσωπικού υπολογιστή και κατέστησε τις πληροφορίες αυτές διαθέσιμες παντού και πάντοτε. Μεγάλο ενδιαφέρον παρουσιάζουν οι υπηρεσίες γεωγραφικού εντοπισμού και πληροφόρησης, όπου ο χρήστης με ένα απλό τερματικό (λ.χ. κινητό τηλέφωνο, PDA, δέκτη GPS) έχει πρόσβαση σε πληροφορίες που τον ενδιαφέρουν. Επίσης η επεξεργασία του γεωγραφικού στίγματος των χρηστών δίνει την δυνατότητα για υπηρεσίες που μέχρι πρότινος θεωρούνταν αδύνατες. Οι προκλήσεις στις οποίες καλούνται να αντεπεξέλθουν τέτοιου είδους συστήματα είναι σημαντικές και απαιτούν αποδοτικούς αλγόριθμους που συνδυάζουν την ταχύτητα απόκρισης με την ακρίβεια στην απάντηση. Ιδιαίτερο ερευνητικό ενδιαφέρον έχει ο χειρισμός των *ερωτημάτων εγγύτερου γείτονα* και ειδικά η εφαρμογή *γεωμετρικών αλγορίθμων* για την διεκπεραίωσή τους.

Αντικείμενο της διπλωματικής εργασίας αποτελεί η μελέτη των ερωτημάτων εγγύτερων γειτόνων για ένα ρεύμα κινούμενων αντικειμένων, χρησιμοποιώντας *διαγράμματα Voronoi*. Τα διαγράμματα Voronoi αποτελούν έναν γεωμετρικό αλγόριθμο που δίνει άμεση και εύκολα αξιοποιήσιμη λύση στο πρόβλημα των εγγύτερων γειτόνων. Στα πλαίσια της εργασίας γίνεται μια θεωρητική επισκόπηση των συστημάτων διαχείρισης χωρικών δεδομένων και ρευμάτων δεδομένων καθώς και των ιδιοτήτων τους. Επίσης μελετώνται τα διαγράμματα Voronoi και τέλος εισάγεται ένας πρωτότυπος αλγόριθμος για την δημιουργία μεμονωμένων προσεγγιστικών κελιών Voronoi k-τάξης.

Διάρθρωση της εργασίας

Ο τόμος διαρθρώνεται σε πέντε κεφάλαια. Τα κεφάλαια 1 και 2 προσεγγίζουν θεωρητικά τα συστήματα διαχείρισης χωρικών δεδομένων και ρευμάτων δεδομένων. Το κεφάλαιο 3 παρουσιάζει τα διαγράμματα Voronoi και τις ιδιότητές τους. Στο κεφάλαιο 4 περιγράφεται ο πρωτότυπος αλγόριθμος υπολογισμού προσεγγιστικών διαγραμμάτων Voronoi. Τα αποτελέσματα της εργασίας συνοψίζονται στο κεφάλαιο 5.

Στο κεφάλαιο 1 περιγράφεται η έννοια των χωρικών δεδομένων από την σκοπιά των συστημάτων διαχείρισης. Τονίζεται η αδυναμία των συμβατικών συστημάτων βάσεων δεδομένων να αντεπεξέλθουν αποδοτικά σε χωρικά προβλήματα και ανα-

λύονται οι τροποποιήσεις και οι επεκτάσεις που οδήγησαν στα συστήματα χωρικών βάσεων δεδομένων, τονίζοντας την συμμετοχή των γεωμετρικών αλγορίθμων. Τέλος δίνεται μια συνοπτική περιγραφή των χωροχρονικών συστημάτων.

Στο κεφάλαιο 2 γίνεται εισαγωγή στα ρεύματα δεδομένων και στα ειδικά χαρακτηριστικά που παρουσιάζουν. Δίνονται οι απαιτήσεις λειτουργίας τους και γίνεται περιγραφή των προδιαγραφών των συστημάτων διαχείρισής τους. Ιδιαίτερη μνεία γίνεται στις τεχνικές που χρησιμοποιούνται για την αντιμετώπιση μεγάλου μεγέθους ρευμάτων και την επικέντρωση στο τμήμα των δεδομένων που ενδιαφέρει.

Το κεφάλαιο 3 αναλύει τα διαγράμματα Voronoi,τις ιδιότητές τους και τις επεκτάσεις τους. Παρουσιάζεται η χρησιμότητά τους στην απάντηση ερωτημάτων εγγύτερου γείτονα και οι αλγόριθμοι κατασκευής τους. Περιγράφεται ο αλγόριθμος του Fortune και παρουσιάζονται τα αποτελέσματα της πειραματικής αξιολόγησης σε ερωτήματα εγγύτερου γείτονα για ρεύματα κινούμενων αντικειμένων.

Στο κεφάλαιο 4 παρουσιάζεται ο πρωτότυπος αλγόριθμος για τον υπολογισμό προσεγγιστικών κελιών Voronoi k-τάξης για ένα ρεύμα δυναμικά κινούμενων αντικειμένων. Συγκεκριμένα παρουσιάζονται δύο παραλλαγές του, μία για σύγχρονη και μια για ασύγχρονη ανανέωση των θέσεων των κινούμενων αντικειμένων. Ο αλγόριθμος αξιολογείται πειραματικά και παρουσιάζονται τα αποτελέσματα.

Στο κεφάλαιο 5 εκτίθενται ορισμένα συμπεράσματα, καθώς και πιθανές μελλοντικές προοπτικές της εργασίας.

Ευχαριστίες

Η ολοκλήρωση της συγγραφής της παρούσας διπλωματικής εργασίας αποτέλεσε μια πολύμηνη και συχνά επίπονη διαδικασία, που όμως μου άφησε ένα αίσθημα ιδιαίτερης ικανοποίησης και περηφάνιας για το τελικό αποτέλεσμα. Στο τέλος αυτής της πορείας λοιπόν, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Κώστα Πατρούμπα τόσο για την αμέριστη συμπαράσταση και υποστήριξη που μου προσέφερε, όσο και για την προθυμία του να με καθοδηγήσει στα προβλήματα και στις δυσκολίες που συνάντησα. Επίσης, ιδιαίτερες ευχαριστίες οφείλω στον καθηγητή μου κ. Τίμο Σελλή για την υπεύθυνη επίβλεψη και το ιδιαίτερο ενδιαφέρον που έδειξε για αυτή την εργασία. Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου σε όλους όσους με στήριξαν σε αυτή την προσπάθεια.

Μηνόγιαννης Θεοφάνης
Αθήνα, Ιούνιος 2007

Περίληψη

Αντικείμενο της διπλωματικής εργασίας είναι η διερεύνηση της εφαρμογής των διαγραμμάτων *Voronoi* για την διαχείριση *χωρικών ρευμάτων δεδομένων* (spatial data streams) τα οποία προκύπτουν λ.χ. από τις καταγραφές αισθητήρων ή δεκτών GPS. Τα χωρικά ρεύματα δεδομένων δημιουργούνται από την καταγραφή των σημειακών θέσεων πολλών κινούμενων αντικειμένων τα οποία ανανεώνουν συχνά το στίγμα τους. Τα δεδομένα λοιπόν καταφθάνουν με μεγάλο και δυναμικά μεταβλητό ρυθμό σε πραγματικό χρόνο ,οπότε για την επεξεργασία τους απαιτείται η χρήση μόνο της κύριας μνήμης. Ενδεχόμενη αποθήκευσή τους θα επέφερε ανεπιθύμητες καθυστερήσεις, γι' αυτό και δεν ενδείκνυται η χρήση συμβατικών χωρικών βάσεων δεδομένων.

Ειδικότερα, μελετάται το ζήτημα της ταχείας επεξεργασίας *ερωτημάτων εγγύτερου γείτονα* (nearest neighbor) για στατικές εστίες, καθώς και ο online υπολογισμός μεμονωμένων κελιών *Voronoi* για δυναμικά κινούμενες εστίες.

Η μελέτη των διαγραμμάτων *Voronoi* στατικών αντικειμένων, αποτέλεσε χρήσιμο υπόβαθρο στην κατανόηση της εφαρμογής γεωμετρικών αλγορίθμων στο πεδίο των βάσεων δεδομένων. Ο αλγόριθμος του *Fortune* αξιοποιήθηκε για την κατασκευή του διαγράμματος *Voronoi* ενός συνόλου στατικών αντικειμένων και την απάντηση ερωτημάτων εγγύτερου γείτονα για ένα ρεύμα κινούμενων αντικειμένων επί αυτού.

Για την περίπτωση των μεμονωμένων κελιών προτείνεται πρωτότυπος αλγόριθμος, που υπολογίζει και ανανεώνει ένα *προσεγγιστικό κελί Voronoi k-τάξεως* χρησιμοποιώντας ένα ρεύμα κινούμενων σημείων ως είσοδο. Ο αλγόριθμος που προτείνεται επιλέχθηκε να είναι προσεγγιστικός και παρέχει μια κομψή και εύκολα υλοποιήσιμη λύση στο πρόβλημα αναζήτησης εγγύτερου γείτονα για τις περιπτώσεις ταυτόχρονης και ασύγχρονης ανανέωσης των θέσεων των αντικειμένων. Ο υπολογισμός των προσεγγιστικών κελιών τον καθιστά κατάλληλο για προβλήματα πραγματικού χρόνου, όπου η ακρίβεια στην απάντηση μπορεί να θυσιαστεί για χάρη της γρήγορης απόκρισης.

Λέξεις κλειδιά : ερωτήματα εγγύτερων γειτόνων, κινούμενα αντικείμενα, ρεύμα δεδομένων, διαμέριση επιπέδου, κελί Voronoi.

Abstract

The focus of this thesis is on studying the use of *Voronoi diagrams* for handling *spatial data streams* that derive from sources such as sensors or GPS receivers. Spatial data streams describe frequent positional updates for large sets of moving objects. Thus spatial data have a high and variable incoming rate and must be processed on line by using exclusively main memory techniques. Traditional spatial database management systems are unable to handle efficiently such data streams, as the use of secondary memory induces an unwanted delay overhead.

Specifically, we study the rapid process of *nearest neighbor queries* for static sites and the efficient on line calculation and maintenance of an *approximate order-k Voronoi Cell* around a certain point of interest when all objects continuously change their locations.

The study of *Voronoi diagrams* for static sites, provides a useful guide in understanding how geometric algorithms can be applied in databases for efficient handling of spatial data. *Fortune*'s algorithm was used to construct a Voronoi diagram for sets of static sites in the plane and to answer nearest neighbor queries over a stream of moving objects.

We also introduce an original processing technique for efficiently maintaining an approximate order-k Voronoi cell around a point of interest, when frequent positional update occur for all the other moving objects in the plane. This technique provides an elegant and easily implemented solution to the k-nearest neighbors problem both for concurrent and asynchronous positional updates, applicable in system where approximation is favored for the sake of an on line response.

Key Words : k-nearest neighbors query, moving objects, data stream, planar partition, Voronoi cell.

Περιεχόμενα

Κεφάλαιο 1

Διαχείριση χωρικών δεδομένων

Εισαγωγή

Η διείσδυση της πληροφορικής τα τελευταία χρόνια στους διάφορους τεχνικούς τομείς, συμβαδίζει με την όλο και μεγαλύτερη ανάγκη για υποστήριξή τους από αποδοτικές και συχνά πολύπλοκες εφαρμογές. Το πλαίσιο των απλών, μικρών και εξειδικευμένων προγραμματιστικών λύσεων είναι μάλλον παρωχημένο. Η εισαγωγή ολοκληρωμένων και αποτελεσματικών πακέτων λογισμικού οδήγησε στην έκρηξη που παρουσιάζεται στις μέρες μας. Ο τομέας των βάσεων δεδομένων αποτελεί ένα πολύ χειροπιαστό παράδειγμα της ανάπτυξης αυτής, μιας και όλο και περισσότερες εφαρμογές απαιτούν οργανωμένο και αποδοτικό τρόπο διαχείρισης των δεδομένων τους.

Η επέκταση όμως των εφαρμογών στα διάφορα επιστημονικά πεδία, εισήγαγε νέες ανάγκες και απαιτήσεις από τα συμβατικά συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ), τα οποία στην πρωταρχική τους μορφή αδυνατούσαν να καλύψουν. Πολλές εφαρμογές χειρίζονται γεωμετρικά, γεωγραφικά ή χωρικά δεδομένα. Το είδος των δεδομένων αλλά και ο ίδιος ο χώρος στον οποίο βρίσκονται είναι κάτι που ποικίλει ανάλογα με την εφαρμογή. Έτσι για παράδειγμα εφαρμογές κτηματολογίου χειρίζονται δισδιάστατες απεικονίσεις περιοχών της γης, ενώ ένα πρόγραμμα CAD δημιουργεί και επεξεργάζεται σχεδιαγράμματα VLSI. Ακόμα υπάρχουν εφαρμογές που ασχολούνται με τις χωροταξικές κατανομές των πρωτεϊνών. Τα ΣΔΒΔ παρουσιάζουν μεγάλα προβλήματα στην διαχείριση τέτοιων εξειδικευμένων δομών δεδομένων, με αποτέλεσμα την δημιουργία ανάγκης για ανάπτυξη νέων συστημάτων που μπορούν να ανταπεξέλθουν στον χειρισμό μεγάλου αριθμού γεωμετρικών σχημάτων.

Διάφορες τεχνικές προτάθηκαν με αποτέλεσμα να προκύψουν αρκετές λύσεις διαφορετικής προσέγγισης. Ανάλογα με το πρόβλημα που στόχευαν να λύσουν, τα προτεινόμενα συστήματα διαχείρισης ονομάστηκαν εικονικά (*pictorial · image*),

γεωμετρικά (*geometric*), γεωγραφικά (*geographic*) ή χωρικά (*spatial*). Αν και τα εικονικά συστήματα διαχείρισης έχουν κοινή αφετηρία προσέγγισης με τα χωρικά, εν τέλει αποτελούν δύο ξεχωριστές λύσεις. Στην συνέχεια θα αναλυθούν σε μεγαλύτερο βάθος τα συστήματα διαχείρισης χωρικών βάσεων δεδομένων.

1.1 Συστήματα διαχείρισης χωρικών βάσεων δεδομένων

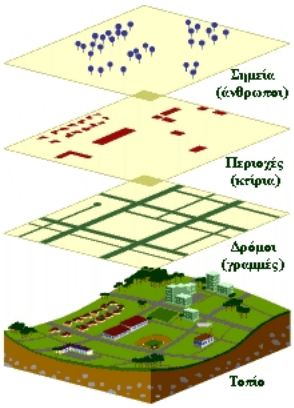
Ο ορισμός ενός *συστήματος διαχείρισης χωρικών βάσεων δεδομένων* (ΣΔΧΒΔ) κινείται γύρω από τρεις προϋποθέσεις:

- Είναι ένα σύστημα διαχείρισης βάσεων δεδομένων.
- Περιέχει καλά ορισμένους χωρικούς τύπους στο μοντέλο δεδομένων και στην γλώσσα ερωτημάτων του.
- Προσφέρει υποστήριξη στα χωρικά δεδομένα με μεθόδους για χωρική δεικτοδότηση και με εξειδικευμένους τελεστές πράξεων επί αυτών.

Η πρώτη απαίτηση δείχνει καθαρά ότι ένα τέτοιο σύστημα οφείλει εκτός από χωρικά δεδομένα, να μπορεί να χειριστεί εξίσου αποτελεσματικά και συμβατικά δεδομένα. Δεν είναι καθόλου σπάνιο άλλωστε το φαινόμενο μια εφαρμογή να απαιτεί την χρήση και των δύο τύπων. Αυτό το εύρος δυνατοτήτων άλλωστε είναι εκείνο που ξεχωρίζει αυτά τα συστήματα από τις εξειδικευμένες εφαρμογές που προϋπήρχαν.

Η δεύτερη απαίτηση αποτελεί το ουσιώδες συστατικό στα θεμέλια των χωρικών βάσεων. Οι χωρικοί τύποι δεδομένων αποτελούν τις βασικές μονάδες μοντελοποίησης του εκάστοτε προβλήματος. Ανάλογα με την φύση του, οι τύποι αυτοί διαφοροποιούνται, από γραμμές και πολύγωνα σε ένα γεωμετρικό περιβάλλον έως εξειδικευμένα σύμβολα σε CAD εφαρμογές για VLSI. Το σύστημα παρέχει και τους μηχανισμούς χειρισμού αυτών των τύπων στην γλώσσα ερωτημάτων του. Δηλαδή υποστηρίζει άμεση χρήση εκφράσεων που περιγράφουν αλληλεπίδραση (π.χ. αν δύο ευθείες τέμνονται), ιδιότητα (π.χ. αν ο όγκος ενός πολυέδρου είναι μικρότερος μιας τιμής) και αποτίμηση (π.χ. οι δύο μεγαλύτερες πλευρές ενός τριγώνου).

Η τρίτη απαίτηση έρχεται να δώσει την απαραίτητη αποδοτικότητα στα συστήματα χωρικών βάσεων δεδομένων. Οι μέθοδοι δεικτοδότησης όπως και στα συμβατικά συστήματα, αποτελούν κλειδί για να μπορεί να υπάρχει άμεση απόκριση στα ερωτήματα. Όμως η διαφορετική φύση των δεδομένων , απαιτεί ειδικό χειρισμό, οπότε είτε εισάγονται νέες δομές ευρετηρίων είτε επεκτείνονται οι ήδη υπάρχουσες. Για παράδειγμα, είναι εξαιρετικά χρήσιμο να μπορεί το σύστημα να εξάγει από ένα επιλεγμένο τμήμα του επιπέδου τα αντικείμενα που περικλείονται εντός αυτού, χρησιμοποιώντας γεωμετρικές ιδιότητες και όχι βασιζόμενο στην επεξεργασία των καρτεσιανών συντεταγμένων του συνόλου των αντικειμένων στον χώρο.



Σχήμα 1.1: Αποδόμηση τοπίου στους βασικούς χωρικούς τύπους δεδομένων

1.2 Χωρικοί τύποι δεδομένων

Όπως προαναφέρθηκε, η εισαγωγή ειδικών τύπων δεδομένων κρίνεται αναγκαία για τον ορθό ορισμό ενός συστήματος χωρικών βάσεων δεδομένων. Το εύρος όμως των εφαρμογών που καλείται να υποστηρίξει το μοντέλο των χωρικών βάσεων, δημιουργεί ένα αντιστοίχου ποικιλίας σύνολο τύπων για τα δεδομένα. Το μεγαλύτερο ενδιαφέρον έχει επικεντρωθεί στα Γεωγραφικά Πληροφοριακά Συστήματα (ΓΠΣ) (GIS) όπου βρίσκουν κύρια εφαρμογή τα συστήματα χωρικών βάσεων δεδομένων. Σε αυτό το πλαίσιο, τα δεδομένα έχουν γεωμετρικά χαρακτηριστικά και άρα κάποιες από τις βασικές και πρωτογενείς δομές της Ευκλείδειας γεωμετρίας χρησιμοποιούνται για να ορίσουν και τους τύπους.

Περιοριζόμενοι λοιπόν σε έναν δισδιάστατο κόσμο, με τις απαραίτητες εγγυήσεις όμως ότι μπορούμε να επεκταθούμε σε περισσότερες διαστάσεις αν αυτό κριθεί αναγκαίο, τα αντικείμενα που καλείται να μοντελοποιήσει το σύστημα μπορούν να ιδωθούν από δύο όψεις. Είτε αναφερόμαστε σε διακριτές οντότητες στο χώρο, όπου καθεμία απαιτεί ξεχωριστή περιγραφή, είτε πρέπει να αποδώσουμε μια περιγραφή στην ολότητα του χώρου, δηλαδή για κάθε σημείο του.

Στην πρώτη περίπτωση, οι πιο συνηθισμένοι τύποι δεδομένων είναι:

- Το *σημείο* (point) που αναπαριστά την γεωμετρική θέση ενός αντικειμένου, αδιαφορώντας για τις ιδιότητές του στο χώρο, όπως μια πόλη σε έναν χάρτη.

- Η *γραμμή* (line), που μπορεί να αποτελείται από ένα ή περισσότερα ευθύγραμμα τμήματα, δύναται να αναπαραστήσει συνδέσεις μεταξύ αντικειμένων, όπως για παράδειγμα δρόμοι και ποτάμια σε ένα χάρτη.

- Η *περιοχή* (region) που χρησιμεύει στην μοντελοποίηση αντικειμένων που εκτείνονται στο επίπεδο. Τέτοια είναι οι νομοί σε έναν πολιτικό χάρτη ή η κατάτμηση της γης σε μια εφαρμογή κτηματολογίου. Μια περιοχή μπορεί να περιέχει οπές ή και να αποτελείται από πολλά, μη επικαλυπτόμενα μέρη.

Όταν θέλουμε να αποδώσουμε μια περιγραφή σε όλο τον υπό μελέτη χώρο, οι δύο βασικές μονάδες είναι ο *τεμαχισμός* (partitions) και η *δικτύωση* (networks). Ο τεμαχισμός είναι μια μέθοδος που κατανέμει τον χώρο σε διακριτές και μη επικαλυπτόμενες υποπεριοχές, αναδεικνύοντας ιδιότητες αυτών όπως η γειτνίαση. Από την άλλη, με την δικτύωση μπορούμε να περιγράψουμε δίκτυα κίνησης ή επαφής ανάμεσα σε αντικείμενα στον χώρο. Για παράδειγμα φτιάχνοντας ένα δίκτυο από σημεία και γραμμές που τα ενώνουν μπορούμε να περιγράψουμε τις κύριες πόλεις μιας χώρας και το εθνικό οδικό δίκτυο που τις συνδέει.

Από τις παραπάνω περιγραφές, είναι φανερό ότι η φύση των τύπων δεδομένων ενός ΣΔΧΒΔ είναι τέτοια που εισάγει αυξημένη πολυπλοκότητα στον χειρισμό και την αποθήκευσή τους. Άρα το κλασικό σχεσιακό μοντέλο πρέπει να προσαρμοστεί στις νέες απαιτήσεις υπολογιστικής ισχύος και αποθηκευτικού χώρου. Μια σχεσιακή άλγεβρα που θα μπορεί να αναλάβει την διαδικασία αυτή δεν έχει γίνει ακόμα κοινώς αποδεκτή, αν και έχουν προταθεί αρκετές λύσεις.

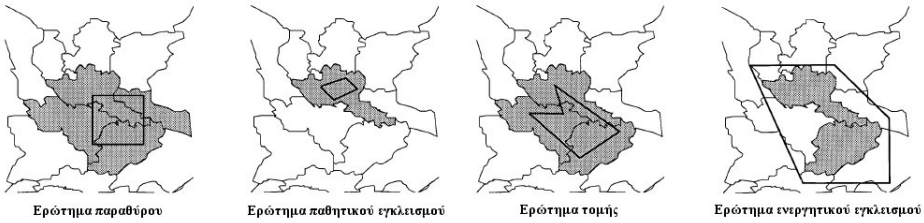
Ένα πρόβλημα που συναντάμε και στα κλασσικά ΣΔΒΔ, είναι αυτό της ακρίβειας στην αναπαράσταση των διαφόρων τύπων. Για έναν τύπο ακέραιου αριθμού για παράδειγμα, γνωρίζοντας από πριν την ακρίβεια που μπορεί να προσφέρει η εκάστοτε υλοποίηση, μπορούμε να κάνουμε τους απαραίτητους συμβιβασμούς κατά την υλοποίηση. Όμως στα χωρικά δεδομένα, η ακρίβεια παίζει έναν αρκετά πιο σημαντικό ρόλο και οι περιορισμοί που θέτει πρέπει να αντιμετωπιστούν καταλλήλως. Αν θεωρήσουμε ως βάση την ευκλείδεια γεωμετρία, τότε σε κάθε σημείο θα αντιστοιχεί ένα ζεύγος συντεταγμένων. Οι συντεταγμένες ως αριθμοί έχουν μια δεδομένη ακρίβεια, αυτήν που τους δίνει ο σχεδιαστής του συστήματος. Επειδή όμως η εσωτερική αναπαράσταση του χώρου στον υπολογιστή γίνεται με την αντιστοίχησή του σε ένα πλέγμα σημείων, οι πραγματικές συντεταγμένες προσεγγίζονται με την πλησιέστερη δυνατή θέση του πλέγματος αυτού.

Η τεχνική αυτή ενέχει επιπλοκές, όταν για παράδειγμα βρεθεί και αποθηκευτεί το σημείο τομής δύο ευθειών που αναπαριστώνται με τις εξισώσεις τους. Το αποτέλεσμα αποθηκεύτηκε με τις πλησιέστερες συντεταγμένες του πλέγματος στις πραγματικές συντεταγμένες τομής. Σε μετέπειτα έλεγχο είναι δυνατό αυτό το σημείο τομής να μην βρίσκεται πάνω σε μία από τις δύο ευθείες λόγω του αριθμητικού λάθους που υπεισήλθε κατά την αποθήκευσή του. Η αντιμετώπιση του προβλήματος αυτού, είναι προτιμότερο να γίνει στο επίπεδο των τύπων δεδομένων και όχι στο επίπεδο χειρισμού τους από την σχεσιακή άλγεβρα. Έχουν προταθεί διάφορες λύσεις, οι οποίες παρακάμπτουν το πρόβλημα, ανάγοντας και τον χώρο περιγραφής στο πλέγμα που χρησιμοποιεί ο υπολογιστής εσωτερικά, ώστε να απαλειφθούν τα αριθμητικά λάθη. Για να το επιτύχουν αυτό χρησιμοποιούν περιγραφικές έννοιες και αντικείμενα για να μοντελοποιήσουν τους βασικούς τύπους που προαναφέρθηκαν.

1.3 Χωρικοί τελεστές

Οι πιο δημοφιλείς τελεστές μεταξύ χωρικών οντοτήτων είναι :

- **Ισότητας** (*exact match query*) : Δοθέντος ενός αντικειμένου με μια γεωμετρία να βρεθούν όλα τα αντικείμενα με την ίδια γεωμετρία.



Σχήμα 1.2: Παραδείγματα χωρικών τελεστών

- **Σημείου** (*point query*) : Δοθέντος ενός σημείου του k-διάστατου χώρου, να βρεθούν όλα τα αντικείμενα που το περιέχουν.
- **Παραθύρου** (*window query*) : Δοθέντος ενός k-διάστατου παραθύρου, να βρεθούν όλα τα αντικείμενα που έχουν ένα τουλάχιστον κοινό σημείο με το παράθυρο (σχήμα 1.2).
- **Τομής** (*intersection query*) : Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα με τα οποία έχει κοινά εσωτερικά σημεία (σχήμα 1.2).
- **Παθητικού εγκλεισμού** (*enclosure query*) : Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα που το περιέχουν (σχήμα 1.2).
- **Ενεργητικού εγκλεισμού** (*containment query*) : Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα που αυτό περιέχει (σχήμα 1.2).
- **Γειτνίασης** (*adjacent query*) : Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα με τα οποία δεν έχει κανένα κοινό εσωτερικό σημείο αλλά το περίγραμμά τους έχει κοινά σημεία.
- **Εύρεσης k- εγγύτερων γειτόνων** (k-*nearest neighbor*) : Δοθέντος ενός συνόλου ακίνητων αντικειμένων Q και ενός τυχαίου σημείου P (κινούμενου ή μη) να προσδιορισθούν τα k κοντινότερα στο P σημεία του συνόλου Q.
- **Χωρικής σύνδεσης** (*spatial join*) : Δοθέντος ενός χωρικού γνωρίσματος, πραγματοποιεί τον συνδυασμό δύο ή περισσοτέρων συνόλων δεδομένων που το περιέχουν.

Μια σημαντική κλάση χωρικών τελεστών είναι οι τελεστές επιλογής χωρικών δεδομένων γιατί η ανεύρεση και η ενημέρωση τέτοιων δεδομένων εξαρτάται και από την θέση τους στο χώρο. Οι μέθοδοι που χρησιμοποιούν αυτοί οι τελεστές διακρίνονται σε δύο μεγάλες κατηγορίες:

- Στις *μεθόδους προσπέλασης σημείων* (point access methods) που βρίσκουν εφαρμογές σε βάσεις που αποθηκεύουν σημεία. Τα σημεία μπορεί να είναι σε οποιοδήποτε αριθμό διαστάσεων, τα δεδομένα της βάσης όμως, δεν περιγράφουν άλλες χωρικές ιδιότητες (λ.χ. σχήμα).

- Στις *μεθόδους προσπέλασης περιοχών* (spatial access methods) που χειρίζονται γεωμετρικά αντικείμενα ενδογενώς, όπως γραμμές, πολύγωνα και πολύεδρα.

Η ανάγκη για εισαγωγή νέων ειδικών μεθόδων προσπέλασης προκύπτει από το πρόβλημα της απουσίας ολικής διάταξης στον πολυδιάστατο χώρο. Αυτό περιγράφει την αδυναμία απόλυτης αντιστοίχησης του δισδιάστατου χώρου (ή του κ-διάστατου γενικά) στο μονοδιάστατο χώρο. Έτσι δεν είναι εύκολο να συμπεράνουμε ότι αν δυο αντικείμενα βρίσκονται κοντά στον δισδιάστατο χώρο,τότε θα βρίσκονται κοντά και στον αντίστοιχο μονοδιάστατο.

1.4 Χωρικές μέθοδοι δεικτοδότησης

Η επιλογή ενός μέρους από το σύνολο των χωρικών δεδομένων μιας βάσης, αποτελεί ένα πρόβλημα με μεγάλες προκλήσεις και απαιτήσεις. Εκτός από την έλλειψη τυποποιημένης άλγεβρας και γλώσσας ερωτημάτων και την απουσία ολικής διάταξης, η επέκταση των ήδη υπαρχουσών παραδοσιακών μεθόδων προσπέλασης όπως το B-tree δεν είναι μια απλή διαδικασία. Η αρχική προσέγγιση της εφαρμογής μιας τέτοιας δεικτοδότησης σε κάθε διάσταση ενδιαφέροντος, αν και αποτελεί μια πρόταση απλή στην υλοποίηση δεν επιφέρει ικανοποιητικά αποτελέσματα. Νέες μέθοδοι δεικτοδότησης προτάθηκαν και πολλές και ικανοποιητικές λύσεις βγήκαν από τα αποτελέσματα της έρευνας, αν και δεν υπάρχει ακόμα μια καθολικά εφαρμόσιμη και αποδεκτή λύση, λόγω της πολυπλοκότητας του προβλήματος.

Μία *μέθοδος πολυδιάστατης προσέγγισης* (πολυδιάστατο ευρετήριο) πρέπει να ικανοποιεί ορισμένες προϋποθέσεις για να αποδειχτεί αποδοτική σε επίπεδο εφαρμογών. Πολλές από αυτές προέρχονται από τον χώρο των κλασσικών ΣΔΒΔ, ενώ άλλες εισήχθησαν για να αντιμετωπιστούν οι ιδιαιτερότητες των χωρικών δεδομένων. Οι *Gaede* και *Gunther* [GG98] προτείνουν τις ακόλουθες, βασιζόμενοι στις ιδιαιτερότητες των χωρικών δεδομένων και στις απαιτήσεις σε επίπεδο εφαρμογής.

- **Δυναμική :** Το ευρετήριο πρέπει να είναι ικανό να παρακολουθεί απρόσκοπτα την κατάσταση των δεδομένων, ανεξάρτητα από την συχνότητα εισαγωγής, διαγραφής και ανανέωσής τους.
- **Ενδελεχής υποστήριξη και αξιοποίηση δευτερεύουσας και τριτεύουσας μνήμης :** Το μέγεθος των χωρικών δεδομένων στην πλειοψηφία των περιπτώσεων κλιμακώνεται ταχύτατα. Η κύρια μνήμη δεν επαρκεί για την αποθήκευσή τους, με αποτέλεσμα η χρήση κι άλλων επιπέδων μνήμης να καθίσταται αναγκαία.
- **Γενικότητα :** Ένα καλά ορισμένο ευρετήριο δεν δίνει εμφανή προτεραιότητα σε κάποιες συγκεκριμένες λειτουργίες, αλλά φροντίζει την επαρκή υποστήριξη όλων όσων απαιτεί η εφαρμογή.
- **Ομοιογένεια συμπεριφοράς :** Η σειρά και ο ρυθμός εισαγωγής των δεδομένων στο ευρετήριο δεν θα πρέπει να έχει κανέναν αντίκτυπο στις επιδόσεις του.

- **Απλότητα :** Προτιμώνται απλοί και μη εξειδικευμένοι αλγόριθμοι κατά την δημιουργία του ευρετηρίου, ώστε να ελαχιστοποιηθούν τόσο τα λάθη και το κόστος συντήρησης , όσο και η προσαρμοστικότητά του σε μεγάλου μεγέθους εφαρμογές.

- **Κλιμάκωση :** Είναι θεμιτή η καλή απόκριση του ευρετηρίου στην κλιμακωτή αύξηση του μεγέθους των δεδομένων.

- **Μικρή χρονική απόκριση :** Ένα ευρετήριο εκτός από σωστά πρέπει να αποκρίνεται και γρήγορα. Ιδανικά θα πρέπει να έχει το πολύ λογαριθμικό χρόνο απόκρισης, κάτι το οποίο οφείλει να ισχύει ανεξάρτητα από το είδος και το μέγεθος των δεδομένων.

- **Μικρές απαιτήσεις αποθήκευσης :** Ο αποθηκευτικός χώρος που καταλαμβάνει πρέπει να είναι μικρός και σαφέστατα μικρότερος από αυτόν του συνόλου των δεδομένων.

- **Υποστήριξη πολλαπλών χρηστών :** Εφόσον όλα τα σύγχρονα συστήματα διαχείρισης υποστηρίζουν ταυτόχρονη πρόσβαση από πολλούς χρήστες, η λειτουργία αυτή πρέπει να υποστηρίζεται και στο ευρετήριο για αποφυγή τόσο πιθανής καταστροφής δεδομένων όσο και για λόγους απόδοσης.

- **Ελάχιστη επιβάρυνση στο σύστημα :** Η εισαγωγή του ευρετηρίου στο σύστημα δεν θα πρέπει να επιφέρει σημαντικές επιπτώσεις στην απόδοση των υπολοίπων τμημάτων του.

1.4.1 Μέθοδοι προσπέλασης σημείων

Εδώ θα αναφερθούμε ενδεικτικά τόσο σε μεθόδους που στηρίζονται στον κατακερματισμό όσο και σε ιεραρχικές μεθόδους (βασισμένες σε δομές δένδρων). Στόχος τους να οργανώσουν τα σημειακά αντικείμενα σε ομάδες, βάσει της γειτνίασής τους στο επίπεδο.

- Το *αρχείο πλέγματος* (Grid file) υπερθέτει ένα k-διάστατο ορθογώνιο πλέγμα στον k-διάστατο χώρο. Τα κελιά του πλέγματος μπορούν να έχουν διαφορετικό μέγεθος και σχήμα.

- Το *αρχείο ισοσταθμισμένου και φωλιασμένου πλέγματος* (balanced and nested grid file -BANG file) χρησιμοποιεί δενδρικές δομές αναζήτησης πάνω στο k-διάστατο ορθογώνιο πλέγμα (grid). Η διαφορά με το grid file είναι ότι οι υποχώροι του πλέγματος μπορεί να τέμνονται.

- Το *δένδρο φίλος* (The buddy tree) στηρίζεται σε δυναμικό κατακερματισμό και ο κατάλογός του είναι δομημένος ως δένδρο.

- Το *K-D-B-δένδρο* είναι ένα δυαδικό δέντρο το οποίο παριστάνει την διαδοχική υποδιαίρεση του D-διάστατου χώρου σε υποχώρους μέσω υπερεπιπέδων διάστασης $D-1$.

- Το *hB-δένδρο* εφαρμόζει τις ιδέες επικαλυπτόμενης διαμέρισης του χώρου του BANG file με την δενδρική δομή των buddy tree και K-D-B tree, προσθέτοντας όμως μεγαλύτερη ευελιξία στην διάσπαση των κόμβων, για να αποφευχθεί η κλιμάκωση των διαχωρισμών τους.

- Το *δένδρο τοπικής απόφασης διάσπασης* (local split decision tree - LSD tree) είναι ένα ισοσταθμισμένο δένδρο, το οποίο χωρίζει τον χώρο σε κελιά διαφόρων διαστάσεων και τα αποθηκεύει σε κατάλογο επίσης δενδρικής δομής.

1.4.2 Μέθοδοι προσπέλασης περιοχών

Συχνά αναφέρονται στην βιβλιογραφία και ως χωρικές μέθοδοι προσπέλασης ή και χωρικά ευρετήρια. Ο χειρισμός μη-σημειακών χωρικών δεδομένων με έκταση απαιτεί ιδιαίτερες τεχνικές λόγω της ιδιαίτερης φύσης των δεδομένων. Ενδεικτικά αναφέρονται οι εξής:

- Η τεχνική *μετασχηματισμού*, σύμφωνα με την οποία ένα αντικείμενο με έκταση στο δισδιάστατο χώρο (ή k-διάστατο γενικά) επιχειρείται να καθοριστεί πλήρως με στοιχεία του μονοδιάστατου χώρου (λ.χ. space-filling curves).

- Στην τεχνική *αλληλοεπικαλυπτόμενων περιοχών* αφήνουμε διαφορετικούς κάδους δεδομένων να αντιστοιχούν σε αλληλεπικαλυπτόμενες περιοχές (overlapping regions) του χώρου.

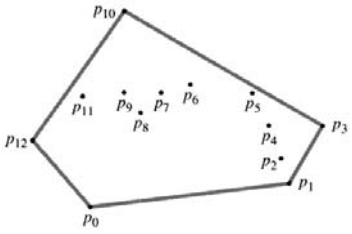
- Η τεχνική *αποκοπής* σύμφωνα με την οποία δεν υπάρχει ποτέ αλληλοεπικάλυψη. Αν ένα αντικείμενο εκτείνεται σε περισσότερες από μια περιοχές, είτε το σπάει και αποθηκεύεται κάθε κομμάτι σε διαφορετικό κάδο, είτε αποθηκεύεται ολόκληρο σε κάθε έναν από τους κάδους.

Το *R-δένδρο* είναι ένα ισοσταθμισμένο δένδρο με αναζήτηση παρόμοια με εκείνη των B-δένδρων και αποτελεί μια δομή φωλιασμένων ορθογώνιων περιοχών του χώρου αναζήτησης ομαδοποιώντας τις περιοχές με χρήση ελάχιστων περιβαλλόντων ορθογωνίων (minimum bounding rectangles) .Για το R-δένδρο έχουν προταθεί βελτιώσεις όπως το *R+-δένδρο* το οποίο χρησιμοποιεί την τεχνική της αποκοπής αλλά και το *R*-δένδρο* στο οποίο ισχύει η λεγόμενη επιβεβλημένη επανεισαγωγή των στοιχείων.

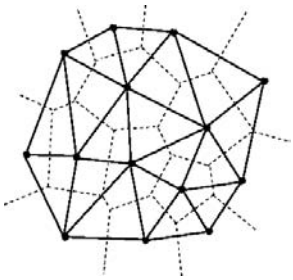
Ενδεικτικά αναφέρουμε ακόμα το *κυτταρικό δένδρο* (Cell tree) όπου χρησιμοποιείται η τεχνική της αποκοπής, για να διευκολυνθεί ο χειρισμός αντικειμένων με αυθαίρετο σχήμα.

1.4.3 Γεωμετρικοί αλγόριθμοι

Στην περίπτωση των σημειακών δεδομένων μπορούμε χρησιμοποιώντας γεωμετρικούς αλγόριθμους να εξάγουμε πληροφορίες για βασικές ιδιότητες των σημείων, κατασκευάζοντας χαρακτηριστικά σχήματα που αναπαριστούν τη μορφή των στοιχείων. Ενδεικτικά τέτοιοι αλγόριθμοι είναι :



Σχήμα 1.3: Κυρτό περίβλημα σημείων



Σχήμα 1.4: Τριγωνισμός Delaunay πάνω σε διάγραμμα Voronoi

Κυρτό περίβλημα

Ορισμός. *Κυρτό περίβλημα* ενός πεπερασμένου συνόλου P από n σημεία στο επίπεδο είναι η κοινή τομή των κυρτών πολυγώνων που καθένα περικλείει όλα τα σημεία του P.

Το περίβλημα είναι το μικρότερο σε επιφάνεια κυρτό σύνολο σημείων στο επίπεδο, που περιλαμβάνει όλα τα σημεία του P (σχήμα 1.3). Το κυρτό περίβλημα αποδεικνύεται ότι είναι το μοναδικό κυρτό πολύγωνο, που περικλείει όλα τα σημεία του P έχοντας ως κορυφές σημεία του P. Μπορεί να δοθεί ένας ισοδύναμος ορισμός με τη μορφή κυρτού γραμμικού συνδυασμού

Ορισμός. Ως *γραμμικός συνδυασμός* n σημείων $p_1, p_2, \ldots, p_n$ στο επίπεδο ορίζεται το άθροισμα $\sum_{i=1}^{n} \lambda_i p_i$. Αν $\forall i, \lambda_i \geq 0$ και $\sum_{i=1}^{n} \lambda_i = 1$, τότε ο γραμμικός συνδυασμός ονομάζεται *κυρτός*.

Το κυρτό περίβλημα ορίζεται τόσο στις δύο όσο και στις τρεις διαστάσεις. Τόσο στο επίπεδο με τον *σταδιακό αλγόριθμο του Andrews*, όσο και στις τρεις διαστάσεις με τον *τυχαιοποιημένο αλγόριθμο*, επιτυγχάνουμε την κατασκευή του περιβλήματος σε χρόνο $O(n \log n)$, για ένα σύνολο n σημείων.

Διαγράμματα Voronoi

Για ένα σύνολο n σημείων (*εστίες*) P στο επίπεδο, το *διάγραμμα Voronoi* αποτελεί μια διαμέριση που δίνει πληροφορία για την εγγύτητα κάθε σημείου του επιπέδου σε κάποια από τις n εστίες. Συγκεκριμένα, για κάθε εστία δημιουργείται ένα κελί που περικλείει όλα τα σημεία του επιπέδου που βρίσκονται εγγύτερα σε αυτήν απ΄ ό,τι σε οποιαδήποτε άλλη εστία του συνόλου P. Τα διαγράμματα *Voronoi* επεκτείνονται και για μεγαλύτερο αριθμό διαστάσεων, καθώς και για διαφορετική ιδιότητα κατασκευής (λ.χ. κελιά δύο εγγύτερων γειτόνων). Μια πιο διεξοδική μελέτη τους παρατίθεται στο τρίτο κεφάλαιο.

Τριγωνισμός Delaunay

Ο *τριγωνισμός Delaunay* αποτελεί το δυικό πρόβλημα του διαγράμματος *Voronoi* και μελετήθηκε για πρώτη φορά το 1908 από τον Voronoi και επεκτάθηκε αργότερα από τον Delaunay. Έχοντας $P = \{p_1, p_2, ..., p_n\}$ ένα σύνολο n διακριτών σημείων (εστίες) στο επίπεδο, ο τριγωνισμός *Delaunay* προκύπτει ως:

Ορισμός. Το σύνολο των πλευρών όλων των τριγώνων που δημιουργούνται από τις εστίες του P, που ο περιγεγραμμένος κύκλος τους δεν περιέχει άλλες εστίες.

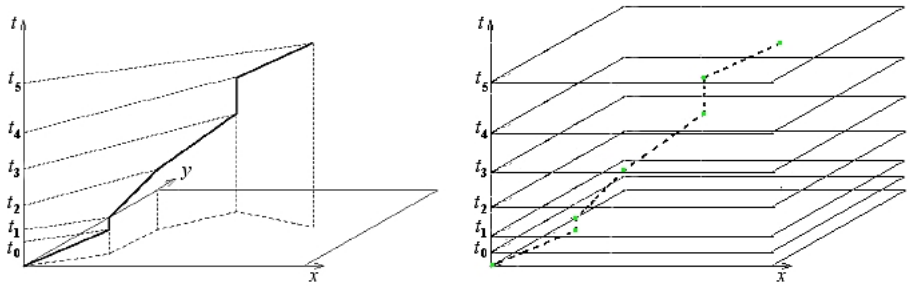
Η κατασκευή του τριγωνισμού είναι πολύ εύκολη αν για το σύνολο των εστιών, υπάρχει διαθέσιμο το διάγραμμα *Voronoi* (σχήμα 1.4). Στην περίπτωση αυτή ενώνουμε με ευθύγραμμα τμήματα τις εστίες που τα κελιά τους έχουν μια κοινή πλευρά και προκύπτει ο τριγωνισμός *Delaunay*. Αντίστοιχα εύκολη είναι η κατασκευή του διαγράμματος *Voronoi*, αν έχουμε τον τριγωνισμό λόγω του δυισμού των προβλημάτων.

1.5 Αποτίμηση χωρικών ερωτημάτων

Όταν διαχειριζόμαστε χωρικά δεδομένα είναι σύνηθες η επεξεργασία τους να περνά από δύο διαδοχικά στάδια. Η ανάγκη αυτή εισάγεται από την πολυπλοκότητα που έχουν τα χωρικά δεδομένα από την φύση τους. Έτσι ένα ερώτημα σε μια χωρική βάση περνά από την φάση *filter* και μετά από μια φάση *refinement*.

Αναλυτικά, στην φάση *filter* ο k-διάστατος χώρος όπου βρίσκονται τα δεδομένα φιλτράρεται με κατάλληλο τρόπο και επιστρέφεται ένα κομμάτι αυτού όπου βρίσκονται τα δεδομένα. Σαφώς η περιοχή αυτή είναι πολύ μικρότερη από τον αρχικό χώρο και αποτελεί ένα υπερσύνολο της λύσης, αλλά δεν εμπεριέχει ακρίβεια στον προσδιορισμό των δεδομένων που αναζητούμε. Αυτός ο παράγοντας ακρίβειας είναι που εισάγει η διαδικασία του *refinement*. Στον χώρο που επέστρεψε το *filter* διεξάγεται διεξοδική αναζήτηση στις περιοχές που περιλαμβάνουν τα στοιχεία της απάντησης.Μπορούμε να περιγράψουμε την διαδικασία ως την εξαγωγή ενός "θολού" υποσυνόλου του αρχικού χώρου από το *filter* στην οποία περιέχονται σίγουρα τα ζητούμενα δεδομένα μαζί με άλλα τα οποία δεν αποτελούν μέρος της απάντησης. Η εκλέπτυνση αυτού του συνόλου έτσι ώστε να περιλαμβάνει μόνο τα ζητούμενα γίνεται από την διαδικασία *refinement* που επιστρέφει και την τελική απάντηση.

Η υλοποίηση αυτής της τεχνικής προϋποθέτει τον χωρισμό του αρχικού χώρου δεδομένων με κατάλληλες δομές χειρισμού χωρικών δεδομένων. Συγκεκριμένα για την διαδικασία του *filter* χρησιμοποιείται οποιαδήποτε από τις χωρικές μεθόδους δεικτοδότησης που περιγράφηκαν παραπάνω. Για την αξιοποίηση των αποτελεσμάτων των μεθόδων αυτών μπορεί να χρησιμοποιηθεί κάποιος γεωμετρικός αλγόριθμος διαχείρισης χωρικών δεδομένων όπως είναι τα διαγράμματα *Voronoi*. Έτσι στο υποσύνολο του χώρου που επιστρέφει για παράδειγμα μια δομή όπως το



Σχήμα 1.5: (α) Αναπαράσταση τροχιάς κινούμενου αντικειμένου. (β) Στιγμιότυπα τροχιάς με προβολές στο επίπεδο του χρόνου.

grid file περιέχεται ένας αριθμός πιθανών λύσεων που ανήκουν σε κάποιο από τα κελιά του διαγράμματος *Voronoi*. Το μόνο που απομένει είναι να γίνει αναζήτηση σε αυτά τα κελιά χρησιμοποιώντας κάποια μέθοδο, όπως είναι η *point in polygon* ώστε να βρεθούν οι ακριβείς απαντήσεις.

1.6 Διαχείριση χωροχρονικών φαινομένων

Τον τελευταίο καιρό παρατηρείται μεγάλη ζήτηση για εφαρμογές που χειρίζονται δυναμικά χωρικές θέσεις αντικειμένων. Ένα πολύ χαρακτηριστικό παράδειγμα είναι τα μηχανήματα αυτόματης πλοήγησης αυτοκινήτων με χρήση *συστημάτων γεωγραφικού εντοπισμού* (GPS). Τα φαινόμενα που μοντελοποιούνται από αυτές τις εφαρμογές έχουν χωροχρονική φύση, αφού εκτός από την θέση ενός αντικειμένου (λ.χ. ενός αυτοκινήτου) μας απασχολεί και η εξέλιξη της κίνησής του στον χρόνο. Τα συστήματα διαχείρισης που χρησιμοποιούνται στις περιπτώσεις αυτές είναι συστήματα διαχείρισης *κινούμενων αντικειμένων* (moving objects) και αποτελούν γενικεύσεις και επεκτάσεις των χωρικών συστημάτων. Η προσθήκη του χρόνου ως παραμέτρου στο σύστημα, επιφέρει αλλαγές σε πολλά επίπεδα, αφού απαιτείται από τους τελεστές να μπορούν να χειρίζονται αποδοτικά και ενδογενώς τα χωροχρονικά φαινόμενα.

Ο ορισμός των κατάλληλων τύπων για την περιγραφή των κινούμενων αντικειμένων περνάει μέσα από την παραδοχή ότι δεν είναι δυνατή η συνεχής παρακολούθηση της κίνησής τους. Σε θεωρητικό επίπεδο, οι τύποι ορίζονται αναιρώντας την παραδοχή αυτή, για λόγους ορθότητας και πληρότητας. Σε επίπεδο υλοποίησης όμως, ο περιορισμός αυτός δεν μπορεί να αγνοηθεί αφού αυξάνει την πολυπλοκότητα σχεδίασης. Το σύνολο των τύπων των χωρικών συστημάτων επεκτείνεται σε αυτή την περίπτωση με ειδικούς τύπους που προσδίδουν την διάσταση του χρόνου στους υπάρχοντες. Για παράδειγμα ένα σημείο στο επίπεδο περιγράφεται από την πλειάδα $< x, y, t >$, δηλαδή τις συντεταγμένες του και το χρονόσημο κατά το οποίο βρίσκεται σε αυτές. Ανάλογα επεκτείνονται και οι τελεστές ώστε να μπορούν να επιτελεστούν ερωτήματα που επηρεάζονται από την χρονική εξέλιξη.

Η αδυναμία συνεχούς καταγραφής της κίνησης των αντικειμένων, οδηγεί σε λύσεις προσέγγισης στην εσωτερική τους αναπαράσταση. Για παράδειγμα, έχοντας διαθέσιμες τις διαδοχικές θέσεις ενός αντικειμένου στο επίπεδο για ένα πλήθος χρονοσήμων, χρησιμοποιείται κάποιου είδους παρεμβολή για την παραγωγή θέσεων σε ενδιάμεσες στιγμές. Η γραμμική παρεμβολή είναι η συνηθέστερη μιας και καλύπτει ικανοποιητικά την πλειοψηφία των περιπτώσεων. Υπάρχουν όμως και εφαρμογές που απαιτούν μεγαλύτερη ακρίβεια στον προσδιορισμό της κίνησης, με αποτέλεσμα να χρησιμοποιούνται παρεμβολές ανώτερης τάξης. Το αποτέλεσμα της προσέγγισης αποκαλείται *τροχιά αντικειμένων* (trajectory) και δίνει την μεταβολή της θέσης του αντικειμένου σε συνάρτηση με τον χρόνο. Αν η τροχιά αναπαρασταθεί σε ένα σύστημα αξόνων για τις συντεταγμένες του και τον χρόνο, δίνεται πλήρης πληροφορία για την εξέλιξη της κίνησής του (σχήμα 1.5(α)), αλλά και δυνατότητα απόκτησης στιγμιοτύπων αυτής με προβολές στο επίπεδο του χρόνου (σχήμα 1.5(β)).

Ένα εγγενές πρόβλημα στα χωροχρονικά συστήματα αποτελεί η αβεβαιότητα (uncertainty) που επισέρχεται στην αναπαράσταση. Αυτή οφείλεται τόσο στην παρεμβολή που χρησιμοποιείται για να δημιουργηθούν οι τροχιές όσο και στους περιορισμούς των τεχνολογικών υποδομών που υποστηρίζουν το σύστημα. Για παράδειγμα οι συσκευές GPS διαθέτουν ικανοποιητική μεν ακρίβεια στον προσδιορισμό της θέσης, δεν μπορούν όμως να την ενημερώνουν συνεχώς λόγω περιορισμών στο εύρος ζώνης των καναλιών επικοινωνίας (*bandwidth*) και υψηλού κόστους. Σε περίπτωση που η πηγή του σήματος είναι κάποιος αισθητήρας, υπεισέρχεται και σφάλμα στις συντεταγμένες του σήματος. Έτσι αθροιστικά το σφάλμα θεωρείται αναπόφευκτο και οι προσπάθειες επικεντρώνονται στον μεγαλύτερο δυνατό περιορισμό του. Μια άλλη πηγή αβεβαιότητας είναι η ύπαρξη διακοπών στην αποστολή δεδομένων από τις πηγές. Αυτό μπορεί να συμβαίνει όταν για παράδειγμα ένας αισθητήρας τεθεί εκτός λειτουργίας ή όταν η συσκευή GPS δεν μπορεί να επικοινωνήσει με ικανό αριθμό δορυφόρων ώστε να παράγει στίγμα. Ο χειρισμός τέτοιων ανωμαλιών γίνεται με πρόβλεψη της τροχιάς που θα ακολουθούσε το αντικείμενο, σύμφωνα με παραμέτρους όπως η ταχύτητα, η κατεύθυνση και η επιτάχυνσή του, με χρήση *συναρτήσεων αναπαράστασης* ή *προσχεδίων της κίνησης*. Βεβαίως το σφάλμα εδώ είναι πιθανώς μεγάλο, γι' αυτό και επιδιώκεται η όσο πιο γρήγορη αποκατάσταση της επικοινωνίας.

Εκτός από τα τυπικά ερωτήματα θέσης, όπως λ.χ. εγγύτερου γείτονα, υπάρχουν και τα *ερωτήματα τροχιάς αντικειμένων*. Αυτά διακρίνονται σε:

- **Τοπολογικά** (*topological*) που αναφέρονται στη συνολική πορεία των αντικειμένων, όπως η έναρξη ή η παύση καταγραφής τους, η χωροχρονική σύνδεση και η διασταύρωση της κίνησης αντικειμένων.

- **Πλοήγησης** (*navigational*) που παράγουν δευτερογενείς πληροφορίες όπως η διανυθείσα απόσταση, η ταχύτητα,ο χρόνος διαδρομής, ο προσανατολισμός και η περιοχή κάλυψης. Τα ερωτήματα αυτά παρέχουν χρήσιμες πληροφορίες και χρησιμοποιούνται κατά κόρον σε εφαρμογές πλοήγησης.

Η πολυπλοκότητα που εισάγεται στον χειρισμό των ερωτημάτων αυτών καθιστά επιτακτική την εισαγωγή δομών *δεικτοδότησης* (indexing) για την αποδοτική διεκπεραίωσή τους. Οι λύσεις που έχουν προταθεί είναι αρκετές και γενικά τείνουν να ανταποκρίνονται στην ιδιαίτερη φύση των δεδομένων, στην αναγκαιότητα απάντησης σε *πραγματικό χρόνο* (real-time response) αλλά και σε ενδεχόμενους περιορισμούς στην κίνηση των αντικειμένων που προκύπτουν από το περιβάλλον κίνησης. Πολύ συχνά χρησιμοποιήθηκε ως βάση το R-tree και δημιουργήθηκαν παραλλαγές όπως τα *STR-tree*, *TB-tree* και *STAR-tree*. Άλλες λύσεις αποτελούν ο *δυϊκός μετασχηματισμός* της αναπαράστασης της κίνησης και οι *κινητικές δομές δεδομένων*, ιδέες απο το συγγενές πεδίο της Υπολογιστικής Γεωμετρίας.

Κεφάλαιο 2

Ρεύματα δεδομένων

Εισαγωγή

Τα τελευταία χρόνια παρατηρείται έξαρση στον τομέα των εφαρμογών που δεν χειρίζονται στατικά δεδομένα, αλλά αντιθέτως δέχονται ως είσοδο, επεξεργάζονται και πιθανώς έχουν ως έξοδο *ρεύματα δεδομένων* (data streams). Αντιθέτως μια παραδοσιακή βάση είναι σε θέση να απαντήσει ερωτήματα του χρήστη χρησιμοποιώντας την τρέχουσα κατάστασή της. Η προϋπόθεση για την λειτουργία του παραπάνω σχήματος είναι τα δεδομένα να αποθηκεύονται σε μόνιμα μέσα και να είναι πάντοτε προσπελάσιμα. Οι χρήστες είναι σε θέση να υποβάλουν διαφόρων τύπων ερωτήματα (queries) όπως και να εκτελούν δοσοληψίες (λ.χ. εισαγωγή, διαγραφή ή τροποποίηση δεδομένων). Το παραδοσιακό μοντέλο εξασφαλίζει ακριβείς απαντήσεις στα ερωτήματα των χρηστών, με την προϋπόθεση ότι τα δεδομένα είναι άμεσα διαθέσιμα κατά την διάρκεια των υπολογισμών. Βεβαίως το μέγεθος των υπολογισμών μπορεί να είναι τέτοιο, που η απάντηση να χρειαστεί αρκετό χρόνο για να παραχθεί.

Αντίθετα, στο μοντέλο του ρεύματος δεδομένων η διαχείριση των στοιχείω χρειάζεται να διαφοροποιηθεί αρκετά. Το παραδοσιακό μοντέλο δεδομένων και σύνολο πράξεων πρέπει να προσαρμοστούν ώστε να χειρίζονται όχι στατικά δεδομένα, αλλά ένα συνεχές ρεύμα στοιχείων, πιθανώς άπειρο σε μήκος. Πολλές σύγχρονες εφαρμογές μπορούν να προταθούν ως τέτοια παραδείγματα : δεδομένα προερχόμενα από αισθητήρες, τιμές χρηματιστηριακών συναλλαγών, παρακολούθηση δραστηριότητας δικτύων, εφαρμογές τηλεδιαχείρισης. Με την διαδικτυακή έκρηξη που οι εφαρμογές απαιτούν απόκριση σε πραγματικό χρόνο, η χρήση ρευμάτων στοιχείων ως πηγών δεδομένων θα είναι όλο και συχνότερη. Ο χειρισμός ενός ρεύματος δεν είναι δυνατό να αναχθεί στην αποθήκευσή του σε μια παραδοσιακή βάση δεδομένων και στην περαιτέρω επεξεργασία του εκεί. Τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) αδυνατούν να αντεπεξέλθουν σε ερωτήματα των οποίων οι απαντήσεις χρειάζεται να ανανεώνονται με γρήγορο ρυθμό, ακολουθώντας την εισροή στοιχείων του ρεύματος.

2.1 Η ανεπάρκεια των συμβατικών συστημάτων

Ένα σύστημα που διαχειρίζεται μια παραδοσιακή βάση δεδομένων, σχεδιάζεται με γνώμονα την γρήγορη και κυρίως ακριβή απάντηση στα ερωτήματα του χρήστη. Η επίτευξη αυτών των στόχων προϋποθέτει την εκτενή μελέτη του περιβάλλοντος που καλείται να υποστηρίξει η βάση και την βελτιστοποίηση πολλαπλών υποσυστημάτων της. Πολλοί παράγοντες ,όπως το μέγεθος του αποθηκευτικού χώρου και η παραχώρηση προτεραιότητας ,μέσω της βελτιστοποίησης σε κάποιες μορφές ερωτημάτων που θα έχουν μεγαλύτερη συχνότητα, συμβάλλουν σε αυτό. Η τεχνολογία των παραδοσιακών ΣΔΒΔ έχει δοκιμαστεί και έχει ωριμάσει με το πέρασμα του χρόνου, τόσο ώστε πλέον να θεωρείται πλήρως αξιόπιστη. Η εμφάνιση όμως των εφαρμογών πραγματικού χρόνου, είναι μια περίπτωση όπου το υπάρχον μοντέλο διαχείρισης δεν μπορεί να χειριστεί αποτελεσματικά. Κύρια αιτία του γεγονότος αυτού είναι ο ριζικά διαφορετικός τρόπος χειρισμού που απαιτούν τα δεδομένα, ώστε να έχουμε λειτουργικά και αξιόπιστα προγράμματα.

Στα ΣΔΒΔ τα δεδομένα καταφθάνουν ή τροποποιούνται με σχετικά αργούς ρυθμούς, ώστε πρακτικά να μπορούν να θεωρηθούν στατικά. Τα δε ερωτήματα που επιλέγει ο χρήστης να θέσει υπόκεινται σε διαδικασίες βελτιστοποίησης κατά τον σχεδιασμό της βάσης, ώστε να μπορούν να εφαρμοστούν γρήγορα και αποδοτικά. Η εκτεταμένη χρήση μέσων αποθήκευσης όπως ο σκληρός δίσκος είναι σύνηθες φαινόμενο στα ΣΔΒΔ, αφού η συχνότητα στην πρόσβαση των δεδομένων δεν υπερβαίνει την τιμή στην οποία ο χρόνος απόκρισης των μέσων αυτών θα δημιουργούσε ανεπιθύμητη καθυστέρηση.

Σε εφαρμογές πραγματικού χρόνου, η ροή των δεδομένων εισόδου είναι αδιάκοπη, δυναμική και πολλές φορές απρόβλεπτη. Η δε απαιτούμενη απόκριση του συστήματος διαχείρισης, οφείλει να είναι ακαριαία και με ελάχιστο φόρτο για το υπολογιστικό σύστημα στο οποίο εκτελείται. Οι πρωταρχικές αυτές απαιτήσεις έρχονται σε σύγκρουση με θεμελιώδεις αρχές που διέπουν τα ΣΔΒΔ. Είναι φανερή η ανάγκη εκ θεμελίων τροποποίησής τους και ο ορισμός ενός νέου συστήματος διαχείρισης, ικανού να αντεπεξέλθει στις προκλήσεις αυτές.

2.2 Μοντέλο ρεύματος δεδομένων

Ένα ρεύμα δεδομένων μπορεί να αποτελείται από σχεσιακές πλειάδες, όπως και στα παραδοσιακά συστήματα. Ο χειρισμός τους όμως διαφέρει, λόγω του ότι δεν αποθηκεύονται πλέον σε σχεσιακούς πίνακες, αλλά επιδέχονται επεξεργασίας κατά την άφιξή τους. Το ρεύμα που αποτελεί είσοδο στο σύστημα, μπορεί να προέρχεται από την ένωση δύο ή περισσοτέρων άλλων ρευμάτων. Είναι απαραίτητος λοιπόν ένας τρόπος ταυτοποίησης και ταξινόμησης των στοιχείων του ρεύματος. Αυτό επιτυγχάνεται με ειδικά πεδία στις πλειάδες, που αναφέρουν την προέλευση, αλλά και ένα *χρονόσημο* (timestamp) που σηματοδοτεί είτε την χρονική στιγμή άφιξης στο σύστημα είτε την χρονική στιγμή δημιουργίας. Μετά την επεξεργασία ενός στοιχείου του ρεύματος αυτό στην συνηθέστερη περίπτωση απορρίπτεται, μιας και

το κόστος για την διατήρηση του συνόλου των δεδομένων είναι απαγορευτικό. Σε συστήματα όμως όπου μας ενδιαφέρει και το ιστορικό της εισόδου, ορισμένες πλειάδες αποθηκεύονται δημιουργώντας έτσι περιλήψεις των δεδομένων. Οι περιλήψεις αυτές έχουν πολύ μικρό μέγεθος συγκριτικά με αυτό του ρεύματος, ενώ δίνουν ικανοποιητική ακρίβεια στην περιγραφή τους. Συμπερασματικά επιχειρώντας έναν, όσο κατά το πιο δυνατόν, τυπικό ορισμό:

> *Ως ρεύμα δεδομένων ορίζεται μια συνεχής, πραγματικού χρόνου, πιθανώς απρόβλεπτη, ταξινομημένη (ρητά κατά χρόνο άφιξης ή έμμεσα με χρήση χρονοσήμων) και μη φραγμένη ακολουθία στοιχείων.*

Ένα *σύστημα διαχείρισης ρευμάτων δεδομένων* (ΣΔΡΔ) βασίζεται σαφώς σε αρκετές ιδέες και αρχές που διέπουν την λειτουργία των ΣΔΒΔ, τροποποιεί όμως και αναπροσαρμόζει άλλες. Μια σημαντική διαφοροποίηση είναι ότι ο ρόλος του χρήστη στην υποβολή των ερωτημάτων περιορίζεται. Η ίδια η φύση των εφαρμογών πραγματικού χρόνου απαιτεί από το σύστημα με την αδιάκοπη εισροή νέων δεδομένων στο σύστημα, να παράγει απόκριση σε αυτά με πολύ μεγάλη συχνότητα, αν όχι συνεχώς. Τα ερωτήματα συνήθως ορίζονται και αποθηκεύονται στο σύστημα, πριν την έναρξη της λειτουργίας του. Υπάρχει βεβαίως δυνατότητα για υποβολή καινούργιων ερωτημάτων, αλλά όπως θα περιγραφεί σε επόμενη ενότητα η υλοποίησή τους παρουσιάζει δυσκολίες. Τα δεδομένα που εισρέουν στο σύστημα μπορεί να έχουν την σχεσιακή μορφή των παραδοσιακών δεδομένων και να αποτελούνται από έναν αριθμό πλειάδων. Όμως υπάρχει και η περίπτωση όπου συνυπάρχουν δύο ή και παραπάνω πηγές - ρεύματα, με αποτέλεσμα να δημιουργείται η απαίτηση συνένωσης των στοιχείων τους και εξαγωγής συνολικών απαντήσεων.

Στα συστήματα διαχείρισης ρευμάτων δεδομένων έχουν τεθεί ορισμένες προδιαγραφές που θα πρέπει να τηρούνται ώστε να λειτουργούν με ορθό τρόπο [SCZ05]. Οι κανόνες αυτοί πηγάζουν από την διαφορετική φύση και λειτουργία των συστημάτων αυτών από τα παραδοσιακά ΣΔΒΔ. Επιγραμματικά οι προδιαγραφές είναι:

- Τα δεδομένα θα πρέπει να επεξεργάζονται από το σύστημα την στιγμή που έρχονται χωρίς την ανάγκη για προηγούμενη αποθήκευσή τους.
- Το σύστημα θα πρέπει να προσφέρει μια υψηλού επιπέδου γλώσσα ερωτημάτων, παρόμοια με την SQL, που όμως θα υποστηρίζει δομές και τελεστές ειδικά προσανατολισμένους στην διαχείριση ρευμάτων δεδομένων
- Είναι απαραίτητο να υπάρχουν μηχανισμοί στο σύστημα που θα εξασφαλίζουν την απρόσκοπτη λειτουργία του στην περίπτωση που αντιμετωπίζει ρεύματα με ατέλειες. Περιπτώσεις όπως ελλιπή δεδομένα ή πακέτα εκτός σειράς είναι συνήθεις και αποτελούν χαρακτηριστικό παράδειγμα τέτοιων ειδικών καταστάσεων.
- Το σύστημα διαχείρισης οφείλει να επιτυγχάνει προβλέψιμα και συνεπή αποτελέσματα με ντετερμινιστικό τρόπο.
- Το σύστημα πρέπει να παρέχει αποδοτικούς μηχανισμούς αποθήκευσης, ανάκλησης και τροποποίησης μεταβλητών κατάστασης και να μπορεί να συνδυάζει τις μεταβλητές αυτές με το ρεύμα δεδομένων εισόδου. Η διαδικασία αυτή

είναι θεμιτό να γίνεται αδιαφανώς για τον χρήστη, χρησιμοποιώντας και για τους δύο τύπους δεδομένων την ίδια γλώσσα χειρισμού.

- Κρίνεται απαραίτητη η εξασφάλιση της λειτουργίας και διαθεσιμότητας των εφαρμογών αλλά και η ακεραιότητα των δεδομένων συνεχώς, ανεξαρτήτως πιθανών αποτυχιών και βλαβών του συστήματος.

- Η χρήση πολλαπλών επεξεργαστών και/ή μηχανημάτων στην διαδικασία διαχείρισης και επεξεργασίας των ρευμάτων θα πρέπει ιδανικά να γίνεται αυτόματα και αδιαφανώς ως προς τον χρήστη.

- Ο μηχανισμός εκτέλεσης στην οποία θα βασίζεται το σύστημα οφείλει να αποδίδει απαντήσεις σε πραγματικό χρόνο ακόμα και για ρεύματα πολύ μεγάλου όγκου δεδομένων.

- Το μοντέλο δεδομένων και η σημασιολογία των ερωτημάτων θα πρέπει να επιτρέπουν τελεστές βασιζόμενους τόσο στην σειρά όσο και στον χρόνο (πχ. ερωτήματα σε δεδομένα που εισήλθαν τα τελευταία 5 λεπτά).

- Το γεγονός ότι είναι αδύνατο να αποθηκευτεί στην ολότητά του το ρεύμα δεδομένων, καθιστά αναγκαία την ύπαρξη δομών σύνοψης, που χρησιμοποιούνται για παροχή προσεγγιστικών απαντήσεων σε ερωτήματα που εμπλέκουν όλο το ρεύμα δεδομένων.

- Η γλώσσα ερωτημάτων δεν επιτρέπεται να περιέχει τελεστές που αναστέλλουν την διαδικασία ερώτησης μέχρι να διαβαστεί ολόκληρο το ρεύμα. Ειδικά για την συχνή περίπτωση εισόδου άπειρου μήκους, αυτό αποβαίνει καταστροφικό για το σύστημα.

2.2.1 Τελεστές

Οι τελεστές πράξεων των ΣΔΒΔ δεν μπορούν να αντεπεξέλθουν στο μοντέλο ρεύματος. Έτσι ένα νέο σύνολο τελεστών προτάθηκε [GO03] για να καλύψει την ανάγκη των εφαρμογών για ορθή και αποδοτική επεξεργασία ρεύματος δεδομένων. Οι τελεστές αυτοί αποτελούν κατά κύριο λόγο επεκτάσεις και εξειδικεύσεις των παραδοσιακών τελεστών των ΣΔΒΔ. Καθώς νέες εφαρμογές αναπτύσσονται, καινούργιοι τελεστές θα προταθούν για να καλύψουν το κενό στον χειρισμό των νέων αναγκών. Οι τελεστές που προτείνονται είναι:

- **Επιλογή (selection) :** Ικανοποιεί το πολύπλοκο φιλτράρισμα των δεδομένων που απαιτούν οι εφαρμογές.

- **Ένθετη συνάθροιση (nested aggregation) :** Η δυνατότητα προσδιορισμού "τάσεων" στα δεδομένα, απαιτεί πολύπλοκες συναθροίσεις, συμπεριλαμβανομένων και ένθετων.

- **Σύνδεση (join) :** Υποστήριξη τόσο για συνδυασμό πολλών ρευμάτων όσο και συνδυασμό ανάμεσα σε ρεύματα και στατικά μετα-δεδομένα.

- **Πολύπλεξη και Απόπλεξη (multiplexing and demultiplexing) :** σε αναλογία με τους παραδοσιακούς τελεστές ένωσης (union) και ομαδοποίησης (group-by) , οι τελεστές αυτοί αναλαμβάνουν την ένωση και την αποσύνθεση λογικών ρευμάτων.

- **Τελεστές συχνότητας (Frequent item queries) :** Χρησιμοποιούνται με κατάλληλη συνθήκη για να αποδώσουν συχνότητα εμφάνισης μιας τιμής ή ενός εύρους τιμών στο ρεύμα.

- **Τελεστές εξόρυξης δεδομένων (stream mining) :** Παρέχουν λειτουργίες όπως ταίριασμα προτύπων, αναζήτηση ομοιότητας και πρόβλεψης.

Όλοι οι παραπάνω τελεστές πρέπει να μπορούν να εφαρμοστούν και σε έκταση χρόνου, όπως για παράδειγμα στα δεδομένα που αφορούν την τελευταία βδομάδα.

2.3 Κατηγορίες ερωτημάτων

Η έννοια των ερωτημάτων στα συστήματα διαχείρισης ρευμάτων δεδομένων είναι άρρηκτα συνδεδεμένη με την αντίστοιχη για ερωτήματα σε παραδοσιακές βάσεις δεδομένων. Όμως η ίδια η φύση των δεδομένων αλλά και των απαιτήσεων των εφαρμογών που στηρίζονται πάνω στα συστήματα αυτά, διαφοροποιούν τη σημασιολογία των ερωτημάτων και την λειτουργία των τελεστών τους. Σε ένα ΣΔΒΔ ο χρήστης επιλέγει την στιγμή και τον τρόπο με τον οποίο θα θέσει ένα ερώτημα, πάνω σε ένα στατικό και αποθηκευμένο σύνολο δεδομένων, αναμένοντας μια εξίσου στατική και προδιαγεγραμμένη απάντηση. Το μοντέλο όμως ερωτημάτων σε ρεύματα δεδομένων λειτουργεί αλλιώς. Τα δεδομένα πλέον παύουν να είναι στατικά, αλλά καταφθάνουν στο σύστημα με ένα δυναμικό ρυθμό και το μέγεθός τους δεν είναι απαραίτητα φραγμένο. Ο χρήστης οριοθετεί τις ανάγκες και τις επιθυμίες του στην αρχή της λειτουργίας. Το σύστημα αναλαμβάνει τις διεκπεραιώσει, παρέχοντας απαντήσεις στον χρήστη. Αν και είναι δυνατή η εκ των υστέρων υποβολή νέου ερωτήματος, συνήθως αυτά είναι γνωστά από πριν. Το μοντέλο αυτό θα γίνει πιο κατανοητό αν εξετάσουμε τις κατηγορίες ερωτημάτων σε ένα ΣΔΡΔ.

Ο πρώτος προφανής διαχωρισμός είναι ανάμεσα στα ερωτήματα *στιγμιοτύπου* (one-time queries) και στα ερωτήματα *διαρκείας* (continuous queries) . Τα ερωτήματα στιγμιοτύπου αποτελούν ευθεία αναλογία των ερωτημάτων που τίθενται στα κλασικά ΣΔΒΔ, όπου το ερώτημα εξετάζεται και απαντάται μια φορά, βάσει των δεδομένων που υπάρχουν μέσα στο σύστημα εκείνη τη στιγμή. Για παράδειγμα σε ένα σύστημα παρακολούθησης αισθητήρων θερμοκρασίας, το ερώτημα που επιστρέφει τους 3 κόμβους με τις μεγαλύτερες θερμοκρασίες κατά την διάρκεια των 10 τελευταίων λεπτών. Φυσικά η δυναμική των δεδομένων απαιτεί να οριστούν επιπρόσθετοι ή να τροποποιηθούν υπάρχοντες τελεστές των κλασικών ΣΔΒΔ, με αποτέλεσμα η κατηγορία αυτή ερωτημάτων να αποτελεί ένα υπερσύνολο των τυπικών σχεσιακών τελεστών.

Εκτός από τα ερωτήματα στιγμιοτύπου, υπάρχουν και τα ερωτήματα διαρκείας , που εκτελούνται στο σύστημα επί μακρόν. Η απάντηση σε ένα ερώτημα διαρκείας είναι δυναμική και αλλάζει καθώς καταφθάνουν νέα δεδομένα. Η απόκριση του

συστήματος αντικατοπτρίζει το ρεύμα δεδομένων που έχει εισαχθεί στο σύστημα ακριβώς μέχρι εκείνη την χρονική στιγμή. Για παράδειγμα σε ένα σύστημα δικτυακής παρακολούθησης το ερώτημα που επιστρέφει ανά πάσα στιγμή τους κόμβους με την μεγαλύτερη συμφόρηση πακέτων. Αν και η αποθήκευση της μέχρι τώρα απάντησης είναι θεμιτή, από ένα χρονικό διάστημα και μετά αυτό καθίσταται πρακτικά αδύνατο, λόγω του πιθανώς άπειρου μεγέθους που μπορεί να φτάσει. Το πρόβλημα αυτό έχει μελετηθεί ευρέως και ανάμεσα στις λύσεις που προτάθηκαν είναι η χρήση συνόψεων και περιλήψεων αλλά και ο φραγμός με παράθυρα χρόνου της εφαρμογής των ερωτημάτων. Το ερευνητικό ενδιαφέρον σε αυτήν την κατηγορία είναι μεγάλο, καθώς η σωστή και αποδοτική σχεδίαση και υλοποίησή τους αποτελεί πρόκληση.

Ένας δεύτερος διαχωρισμός μπορεί να επέλθει ανάλογα με το αν τα ερωτήματα είναι *προκαθορισμένα* (predefined queries) ή *μη προβλέψιμα* (ad hoc queries). Τα προκαθορισμένα ερωτήματα δίνονται στο σύστημα διαχείρισης πριν την άφιξη των δεδομένων και μπορεί να είναι είτε στιγμιοτύπου είτε διαρκείας. Τα μη προβλέψιμα ερωτήματα από την άλλη μεριά τίθενται κατά την διάρκεια της επεξεργασίας του ρεύματος δεδομένων, συχνά επειδή τα μέχρι τώρα δεδομένα που έχουμε επεξεργαστεί απαιτούν ένα τέτοιο ερώτημα. Η δυσκολία στην αντιμετώπισή τους έγκειται στο ότι δεν μπορούν να βελτιστοποιηθούν αφού δεν είναι a priori γνωστά. Επιπλέον, για την σωστή απάντησή τους μπορεί να απαιτούν δεδομένα που έχουν αφιχθεί σε πρότερη χρονική στιγμή, τα οποία πιθανώς να έχουν απορριφθεί από το σύστημα. Όπως και με τα προκαθορισμένα, μπορούν να είναι είτε στιγμιοτύπου είτε διαρκείας.

2.4 Προβλήματα σε ερωτήματα πάνω σε ρεύματα δεδομένων

2.4.1 Μη φραγμένες απαιτήσεις μνήμης

Όπως έχει αναφερθεί και προηγουμένως, η ροή των δεδομένων είναι συνεχής, αλλά επίσης είναι πιθανώς άπειρη (μη φραγμένη) σε μέγεθος. Είναι λογικό λοιπόν ότι η αποθήκευση του ρεύματος δεδομένων στην ολότητά του είναι πρακτικά αδύνατη στην πλειοψηφία των περιπτώσεων. Ο μοναδικός τρόπος για να αποθηκεύσουμε όλο τον όγκο των δεδομένων, θα ήταν σε συστοιχίες σκληρών δίσκων. Η λύση αυτή όμως εκτός από δαπανηρή και δύσκολα κλιμακούμενη, αφαιρεί αυτομάτως από τις εφαρμογές διαχείρισης ρευμάτων δεδομένων την δυνατότητα να αποκρίνονται σε πραγματικό χρόνο στην είσοδο που δέχονται. Το κόστος ανάκτησης των δεδομένων από τους δίσκους, όπως επίσης και της πολύ πιθανής ανάγκης για συνένωση δεδομένων που βρίσκονται σε διαφορετικές φυσικές διαμερίσεις, είναι τέτοιο που δεν αφήνει περιθώρια απόκρισης πραγματικού χρόνου.

Η χρήση της κύριας μνήμης μπορεί να προσφέρει μεν ταχύτατες αποκρίσεις, το μέγεθός της όμως αποτελεί αδιαμφισβήτητα δεσμευτικό παράγοντα. Σε αυτά πρέπει να προσθέσουμε και το γεγονός ότι σε μια τέτοια εφαρμογή, η επεξεργασία των δεδομένων γίνεται παράλληλα με την υποδοχή νέων. Άρα είναι αδύνατη η διακοπή όλων των υπολοίπων λειτουργιών του συστήματος, ώστε να απελευθερωθούν οι

απαραίτητοι πόροι για την εξυπηρέτηση ερωτημάτων. Αυτό θα οδηγούσε αυτόματα στην απώλεια τμήματος του ρεύματος δεδομένων εισόδου.

Εξάγουμε λοιπόν αβίαστα το συμπέρασμα ότι υπάρχει ανάγκη για συστήματα διαχείρισης που έχουν πολύ μικρό χρόνο επεξεργασίας ανά μονάδα δεδομένων και με αλγόριθμους που εκμεταλλεύονται την κύρια μνήμη, χωρίς την ανάγκη πρόσβασης σε σκληρούς δίσκους. Έχουν γίνει κάποιες προσπάθειες [ABB+02], ώστε να γίνει διάκριση μεταξύ ερωτημάτων που μπορούν να απαντηθούν επακριβώς, δοσμένου ενός ορισμένου ποσού μνήμης και ερωτημάτων που πρέπει να απαντηθούν προσεγγιστικά, για να αποφύγουμε την πρόσβαση στον δίσκο. Αν και η μελέτη αφορά περιορισμένο σύνολο τελεστών, προκύπτει ότι δίχως να υπάρχει πρότερη γνώση του μεγέθους του ρεύματος δεδομένων εισόδου, είναι αδύνατο να τοποθετηθούν όρια στην απαιτούμενη μνήμη. Μόνο αν περιορίσουμε την πολυπλοκότητα στην δομή των εισερχόμενων δεδομένων μπορεί να βρεθεί η ποσότητα της μνήμης που χρειαζόμαστε. Το πρόβλημα αποτελεί ανοιχτό ερευνητικό πεδίο.

2.4.2 Προβληματικοί τελεστές

Όπως αναφέρθηκε και πριν, οι τελεστές ερωτημάτων που προτάθηκαν, στην ουσία αποτελούσαν επεκτάσεις των δοκιμασμένων και λειτουργικών τελεστών των ΣΔΒΔ. Όμως υπήρξαν και περιπτώσεις, όπου η επέκταση δεν ήταν απλή υπόθεση. Κάποιοι τελεστές αδυνατούσαν να προσαρμοστούν στο νέο μοντέλο δεδομένων, αφού στοιχειώδεις λειτουργίες τους συγκρούονταν με βασικές αρχές του.

Ανασχετικοί τελεστές

Ανασχετικούς τελεστές (blocking operators) ονομάζουμε τους τελεστές, που αναμένουν την πλήρη συλλογή της εισόδου, πριν αρχίσουν να αποτιμούν την έξοδο. Τέτοιοι για παράδειγμα είναι οι τελεστές αθροίσματος (SUM), μέσου όρου (AVERAGE) αλλά και ταξινόμησης. Η εφαρμογή τους σε ένα ρεύμα δεδομένων, εφόσον αυτό είναι πιθανόν άπειρο σε μήκος, μπορεί να προκαλέσει την επ΄ αόριστον διακοπή της εκτέλεσης του προγράμματος, εφόσον θα αναμένεται από τον τελεστή αποτέλεσμα.

Το μέγεθος του προβλήματος που προκαλείται στην λειτουργία του συστήματος, εξαρτάται από τη θέση του ανασχετικού τελεστή στο *δένδρο εκτέλεσης* του ερωτήματος (query tree). Η επίδρασή του είναι πολύ μεγαλύτερη αν αποτελεί φύλλο από το αν βρίσκεται στην ρίζα του προσχεδίου εκτέλεσης. Ο λόγος είναι ότι ένα φύλλο με ανασχετικό τελεστή θα παγώσει οποιαδήποτε αποτίμηση βρίσκεται σε υψηλότερο επίπεδο καθώς θα αναμένει το αποτέλεσμά του. Αν ο τελεστής βρίσκεται στην ρίζα, είναι δυνατό το πρόβλημα να παρακαμφθεί, αν σε τακτά χρονικά διαστήματα ο τελεστής δίνει μέρος της απάντησης. Η τεχνική αυτή όμως δεν μπορεί να λειτουργήσει για όλων των ειδών τα ερωτήματα.

Μια διαφορετική προσέγγιση προτείνει να αντικατασταθούν οι προβληματικοί τελεστές, από νέους, μη ανασχετικούς τελεστές που επιτελούν την ίδια δουλειά. Ως παράδειγμα στην βιβλιογραφία [RRH99] απαντάται ο τελεστής *juggle*, ένας μη ανασχετικός τελεστής που προτείνεται στην θέση του ανασχετικού sort. Τα αποτελέσματά του έχουν ικανοποιητική προσέγγιση ως προς την ακρίβεια, αλλά

το λάθος που υπεισέρχεται δεν μπορεί ακόμα να εκτιμηθεί. Δυστυχώς δεν έχουν βρεθεί ακόμα αντίστοιχες λύσεις για όλους τους ανασχετικούς τελεστές και άρα η λύση περιορίζεται σε μικρό αριθμό αυτών.

Μια εύκολη στην υλοποίηση, αλλά και αποδοτική λύση, είναι ο χωρισμός του θεωρητικά άπειρου ρεύματος δεδομένων, σε πολλά πεπερασμένου μήκους κομμάτια. Η αρχή και το τέλος κάθε τέτοιου υπορεύματος σηματοδοτείται από ειδικές πλειάδες, που τοποθετούνται εμβόλιμα και ονομάζονται *σημεία στίξης* (punctuations) [TMSF02],[TMS003]. Τα σημεία στίξης πλήν του χωρισμού έχουν και σημασιολογική χρησιμότητα, αφού δίνουν πληροφορίες για τα τμήματα του ρεύματος που έπονται. Για παράδειγμα μπορούν να σηματοδοτούν ότι στις επόμενες πλειάδες, μια ιδιότητα έχει τιμή πάνω από ένα συγκεκριμένο όριο. Αυτό μπορεί να το εκμεταλλευτεί το σύστημα, επιστρέφοντας οριστικά μια απάντηση εφόσον οι επόμενες πλειάδες δεν μπορούν να μεταβάλουν την τιμή της.

Τελεστές διατήρησης κατάστασης

Οι τελεστές *διατήρησης κατάστασης* (unbounded stateful operators) αποτελούν μια δεύτερη κατηγορία προβληματικών τελεστών. Οι τελεστές αυτοί διατηρούν τις τιμές όλων των πλειάδων που εμπλέκονται στο ερώτημα. Παραδείγματα τέτοιων τελεστών αποτελούν η *σύνδεση* (join) μεταξύ ρευμάτων ή μεταξύ ρευμάτων και στατικών σχέσεων, η *τομή* (intersect) ,αλλά και οι *ανασχετικοί* τελεστές. Με μεγάλη πιθανότητα το μέγεθος του ρεύματος δεδομένων μπορεί να οδηγήσει έναν τέτοιο τελεστή να εξαντλήσει το διαθέσιμο ποσό μνήμης, εφόσον αυτό αυξάνεται χωρίς φραγμό.

Στην αντιμετώπιση του προβλήματος μπορούν να συμβάλλουν τα σημεία στίξης. Μπορούν δηλαδή να χαρακτηρίζουν το τμήμα που επακολουθεί κατά τέτοιο τρόπο, ώστε να είναι δυνατή η απόρριψη μέρους του ρεύματος που είναι κρατημένο στην μνήμη και το οποίο δεν χρειάζεται. Για παράδειγμα ένα σημείο στίξης που σηματοδοτεί ότι το πεδίο της σύνδεσης έχει τιμή μεγαλύτερη από όλες τις τιμές του αντίστοιχου πεδίου ενός δεύτερου ρεύματος, οδηγεί σε άμεση αποτίμηση, χωρίς να χρειάζεται να εξεταστούν οι επερχόμενες τιμές.

2.5 Τεχνικές προσέγγισης

Στα συστήματα διαχείρισης ρευμάτων δεδομένων, το στοιχείο που πρέπει να διασφαλιστεί είναι η γρήγορη και σωστή απόκριση στα ερωτήματα. Σε επίπεδο υλοποίησης όμως, το διαθέσιμο ποσό μνήμης δεν είναι ανεξάντλητο, οι υπολογισμοί κοστίζουν ένα μικρό μεν αλλά σημαντικό χρονικό διάστημα, ενώ το ρεύμα των δεδομένων εισόδου είναι θεωρητικά άπειρο και απρόβλεπτο ως προς τον ρυθμό με τον οποίο έρχεται. Σε αυτές τις συνθήκες η ακρίβεια και η ταχύτητα στην απόκριση αλληλοσυγκρούονται, με αποτέλεσμα να χρειάζονται συμβιβασμοί για να επιτύχουμε μια ισορροπία ανάμεσα στα δύο.

Στα ΣΔΒΔ το πλαίσιο λειτουργίας είναι τέτοιο που και οι δύο παράμετροι μπορούν να ικανοποιηθούν σε μεγάλο βαθμό. Στα ΣΔΡΔ όμως, με την λογική ότι είναι προτιμότερη η απρόσκοπτη ροή του συστήματος, ο όρος σωστή απόκριση

μπορεί να αναθεωρηθεί ως προς την αυστηρότητα. Με άλλα λόγια σε κάποιες περιπτώσεις είναι προτιμότερο η απάντηση που δίνει το σύστημα να εμπεριέχει ένα ποσοστό προσέγγισης, αλλά να παρέχεται χωρίς καθυστέρηση. Στην συνέχεια θα περιγραφούν ορισμένες τεχνικές προσέγγισης που χρησιμοποιούνται για τον σκοπό αυτό.

2.5.1 Περιλήψεις

Οι *περιλήψεις ή συνόψεις δεδομένων* (summaries or data synopses) αποτελούν την συνοπτική περιγραφή των δεδομένων με έναν δεδομένο παράγοντα ανακρίβειας. Το μέγεθός τους είναι σημαντικά μικρότερο από αυτό των συνολικών δεδομένων, συνήθως λογαριθμικά ή πολυλογαριθμικά μικρότερο σε τάξη μεγέθους. Επίσης η δημιουργία τους πρέπει να επιτυγχάνεται με ένα πέρασμα των δεδομένων και με τη σειρά που αυτά καταφθάνουν. Η χρήση των συνόψεων αποσκοπεί στο να δώσει μια ικανοποιητική λύση σε προβλήματα όπου δεν υπάρχουν δομές δεδομένων κατάλληλες να διαχειριστούν επαρκώς κάποιους τύπους ερωτημάτων. Σε περιπτώσεις όπου ο διαθέσιμη μνήμη είναι περιορισμένη, το δραστικά μικρότερο μέγεθος των περιλήψεων μαζί με την δυνατότητα αξιοποίησής τους από τον επεξεργαστή ερωτημάτων για εξαγωγή αρκετά αξιόπιστων αποτελεσμάτων, τις καθιστούν ένα πολύ χρήσιμο εργαλείο. Πρέπει να τονιστεί η ιδιαίτερη χρησιμότητά τους σε ερωτήματα σύνδεσης, με παράλληλη χρήση τελεστών συνάθροισης. Ακολουθεί η περιγραφή ορισμένων τεχνικών περιλήψεως.

Σκίτσα δεδομένων

Η τεχνική των *σκίτσων δεδομένων* αποσκοπεί σε παραγωγή περίληψης με τυχαίο τρόπο (randomized sketching). Χρησιμοποιείται σε ερωτήσεις απόστασης ή υπολογισμού του πλήθους των διακριτών τιμών ενός ρεύματος δεδομένων, χρησιμοποιώντας μικρό μέγεθος μνήμης.

Κυματίδια

Μια τεχνική περίληψης προερχόμενη από το πεδίο της θεωρίας σημάτων, είναι η τεχνική των *κυματιδίων* (wavelets). Στηρίζεται σε μαθηματικούς μετασχηματισμούς που αποτυπώνουν την πληροφορία με αριθμητικές συναρτήσεις, οι συντελεστές των οποίων είναι προβολές του συνόλου δεδομένων, σε ένα ορθοκανονικό σύνολο *διανυσμάτων αναφοράς* (basic vectors). Ο τύπος των κυματιδίων εξαρτάται από τα διανύσματα αναφοράς που επιλέγονται. Στις βάσεις δεδομένων εφαρμόζονται τα διανύσματα Haar λόγω της ευκολίας υπολογισμού και αναπαράστασής τους με ένα δυαδικό δέντρο. Η σπουδαιότητα των κυματιδίων στηρίζεται στην ιδιότητα ότι πρακτικά από μικρό πλήθος των σημαντικότερων συντελεστών των γραμμικών προβολών, μπορούν να αναπαρασταθούν τα πρωτότυπα δεδομένα με κριτήριο την ευκλείδεια νόρμα τους.

Αν και τα κυματίδια δεν είναι εφαρμόσιμα για όλους τους τύπους ερωτημάτων, εντούτοις σε απλά ερωτήματα συνάθροισης για μεμονωμένα χαρακτηριστικά των στοιχείων, μπορούν να δώσουν απάντηση άμεσα και με καλή προσέγγιση, έχοντας πολύ μικρό αποθηκευτικό κόστος. Ακόμα και σε πιο πολύπλοκα ερωτήματα συνάθροισης, αν και η αποτελεσματικότητά τους μειώνεται, παραμένουν χρήσιμα και αξιόπιστα.

Ιστογράμματα

Τα *ιστογράμματα* αποτελούν μια εξαιρετικά δημοφιλή δομή στις βάσεις δεδομένων για τη συνοπτική αναπαράσταση της κατανομής των τιμών ενός συνόλου δεδομένων (λ.χ. μία ή περισσότερες στήλες ενός πίνακα). Το αναμενόμενο εύρος του συνόλου τιμών χωρίζεται σε διαστήματα για καθένα από τα οποία διατηρείται η συχνότητα τιμών που καταφτάνει, σχηματίζοντας έτσι ένα ιστόγραμμα, το οποίο μπορεί να αναπαρασταθεί και με ένα διάγραμμα. Σαφώς, το πλήθος και το εύρος των διαστημάτων συνηγορεί στην ακρίβεια της αναπαράστασης. Ένα ρεύμα δεδομένων πρέπει να χωρίζεται σε τέτοια διαστήματα από συγκεκριμένους αλγόριθμους οι οποίοι θα επιτρέπουν τον δυναμικό προσδιορισμό των διαστημάτων. Κατόπιν, βάσει του εύρους των τιμών κάθε διαστήματος επιχειρείται να εντοπισθεί η αντιπροσωπευτικότερη τιμή, η οποία τελικά θα διατηρηθεί αποθηκευμένη. Η ανασύσταση του συνόλου των δεδομένων από το ιστόγραμμα γίνεται με βάση τις αντιπροσωπευτικές τιμές κάθε διαστήματος και τη συχνότητα των στοιχείων που αντιστοιχεί στο εν λόγω διάστημα. Ενώ έχουν προταθεί αρκετοί αλγόριθμοι ιστογραμμάτων (λ.χ. V-optimal, ίσου πλάτους κτλ), το ζήτημα εύρεσης δυναμικών αλγορίθμων προσδιορισμού των διαστημάτων με βάση τις πλειάδες που καταφτάνουν, επιδέχεται αρκετή έρευνα.

Αποβολή φόρτου

Σε ορισμένα συστήματα διαχείρισης ρευμάτων παρέχεται η επιλογή στον χρήστη να ρυθμίσει την *παρεχόμενη ποιότητα υπηρεσιών* (Quality of Service, QoS) των ερωτημάτων που θέτει στο σύστημα. Έχοντας θέσει διάφορες απαιτήσεις στην ποιότητα των απαντήσεων, το σύστημα προσπαθεί να ανταπεξέλθει στο ζητούμενο επίπεδο. Αυτό γίνεται με την αξιολόγηση χαρακτηριστικών των αποτελεσμάτων που προκύπτουν στο ρεύμα εξόδου όπως η αξιοπιστία, οι επιδόσεις και η ακρίβεια. Με την τεχνική της *αποβολής φόρτου* (load shedding) [TCZ+03] το σύστημα μπορεί να ανταπεξέλθει σε δύσκολες καταστάσεις όπως λ.χ. όταν τα δεδομένα καταφθάνουν πιο γρήγορα από ό,τι μπορούν να επεξεργαστούν, χωρίς να δημιουργούνται ανεπιθύμητες καθυστερήσεις στην παραγωγή του ρεύματος εξόδου. Η τεχνική βασίζεται στην ιδέα της αποβολής πλειάδων από το ρεύμα εισόδου, είτε τυχαία είτε βάσει κάποιας συνάρτησης αξιολόγησης της σημαντικότητάς τους, με την αποδοχή βέβαια μιας μειωμένης ακρίβειας στα αποτελέσματα. Σε συστήματα διαχείρισης όπως το Aurora, η αποβολή φόρτου πραγματοποιείται δυναμικά όταν αυτό κρίνεται απαραίτητο, αξιολογώντας έναν *δείκτη χρησιμότητας* (utility value)

που υπολογίζεται επί των χαρακτηριστικών ποιότητας που έχει θέσει ο χρήστης. Σε περιπτώσεις που ο δείκτης αυτός μειωθεί κάτω από το αποδεκτό, το σύστημα αποβάλει μέρος των δεδομένων, ελαφρύνοντας το φόρτο. Έτσι είναι δυνατό να παρέχονται συνεχώς ικανοποιητικές απαντήσεις, που βεβαίως εμφανίζουν μετρήσιμες αποκλίσεις ως προς την ακρίβειά τους.

2.5.2 Προσδιορισμός παραθύρου στα δεδομένα

Μια πολύ χρήσιμη τεχνική για την απόδοση μιας προσεγγιστικής απάντησης σε ερωτήματα πάνω σε ρεύματα δεδομένων, είναι τα *χρονικά παράθυρα*. Η κεντρική ιδέα είναι ότι τα ερωτήματα πλέον δεν αφορούν το σύνολο των δεδομένων που έχουν εισαχθεί στο σύστημα, αλλά μόνο ένα υποσύνολο αυτών. Το πεπερασμένο υποσύνολο αυτό αποτελεί το παράθυρο. Σύνηθες παράδειγμα αποτελεί το παράθυρο στα δεδομένα που αφορούν την καταγραφή στοιχείων για την τελευταία εβδομάδα μόνο και όχι για ολόκληρη την περίοδο λειτουργίας του.

Τα χρονικά παράθυρα αποτελούν μια ντετερμινιστική μέθοδο υπολογισμού, εξαλείφοντας τον κίνδυνο μεγάλου σφάλματος, περίπτωση που είναι κοινή σε άλλες προσεγγιστικές μεθόδους υπολογισμού απαντήσεων. Η τακτική της επεξεργασίας μόνο των πιο πρόσφατων δεδομένων, συμβαδίζει με αυτή των εφαρμογών πραγματικού χρόνου. Πιο σημαντικές και καλύτερα αξιοποιήσιμες απαντήσεις προκύπτουν από πιο πρόσφατα και πιο αντιπροσωπευτικά δεδομένα. Οι παράμετροι του παραθύρου επιλέγονται από τον χρήστη κατά την υποβολή του ερωτήματος. Έτσι η εφαρμογή τους προσφέρει διαφάνεια ως προς την προσέγγιση που γίνεται, αφού ο χρήστης είναι ενήμερος για τις συνθήκες που δημιουργούνται ώστε να επιτύχουμε την απάντηση.

Γίνεται κατανοητό ότι εφόσον η μέθοδος των παραθύρων χρησιμοποιεί κάποιου είδους διάταξη για να επιλέξει τα διαθέσιμα για επεξεργασία δεδομένα, αυτή η διάταξη οφείλει να ενυπάρχει στα ίδια τα δεδομένα. Στα στοιχεία του ρεύματος δεδομένων λοιπόν πρέπει να υπάρχουν *χρονόσημα* (timestamps), δηλαδή ενδείξεις που δηλώνουν συνήθως την χρονική στιγμή παραγωγής τους ή την χρονική στιγμή άφιξης τους στο σύστημα.

Η υποστήριξη των χρονικών παραθύρων από γλώσσες υψηλού επιπέδου όπως η SQL ή η σχεσιακή άλγεβρα, αποτελεί πολύπλοκο πρόβλημα και απαιτεί σημαντικές αλλαγές στους ορισμούς των εντολών. Ακόμα σημαντικότερα και πιο δυσεπίλυτα προβλήματα δημιουργούνται στην βελτιστοποίηση των ερωτημάτων. Σε περιπτώσεις όπου τα παράθυρα είναι τόσο μεγάλα που δεν χωράνε στην ολότητά τους στην κύρια μνήμη, θα πρέπει να χρησιμοποιηθούν αλγόριθμοι που δίνουν προσεγγιστικές απαντήσεις βασιζόμενοι όμως μόνο στην διαθέσιμη μνήμη.

2.6 Πρωτότυπα συστήματα διαχείρισης ρευμάτων δεδομένων

2.6.1 AURORA

Το *AURORA* αποτελεί αποτέλεσμα της σύμπραξης των πανεπιστημίων Brandeis, Brown και του M.I.T. Είναι ένα σύστημα διαχείρισης ρεύματων δεδομένων γενικού σκοπού και πρωτοεμφανίστηκε το 2001. Το σύστημα υποβολής ερωτημάτων χρησιμοποιεί μια ιδέα παρόμοια με αυτή των διαγραμμάτων ροής δεδομένων, δίνοντας στον χρήστη μονάδες όπως "κουτιά" και "βέλη" για να τα σχεδιάσει σε ένα κατάλληλο γραφικό περιβάλλον. Τα επιμέρους κομμάτια της αρχιτεκτονικής του συστήματος συντονίζονται από ένα χρονοπρογραμματιστή (scheduler). Οι πλειάδες που καταλήγουν στην έξοδο, παρακολουθούνται μονίμως από τον Επόπτη Ποιότητας (QoS Monitor), ανατροφοδοτώντας τον χρονοπρογραμματιστή με χρήσιμα στοιχεία για τη συνολική επίδοση του συστήματος. Υπάρχει επίσης ένας διαχειριστής αποθήκευσης (storage manager). Το AURORA παρέχει στους χρήστες την δυνατότητα να επιλέγουν δυναμικά την ποιότητα υπηρεσιών που επιθυμούν για την υλοποίηση των ερωτημάτων.

2.6.2 STREAM (STanford stREam datA Management)

Το *STREAM* είναι επίσης ένα ΣΔΡΔ γενικού σκοπού, που αναπτύχθηκε το 2001 στο Πανεπιστήμιο του Stanford. Αποτελεί εξολοκλήρου πρωτότυπη εργασία, αφού οι σχεδιαστές του επέλεξαν να μην βασιστούν σε κάποιο υπάρχον ΣΔΒΔ, αλλά να δημιουργήσουν εκ θεμελίων ένα νέο, ολοκληρωμένο ΣΔΡΔ. Για τον σκοπό αυτό κινήθηκαν γύρω από τους εξής άξονες:

- Να παρέχει έναν ευέλικτο τρόπο διεπαφής, προκειμένου να διευκολύνεται η ανάγνωση και η εγγραφή ρευμάτων δεδομένων.
- Να επιτυγχάνεται η αποτελεσματική επεξεργασία των ερωτημάτων διαρκείας που διατυπώνονται σε SQL ή με τελεστές της σχεσιακής άλγεβρας, συμπεριλαμβανομένων των συναθροιστικών.
- Να προσφέρεται ένα περιβάλλον διασύνδεσης (API) για την υποβολή των ερωτημάτων διαρκείας και τη λήψη των απαντήσεων σε αυτά.

Για την υποβολή των ερωτημάτων γίνεται χρήση της ειδικά διαμορφωμένης δηλωτικής γλώσσας ερωταποκρίσεων CQL (Continuous Query Language). Συντακτικά, η CQL είναι υπερσύνολο της SQL, με προσθήκη εξειδικευμένων δομών για την υποστήριξη κυλιόμενων παραθύρων και δειγματοληψίας. Σημαντικό στοιχείο της γλώσσας, αποτελεί το γεγονός ότι η σημασιολογία των ερωτημάτων διαρκείας αντιμετωπίζει με παρόμοιο τρόπο, τόσο τα δεδομένα των ρευμάτων, όσο και εκείνα των στατικών σχέσεων.

2.6.3 TelegraphCQ

Το *TelegraphCQ* αποτελεί μια εξέλιξη του πρωτότυπου Telegraph, που συντονίζεται από το πανεπιστήμιο του Berkeley. Στη σχεδίαση του συστήματος δίνεται έμφαση σε δικτυακά περιβάλλοντα, κυρίως για δίκτυα αισθητήρων. Το σύστημα διαθέτει μια πρωτότυπη προσέγγιση στα ρεύματα δεδομένων αφού θεωρείται ότι δεν είναι μόνο τα δεδομένα που εμφανίζουν μεταβλητότητα και ρέουν μέσα στο δίκτυο, αλλά και τα ίδια τα ερωτήματα διαρκείας μπορούν να παρομοιαστούν με ρεύματα, μιας και τόσο ο αριθμός τους όσο και η δομή τους αλλάζει κατά απρόβλεπτο τρόπο με την πάροδο του χρόνου. Η ανάπτυξή του στηρίχθηκε στην προσαρμογή της αρχιτεκτονικής της PostgreSQL προγραμματίζοντας σε C/C++, ώστε να καταστεί εφικτή η από κοινού επεξεργασία ερωτημάτων διαρκείας επί ρευμάτων δεδομένων.. Το ενδιαφέρον εστιάζεται κυρίως στην ευελιξία και προσαρμοστικότητα εκτέλεσης ερωτημάτων διαρκείας, εισάγοντας τον μηχανισμό Eddy.

2.7 Παρακολούθηση κινούμενων αντικειμένων

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, τα συστήματα που επιτυγχάνουν την παρακολούθηση σημειακών θέσεων κινούμενων αντικειμένων αποτελούν ένα ανοιχτό ερευνητικό πεδίο. Οι διαδοχικές θέσεις της πορείας ενός αντικειμένου, μπορούν να θεωρηθούν ότι συγκροτούν ένα ρεύμα δεδομένων. Σε πραγματικές εφαρμογές παρακολούθησης κινούμενων αντικειμένων, όπως για παράδειγμα ενός στόλου οχημάτων, οι σημειακές θέσεις είναι πολυάριθμες και συνήθως καταφθάνουν με καταιγιστικό ρυθμό. Επίσης τα ερωτήματα που ενδιαφέρουν είναι συνήθως ερωτήματα διαρκείας είτε για ολόκληρο το διάστημα κίνησής τους, είτε για ένα πρόσφατο χρονικό διάστημα. Τα χαρακτηριστικά λοιπόν των συστημάτων αυτών παρουσιάζουν μεγάλες ομοιότητες με τα συστήματα διαχείρισης ρευμάτων δεδομένων που περιγράφηκαν. Με τις κατάλληλες επεκτάσεις είναι δυνατός ο αποδοτικός χειρισμός των προβλημάτων αυτών από τα ΣΔΡΔ.

Αν θεωρήσουμε ότι τα υπό παρακολούθηση αντικείμενα είναι σημειακά (δηλαδή είναι αδιάφορη η χωρική τους έκταση), η πληροφορία κίνησής τους που καταφθάνει σαν ρεύμα δεδομένων αποτελεί ένα *ρεύμα τροχιάς αντικειμένων* (trajectory stream). Συνήθως εκτός από τις συντεταγμένες θέσης οι πλειάδες τέτοιων ρευμάτων φέρουν και χρονόσημα, που καθορίζουν είτε την χρονική στιγμή αναφοράς, είτε την χρονική στιγμή λήξης της εγκυρότητας της μέτρησης που αποστέλλεται. Επιπλέον ιδιότητες των αντικειμένων που περιγράφουν την κατάστασή τους, όπως η στιγμιαία ταχύτητα ή η θερμοκρασία, συχνά αποτελούν μέρος των πλειάδων για την κατάλληλη αξιοποίησή τους από το σύστημα.

Τα ερωτήματα που τίθενται είναι συνήθως διαρκείας, επειδή η διαρκώς μεταβαλλόμενη πληροφορία του ρεύματος καθιστά μη πρακτική την συνεχόμενη υποβολή ερωτημάτων για να διαπιστωθούν πιθανές αλλαγές στα στοιχεία. Η τάση αυτή καθιστά απαραίτητη την ταχύτατη απόκριση του συστήματος, μια απαίτηση που τα συστήματα διαχείρισης ρευμάτων δεδομένων μπορούν να ικανοποιήσουν. Επίσης η δυσκολία αποθήκευσης του μεγάλου όγκου δεδομένων που συσσωρεύεται από τα ρεύματα εισόδου, επιτάσσει την εφαρμογή τεχνικών προσεγγίσεως στα δεδομένα.

Η πρακτική αυτή βέβαια συνάδει και με την φύση των εφαρμογών παρακολούθησης, όπου συνήθως οι πρόσφατες καταγραφόμενες σημειακές θέσεις είναι αυτές που έχουν αξία ενώ τα παλιότερα στοιχεία θεωρούνται παρωχημένα.

Ερωτήματα σε κινούμενα αντικείμενα

Η χρονική παράμετρος που εισάγεται στα συστήματα παρακολούθησης αντικειμένων, διαφοροποιεί λίγο την έννοια των ερωτημάτων διαρκείας σε σχέση με αυτά των ΣΔΡΔ. Πρωτεύοντα ρόλο στην σημασιολογία των ερωτημάτων παίζουν η χρονική στιγμή υποβολή τους, ο προσδιορισμός της διάρκειας εκτέλεσής τους καθώς και τυχόν αναφορικότητά τους ως προς συγκεκριμένα χρονόσημα. Τα ερωτήματα μπορεί να είναι *στιγμιαία* (instaneous queries), δηλαδή ερωτήματα διαρκείας που υπολογίζονται για κάθε χρονόσημο ξεκινώντας από την χρονική στιγμή υποβολής τους t και αφεξής. Η απάντηση επιστρέφεται μετά την επεξεργασία για κάθε χρονόσημο. Εφόσον δεν τίθεται κάποιος χρονικός περιορισμός ισχύος του ερωτήματος (λ.χ. τα επόμενα 60 λεπτά), το μέγεθος της απάντησης μπορεί θεωρητικά να γίνει άπειρο.

Υπάρχουν επίσης τα *εξακολουθητικά ερωτήματα* (persistent queries), των οποίων η υποβολή κατά τη χρονική στιγμή t ισοδυναμεί με την εκτέλεση μιας ακολουθίας στιγμιαίων ερωτημάτων καθ' όλη την ιστορική εξέλιξη της τροχιάς, τηρώντας για όλα ως κοινή χρονική αφετηρία το t. Καθώς νέες πλειάδες καταφθάνουν στο σύστημα τα αποτελέσματα που λαμβάνονται ανά χρονόσημο θα διαφέρουν.

Λόγω της σημειακής θεώρησης των κινούμενων αντικειμένων τα χωρικά ερωτήματα διαφοροποιούνται σε σχέση με αυτά των χωρικών βάσεων. Τα ερωτήματα μπορούν να διακριθούν σε *ερωτήματα θέσης* (λοςατιον-βασεδ χυεριες), αν ενδιαφέρει η σημειακή θέση του αντικειμένου σε συγκεκριμένες χρονικές στιγμές και σε ερωτήματα τροχιάς (trajectory-based) που εξετάζουν χαρακτηριστικά της τροχιάς του αντικειμένου.

Μερικά ερωτήματα θέσης είναι τα εξής :

- *Ερωτήματα περιοχής* (range queries) που επιστρέφουν τα αντικείμενα που κινούνται εντός μιας δοθείσης περιοχής , λ.χ. τα οχήματα που βρέθηκαν στο ιστορικό κέντρο της Αθήνας την τελευταία ώρα.

- *Ερωτήματα εγγύτερου γείτονα* (nearest neighbor queries) όπου δοθέντος ενός στατικού ή κινούμενου αντικειμένου, επιστρέφουν τα k εγγύτερα αντικείμενα σε αυτό. Η απάντηση μπορεί να αναφέρεται είτε σε μια χρονική στιγμή είτε σε ένα διάστημα τιμών.

- *Ερωτήματα πυκνότητας* (density queries) όπου δεδομένης μιας τιμής κατωφλίου για την πυκνότητα των αντικειμένων και ενός χρονικού παραθύρου, επιστρέφουν τις ανεξάρτητες περιοχές του χώρου, εντός των οποίων η πυκνότητα των κινούμενων αντικειμένων ξεπερνάει τη δοσμένη ανώτατη τιμή κατά τη διάρκεια του χρονικού παραθύρου.

- *Ερωτήματα χρονικών τεμαχίων* (time-sliced queries) που επιστρέφουν τις θέσεις του συνόλου των αντικειμένων κατά τη διάρκεια ενός χρονικού διαστήματος ή μιας χρονικής στιγμής.

Τα ερωτήματα τροχιάς περιλαμβάνουν τα εξής :

- Τοπολογικά ερωτήματα που εξετάζουν τις χωροχρονικές σχέσεις της τροχιάς αντικειμένων με άλλα στατικά ή κινούμενα αντικείμενα ή με περιοχές του χώρου. Συνηθισμένα χωρικά κατηγορήματα που χρησιμοποιούνται είναι τα: *εντός* (inside), *επικαλύπτει* (overlap), *καλύπτεται* (covered by), *καλύπτει* (covers), *περιλαμβάνει* (contains), *συναντά* (meet), *ισούται* (equal) και *χωρίς συνάφεια* (disjoint). Επιπρόσθετα χρησιμοποιούνται και τα εξής σύνθετα κατηγορήματα που αναφέρονται στην τροχιά του αντικειμένου : *εισέρχεται* (enter), *εξέρχεται* (leave), *διασχίζει* (cross) και *παρακάμπτει* (bypass).

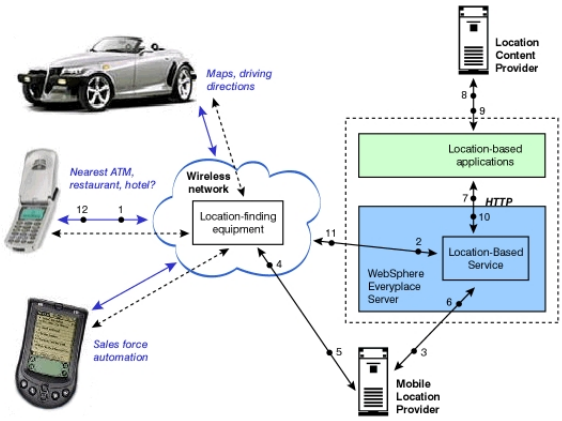
- Ερωτήματα πλοήγησης τα οποία χρησιμοποιούν παραγόμενα μεγέθη της τροχιάς, όπως η διανυθείσα απόσταση και ο συνολικός χρόνος κίνησης για να υπολογίσουν δευτερογενείς πληροφορίες που σχετίζονται με την τροχιά, όπως η ταχύτητα κίνησης και ο προσανατολισμός κίνησης.

Πρέπει να σημειωθεί ότι εφόσον το σύστημα επεξεργάζεται την πληροφορία του ρεύματος και παράγει απάντηση στα ερωτήματα του χρήστη για κάθε χρονόσημο, η απάντηση μπορεί να λάβει την μορφή ενός ρεύματος δεδομένων. Μια τέτοια προσέγγιση όπου τόσο η είσοδος όσο και η έξοδος αποτελούν ρεύματα πληροφορίας, άγνωστου μήκους καθιστά δύσκολο τον χειρισμό και την αποθήκευσή τους. Η αποκλειστική χρήση της κύριας μνήμης είναι απαραίτητη μιας και η αποθήκευση και ανάκληση πληροφοριών από τον δίσκο επιφερει ανεπιθύμητες καθυστερήσεις. Τα κλασσικά συστήματα βάσεων δεδομένων αδυνατούν να ανταπεξέλθουν στις απαιτήσεις αυτές. Αντίθετα τα συστήματα διαχείρισης ρευμάτων δεδομένων πληρούν τις προδιαγραφές αυτές. Πολλές από τις τεχνικές των ΣΔΡΔ μπορούν να εφαρμοστούν για την αποτελεσματική διεκπεραίωση των ερωτημάτων, όπως τα σημεία στίξης, οι διάφορες τεχνικές προσέγγισης (λ.χ. αποβολή φόρτου, κυματίδια) και η αποφυγή χρήσης της δευτερεύουσας μνήμης. Οι τεχνικές αυτές συμβάλλουν στην ομαλή απόκριση του συστήματος που αποτελεί και τον σημαντικότερο στόχο κατά την σχεδίαση.

Υπηρεσίες εντοπισμού

Τα τελευταία χρόνια υπάρχει και ολοένα μεγαλύτερη ζήτηση για εφαρμογές που προσφέρουν *υπηρεσίες εντοπισμού* (location-based services). Η μεγάλη διάδοση των συσκευών δικτύωσης (λ.χ. δέκτες GPS, κινητά τηλέφωνα) έκανε δυνατή την παροχή τέτοιων υπηρεσιών σε χαμηλό κόστος και με πολύ αξιόπιστα αποτελέσματα. Τα διάφορα συστήματα που έχουν αναπτυχθεί προσφέρουν υπηρεσίες στις οποίες ο χρήστης μπορεί να εγγραφεί και να αποκτήσει πρόσβαση στις πληροφορίες που τον ενδιαφέρουν. Χαρακτηριστικά παραδείγματα τέτοιων υπηρεσιών είναι η αναζήτηση των πλησιέστερων σημείων ενδιαφέροντος (λ.χ. πρατηρίων καυσίμων) για έναν οδηγό, το πλησιέστερο μηχάνη αυτόματης ανάληψης ΑΤΜ και η κυκλοφοριακή συμφόρηση που επικρατεί σε μια επιλεγμένη περιοχή του οδικού δικτύου.

Οι υπηρεσίες αυτές αναλαμβάνουν να χειριστούν έναν πολύ μεγάλο όγκο δεδομένων, αποτελούμενο τόσο από στατικά όσο και από κινούμενα αντικείμενα,



Σχήμα 2.1: Υπηρεσίες εντοπισμού

που καταφθάνουν με την μορφή ρεύματος. Οι τεχνικές που αναλύθηκαν προηγουμένως, χρησιμοποιούνται για να διασφαλίσουν την άμεση διεκπεραίωση των ερωτημάτων των συνδρομητών. Στις υπηρεσίες εντοπισμού το κύριο ζητούμενο είναι η άμεση απόκριση στα ερωτήμα, ενώ τα δεδομένα προηγούμενων χρονικών στιγμών θεωρούνται παρωχημένα και άρα δεν χρειάζεται να αποθηκεύονται.

Κεφάλαιο 3

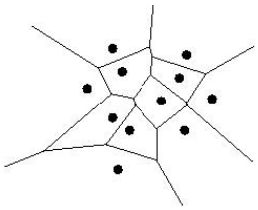
Διαγράμματα Voronoi

Εισαγωγή

Ο τομέας της *υπολογιστικής γεωμετρίας* (computational geometry) ασχολείται με τον σχεδιασμό και την υλοποίηση αλγορίθμων για την επίλυση γεωμετρικών προβλημάτων. Τα τελευταία χρόνια γίνεται όλο και πιο συχνό το φαινόμενο, άλλοι τομείς της επιστήμης να καταφεύγουν στην υπολογιστική γεωμετρία για την επίλυση προβλημάτων του κλάδου τους. Για παράδειγμα τομείς της πληροφορικής, όπως η ρομποτική, η γραφική με υπολογιστές αλλά και η αναγνώριση προτύπων, ωφελήθηκαν από γεωμετρικές λύσεις που μπόρεσε να τους προσφέρει η υπολογιστική γεωμετρία. Ακριβώς λόγω αυτού του ενδιαφέροντος, την τελευταία δεκαετία η υπολογιστική γεωμετρία γνώρισε μεγάλη ανάπτυξη με συνέπεια σήμερα να θεωρείται τομέας που μπορεί να προσφέρει αξιόπιστες λύσεις σε ποικίλα προβλήματα.

Το διάγραμμα *Voronoi* (σχήμα 3.1) αποτελεί μια γεωμετρική λύση στο πρόβλημα της ανεύρεσης του εγγύτερου γείτονα. Είναι ένα πολύ δημοφιλές εργαλείο, αφού μπορεί να δώσει απαντήσεις με ακρίβεια, ευκολία και ταχύτητα. Ένα διάγραμμα *Voronoi* για τα καταστήματα μιας αλυσίδας πάνω σε έναν χάρτη, μπορεί να δείξει με σαφήνεια τις περιοχές επιρροής καθενός. Επίσης σε μια μελέτη τοποθέτησης ποτιστικών μηχανημάτων σε αγροτικές καλλιέργειες παρέχει σαφές ζώνες κάλυψης για κάθε εγκατάσταση. Συχνή είναι η χρήση του και στον χώρο της κινητής τηλεφωνίας, όπου οι κυψέλες κάθε κεραίας μπορούν να αντιστοιχηθούν σε ένα κελί του διαγράμματος. Η χρήση του βέβαια έχει επεκταθεί και σε προβλήματα διαφορετικά από αυτό του εγγύτερου γείτονα, με παρόμοια επιτυχία.

Στο κεφάλαιο αυτό γίνεται μια επισκόπηση των διαγραμμάτων και των ιδιοτήτων τους. Σκοπός είναι να χρησιμοποιηθεί το διάγραμμα *Voronoi* ακίνητων σημείων, για την απάντηση ερωτημάτων εγγύτερου γείτονα σε ένα ρεύμα κινούμενων αντικειμένων. Έτσι μελετήθηκε και υλοποιήθηκε ο αλγόριθμος του *Fortune* που αποτελεί και την βέλτιστη επιλογή. Η εύρεση του εγγύτερου γείτονα για ένα κινούμενο αντικείμενο, αντιστοιχεί στην εύρεση του κελιού του διαγράμματος μέσα στο οποίο βρίσκεται αυτό. Έτσι υλοποιήθηκε ο αλγόριθμος *point in polygon*, που



Σχήμα 3.1: Διάγραμμα Voronoi σημείων στο επίπεδο

μπορεί να δώσει απάντηση στο ερώτημα αυτό, με εφαρμογή του για κάθε κελί του διαγράμματος. Τέλος στο σύστημα έγιναν πειραματικές μετρήσεις για δεδομένο αριθμό ακίνητων σημείων και κινούμενων αντικειμένων.

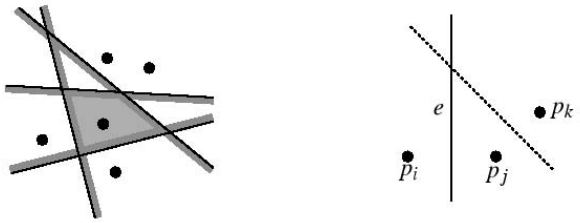
3.1 Θεμελιώδεις έννοιες και ιδιότητες

Έστω $P = \{p_1, p_2, ..., p_n\}$ ένα σύνολο n διακριτών σημείων στο επίπεδο, τα οποία αφεξής θα αποκαλούνται *εστίες* (sites), ενώ ως απόσταση $dist(p, q)$ δύο σημείων p και q στο επίπεδο θεωρείται το μέτρο της Ευκλείδειας νόρμας.

> Διάγραμμα *Voronoi* $Vor(P)$ για το σύνολο P, είναι μια διαμέριση του επιπέδου σε n στοιχειώδη κύτταρα (*ή κελιά - cells*), όπου κάθε κύτταρο $V(p_i)$ αντιστοιχεί σε μια εστία p_i του P και οποιοδήποτε σημείο q του επιπέδου βρίσκεται μέσα σε μία μόνο περιοχή επιρροής κάποιας εστίας (*κελί*).

Πολλές φορές ως διαγράμματα *Voronoi* εννοούμε το σύνολο των ακμών και των κορυφών των αντίστοιχων κυττάρων και όχι το πολύγωνο της επιφάνειάς τους. Ο Καρτέσιος το 1644 μελέτησε για πρώτη φορά τις ιδιότητες των διαγραμμάτων αυτών, ενώ αργότερα έγιναν γνωστά ως *ψηφιδωτά Dirichlet* (tessellations) (1850) ή *πολύγωνα Thiessen*. Το 1907 - 1908 ο Voronoi συστηματοποίησε τη μελέτη των ιδιοτήτων τους και η χρήση τους διαδόθηκε σε πολλές επιστήμες μέσα στον 20ό αιώνα, αξιοποιώντας την έννοια της εγγύτητας προς συγκεκριμένα σημεία αναφοράς. Στο πεδίο των βάσεων δεδομένων, τα σημεία που εμπίπτουν εντός του κυττάρου γύρω από την εστία p αναφέρονται ως *σύνολο επιρροής* (influence set) του p.

Για δύο σημεία p και q στο επίπεδο, η μεσοκάθετος του τμήματος $\overline{pq}$ διαμερίζει το επίπεδο σε δύο ημιεπίπεδα : το ημιεπίπεδο $h(p, q)$ που περιλαμβάνει το p και το ημιεπίπεδο $h(q, p)$ όπου ανήκει το q. Κάθε σημεία r του επιπέδου ανήκει στο $h(p, q)$ αν και μόνο αν $dist(r, p) < dist(r, q)$. Συνεπώς, κάθε κελί *Voronoi* είναι η τομή των $n-1$ ημιεπιπέδων που σχηματίζουν οι μεσοκάθετοι με όλες τις υπόλοιπες (*Σχήμα 3.2(a)*). Επειδή ορισμένα κελιά μπορεί να μην είναι κλειστά (αντιστοιχούν σε ακραία σημεία στο P), κάθε κελί περικλείεται από $n-1$ ακμές και $n-1$ κορυφές το πολύ.



Σχήμα 3.2: (α) Κάθε κελί Voronoi είναι τομή ημιεπιπέδων. (β) Τομή μεσοκαθέτων για μη συνευθειακές εστίες.

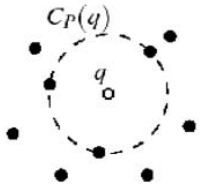
Πρακτικά οι κορυφές κάθε κελιού *Voronoi* $V(p_i)$ προκύπτουν ως τομές των μεσοκαθέτων των ευθυγράμμων τμημάτων, που ενώνουν την εστία p_i με τις γειτονικές της (*Σχήμα 3.2(β)*). Ορισμένες ακμές είναι ευθύγραμμα τμήματα, ενώ κάποιες μπορεί να είναι ημιευθείες (ανοιχτά κελιά). Αποδεικνύεται ότι:

Λήμμα 1. *Καμία ακμή του διαγράμματος δεν θα είναι πλήρης ευθεία, παρά μόνον αν όλες οι εστίες είναι συνευθειακά σημεία.*

Η πολυπλοκότητα της δομής ενός διαγράμματος *Voronoi* είναι αναμενόμενο ότι θα είναι το πολύ $O(n^2)$, αφού υπάρχουν n εστίες και κάθε κελί έχει το πολύ $n-1$ κορυφές και ακμές. Μπορεί να συμβεί κάποιο κελί να έχει γραμμική πολυπλοκότητα, αυτό όμως δεν ισχύει στη γενική περίπτωση. Αποδεικνύεται όμως ότι:

Λήμμα 2. *Για $n \geq 3$, το πλήθος των κορυφών του διαγράμματος Voronoi ενός συνόλου n εστιακών σημείων στο επίπεδο, είναι το πολύ $2n-5$, ενώ ο αριθμός των αντίστοιχων ακμών είναι το πολύ $3n-6$.*

Αν και υπάρχουν $O(n^2)$ μεσοκάθετοι, υπάρχουν κάποιες που μπορεί να μην οδηγούν σε ακμές και επομένως η πολυπλοκότητα του διαγράμματος *Voronoi* είναι $O(n)$. Έτσι, δεν είναι αναγκαίο όλες οι τομές μεταξύ μεσοκαθέτων να αντιστοιχούν σε κορυφές του τελικού διαγράμματος. Εισάγουμε λοιπόν την έννοια του μέγιστου άδειου κύκλου για τον χαρακτηρισμό των ακμών και των κορυφών του διαγράμματος *Voronoi*. Για ένα σημείο q ο μέγιστος άδειος κύκλος $CP(q)$ ως



Σχήμα 3.3: Μέγιστος άδειος κύκλος γύρω από το σημείο q

προς το σύνολο P των εστιών, είναι ο μεγαλύτερος δυνατός κύκλος με κέντρο το σημείο q και ο οποίος δεν περικλείει καμία εστία του P (*Σχήμα 3.3*). Χρησιμοποιώντας τώρα την έννοια του μέγιστου άδειου κύκλου, μπορούμε να εξάγουμε τα εξής δύο συμπεράσματα:

- Ένα σημείο q είναι κορυφή του $Vor(P)$, αν και μόνο αν ο μέγιστος άδειος κύκλος του $CP(q)$ περιλαμβάνει τρεις ή περισσότερες εστίες στην περιφέρειά του.
- Η μεσοκάθετος μεταξύ δύο εστιών p_i και p_j ορίζει μια ακμή στο διάγραμμα $Vor(P)$, αν και μόνο αν υπάρχει ένα σημείο q στη μεσοκάθετο τέτοιο ώστε η περιφέρεια του κύκλου $CP(q)$ να περιλαμβάνει τα σημεία p_i και p_j και καμία άλλη εστία.

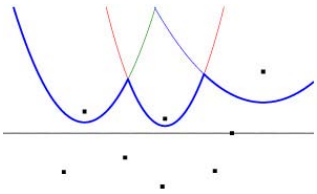
Η μεγάλη δημοτικότητα των διαγραμμάτων *Voronoi*, οφείλεται κατά μεγάλο βαθμό σε δύο λόγους. Πρώτον, στο γεγονός ότι το γραμμικό κόστος του υποδηλώνει ότι το διάγραμμα δεν επιφέρει κατά πολύ μεγαλύτερη πολυπλοκότητα, από αυτήν του υποφαινόμενου συνόλου εστιών. Δεύτερον, ένα διάγραμμα *Voronoi* παρέχει εγγενώς και σαφώς πληροφορία για την γειτνίαση όλων των στοιχείων του συνόλου εστιών, με τρόπο που μπορεί να αξιοποιηθεί πολύ εύκολα προγραμματιστικά, λ.χ. σε προβλήματα όπως αυτό του πλανόδιου πωλητή.

3.2 Αλγόριθμοι κατασκευής διαγράμματος Voronoi

Εφόσον κάθε κελί *Voronoi* προκύπτει ως τομή των ημιεπιπέδων, ο υπολογισμός αυτός είναι εφικτός σε χρόνο $O(n \log n)$ ανά εστία, οπότε ο συνολικός χρόνος ανέρχεται σε $O(n^2 \log n)$, παρόλο που η πολυπλοκότητα του τελικού διαγράμματος είναι γραμμική. Ωστόσο, ο αλγόριθμος του Fortune [For87] που χρησιμοποιήθηκε για την κατασκευή του διαγράμματος *Voronoi* και παρουσιάζεται στην συνέχεια, επιτυγχάνει χρόνο $O(n \log n)$ συνολικά.

Ο αλγόριθμος αυτός είναι βέλτιστος, αφού το πρόβλημα της ταξινόμησης n πραγματικών αριθμών, μπορεί να αναχθεί στην κατασκευή του διαγράμματος *Voronoi*. Κάθε αριθμός απεικονίζεται σε ένα σημείο της ευθείας των πραγματικών αριθμών, οπότε όλα τα σημεία προκύπτουν συνευθειακά. Επομένως το αντίστοιχο διάγραμμα *Voronoi* είναι μια αλυσίδα από παράλληλες ζώνες, οι οποίες σχηματίζουν μια διπλοσυνδεδεμένη λίστα, ταξινομημένη από αριστερά προς τα δεξιά. Συνεπώς ο αλγόριθμος θα χρειαστεί χρόνο $\Omega(n \log n)$ στην χειρότερη περίπτωση.

Ένας άλλος αλγόριθμος που στηρίζεται στην τεχνική "διαίρει και βασίλευε' επινοήθηκε από τους Shamos και Hoey [SH75]. Αρχικά διαιρεί τα n σημεία σε δύο ισάριθμα σύνολα, με βάση την τετμημένη τους, οπότε το διάγραμμα *Voronoi* πρέπει να υπολογιστεί για κάθε υποσύνολο κ.ο.κ. Ωστόσο, μετά από κάθε στάδιο, θα πρέπει να συρραφούν τα δύο υποδιαγράμματα που προκύπτουν και αυτό μπορεί να γίνει σε γραμμικό χρόνο. Η συρραφή γίνεται κατά μήκος μιας τεθλασμένης y-μονότονης γραμμής η οποία συγκροτείται από τμήματα μεσοκαθέτων που θα



Σχήμα 3.4: Η παραλιακή γραμμή

συμπεριληφθούν στο τελικό διάγραμμα. Η αρχή και το τέλος της γραμμής συρραφής είναι ημιευθείες που σχηματίζονται από τις μεσοκαθέτους των 'γεφυρών' που ορίζονται μεταξύ των δύο υποσυνόλων. Για κάθε κελί του διαγράμματος ο χρόνος που χρειάζεται είναι $O(n \log n)$ και άρα ο συνολικός χρόνος για τον υπολογισμό ολόκληρου του διαγράμματος *Voronoi* είναι $O(n^2 \log n)$.

3.3 Ο αλγόριθμος του Fortune

Όπως αναφέρθηκε, για την κατασκευή του διαγράμματος χρησιμοποιήθηκε ο αλγόριθμος του Fortune, ο οποίος είναι βέλτιστος και με κόστος $O(n \log n)$. Στηρίζεται στην τεχνική της *γραμμής σάρωσης* (sweep line), χρησιμοποιώντας την για την σταδιακή παραγωγή του διαγράμματος. Χρησιμοποιώντας μια οριζόντια γραμμή ℓ (παράλληλη προς τον άξονα x) ως γραμμή σάρωσης, ο αλγόριθμός σαρώνει το επίπεδο από πάνω προς τα κάτω, ενημερώνοντας σταδιακά τη δομή δεδομένων όπου τηρείται το διάγραμμα *Voronoi*. Συνήθως όταν χρησιμοποιείται η τεχνική της γραμμής σάρωσης, ενημερώνουμε το αποθηκευμένο αποτέλεσμα με την τομή της γραμμής σάρωσης με την δομή που μελετάμε. Στην συγκεκριμένη περίπτωση όμως, το τμήμα του διαγράμματος *Voronoi* που βρίσκεται πάνω από την γραμμή σάρωσης δεν είναι εξ ολοκλήρου γνωστό. Η ιδιαιτερότητα αυτή έγκειται στο γεγονός ότι η γραμμή σάρωσης φτάνει σε μια κορυφή του κελιού $V(p_i)$ πριν συναντήσει την αντίστοιχη εστία p_i, με συνέπεια να μην είναι διαθέσιμη όλη η πληροφορία που απαιτείται για τον υπολογισμό της κορυφής.

Γι' αυτό το λόγο ο αλγόριθμος τροποποιεί την χρήση της γραμμής σάρωσης με τρόπο τέτοιο, ώστε να τηρείται το τμήμα του διαγράμματος *Voronoi* πάνω από την γραμμή ℓ, το οποίο δεν πρόκειται να αλλάξει από τα σημεία που βρίσκονται κάτω από την ℓ. Η ιδιότητα των σημείων που αποτελούν το αμετάβλητο τμήμα του διαγράμματος είναι ότι βρίσκονται εγγύτερα σε κάποια ήδη γνωστή εστία (πάνω από την ℓ) απ' ό,τι στην γραμμή ℓ, όταν η γραμμή σάρωσης συναντήσει νέες εστίες. Μαθηματικά ο γεωμετρικός τόπος αυτών των σημείων, για κάθε γνωστή εστία, περικλείεται από μια παραβολή, η οποία ορίζεται από την συγκεκριμένη εστία και έχει διευθετούσα την γραμμή ℓ. Αν φέρουμε τις παραβολές για όλες τις γνωστές εστίες τα χαμηλότερα τόξα που σχηματίζονται από την τομή τους αποτελούν μια ακολουθία τόξων που ονομάζεται *παραλιακή γραμμή* (beach line) (*Σχήμα 3.4*).

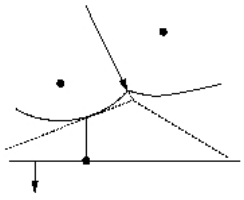
Η παραλιακή γραμμή αποτελεί το σύνορο του αμετάβλητου τμήματος του δια-

γράμματος *Voronoi* σε κάθε βήμα του αλγόριθμου. Τα σημεία καμπής των παραβολών, εφόσον ισαπέχουν από τις αντίστοιχες εστίες, βρίσκονται πάνω στις ακμές του διαγράμματος και ουσιαστικά με την κίνηση της γραμμής σάρωσης ℓ προς μικρότερα y, παρακολουθούν την δημιουργία τους. Αποδεικνύεται ότι κάθε σημείο τομής τριών παραβολών είναι κορυφή του διαγράμματος, εφόσον αυτό αποτελεί ταυτόχρονα και σημείο τομής τριών ακμών του. Κατά τον σχηματισμό της παραλιακής γραμμής από την τομή των παραβολών, είναι δυνατό περισσότερα του ενός τόξα μιας παραβολής να λαμβάνουν μέρος σε αυτήν. Επειδή όμως η παραλιακή γραμμή είναι μονότονη ως προς x, αφού διέρχεται από το χαμηλότερο σημείο όλων των παραβολών, οποιαδήποτε κατακόρυφη γραμμή την τέμνει σε ένα και μόνο σημείο. Χρησιμοποιώντας λοιπόν την έννοια της παραλιακής γραμμής, ο αλγόριθμος δεν χρειάζεται να τηρεί πλέον τις τομές της γραμμής σάρωσης ℓ με το διάγραμμα *Voronoi*, παρά μόνο να ενημερώνει την κατάσταση της παραλιακής γραμμής σε διακριτά βήματα, τα οποία οριοθετούνται από ένα γεγονός κύκλου ή ένα γεγονός εστίας.

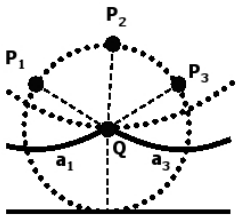
3.3.1 Γεγονός εστίας (site event)

Το *γεγονός εστίας* συμβαίνει όταν η γραμμή σάρωσης συναντήσει μια εστία p_i και άρα ένα νέο παραβολικό τόξο πρέπει να ενταχθεί στην παραλιακή γραμμή (σχήμα 3.5). Εφόσον η παραβολή που θα αντιστοιχεί στην καινούργια εστία, την στιγμή της δημιουργίας της έχει διευθετούσα με ίδια y συντεταγμένη, ξεκινάει ως μια εκφυλισμένη παραβολή με μηδενικό εύρος. Αποτελεί ουσιαστικά μια κάθετη ημιευθεία στην γραμμή ℓ, η οποία όμως διευρύνεται, μεταβάλλοντας την παραλιακή γραμμή καθώς η γραμμή σάρωσης μετακινείται προς χαμηλότερα y. Τα νέα σημεία καμπής του τόξου στην παραλιακή γραμμή παρακολουθούν τις δύο νέες ακμές του διαγράμματος, οι οποίες στην αρχή είναι αποκομμένες. Σε επόμενο βήμα του αλγόριθμου όμως, αυτές θα συναντήσουν κάποιες άλλες ακμές, σχηματισμένες με παρόμοιο τρόπο, ώστε διαδοχικά να συμπληρωθεί ολόκληρο το διάγραμμα *Voronoi*. Αποδεικνύεται ότι:

Λήμμα 3. *Ο μόνος τρόπος για να ενταχθεί ένα νέο τόξο στην παραλιακή γραμμή, είναι όταν συμβαίνει ένα γεγονός εστίας.*



Σχήμα 3.5: Γεγονός εστίας



Σχήμα 3.6: Γεγονός κύκλου

Η παραλιακή γραμμή αποτελείται από το πολύ $2n-1$ τόξα, εφόσον κάθε εστία δημιουργεί μόνο μια παραβολική γραμμή, κάθε τόξο μπορεί να διασπαστεί το πολύ μια φορά και δεν υπάρχει άλλος τρόπος ένταξης ενός τόξου στην παραλιακή γραμμή.

3.3.2 Γεγονός κύκλου (circle event)

Ως *γεγονός κύκλου* ονομάζουμε την κατάσταση όπου η γραμμή σάρωσης ℓ αγγίζει το χαμηλότερο σημείο ενός κύκλου, ο οποίος ορίζεται από τρεις εστίες με διαδοχικά τμήματα στην παραλιακή γραμμή (σχήμα 3.6). Τότε το μεσαίο από τα τρία τόξα απαλείφεται, αφού τα δύο σημεία καμπής που το ορίζουν πλέον ταυτίζονται, οπότε σε εκείνο το σημείο τοποθετείται μια κορυφή του διαγράμματος *Voronoi*. Το σημείο καμπής που προέκυψε από την ταύτιση των δύο προηγούμενων, παρακολουθεί πλέον μια ακμή του διαγράμματος που το ένα άκρο της είναι η προαναφερθείσα κορυφή. Αποδεικνύεται ότι :

Λήμμα 4. *Ένα υπάρχον τόξο μπορεί να απαλειφθεί από την παραλιακή γραμμή, μόνο με την εμφάνιση ενός γεγονότος κύκλου.*

3.3.3 Δομές δεδομένων

Οι δομές δεδομένων που χρησιμοποιούνται από τον αλγόριθμο είναι οι εξής :

- Το διάγραμμα *Voronoi* τηρείται ως μια διπλοσυνδεδεμένη λίστα ακμών D, δομή που ενδείκνυται για την αποθήκευση υποδιαιρέσεων του επιπέδου. Για τις ακμές του διαγράμματος που δεν είναι ευθύγραμμα τμήματα αλλά ημιευθείες, γίνεται η υπόθεση ότι υπάρχει ένα περιβάλλον παραλληλόγραμμο γύρω από τη σκηνή, το οποίο περικλείει όλες τις εστίες.

- Μια ουρά γεγονότων Q υλοποιημένη ως ουρά προτεραιότητας, όπου φυλάσσονται όλα τα επικείμενα γεγονότα εστίας και κύκλου, ταξινομημένα ως προς την y-συντεταγμένη τους. Κάθε κόμβος περιλαμβάνει ένα δείκτη στον κόμβο του δέντρου T, ο οποίος περιγράφει το γεγονός.

- Σε ένα ισοζυγισμένο δυαδικό δένδρου αναζήτησης T τηρείται η κατάσταση της παραλιακής γραμμής. Σε αυτό δεν αποθηκεύονται γραμμές αλλά σημεία. Στα φύλλα του δένδρου αποθηκεύεται η κατάσταση της παραλιακής γραμμής, αφού σε κάθε ένα αποθηκεύεται το εστιακό σημείο p_i που ορίζει ένα τόξο της, από αριστερά προς τα δεξιά. Αντίθετα, κάθε εσωτερικός κόμβος αντιπροσωπεύει το σημείο καμπής των τόξων που αντιστοιχούν στα δύο παιδιά του. Με αυτό τον τρόπο είναι δυνατόν να προσδιορίζεται με λογαριθμικό κόστος ($O(\log n)$) το τόξο που βρίσκεται ακριβώς πάνω από μια νέα εστία, συγκρίνοντας την x-συντεταγμένη του σημείου καμπής σε κάθε κόμβο με αυτήν της νέας εστίας. Το σημείο καμπής ευρίσκεται σε σταθερό χρόνο από

τις δύο εστίες που το ορίζουν και τη γραμμή σάρωσης ℓ. Σε κάθε φύλλο του δένδρου υπάρχει ένα δείκτης (pointer) προς το αντίστοιχο γεγονός κύκλου της ουράς, το οποίο σηματοδοτεί εξαφάνιση του αντίστοιχου τόξου. Ο δείκτης, όταν το γεγονός δεν υπάρχει ή δεν έχει ανιχνευθεί ακόμη, παίρνει την τιμή *νιλ*. Αντίστοιχα στους εσωτερικούς κόμβους του T, υπάρχει ένας δείκτης που δείχνει στην ακμή της λίστας D, την οποία παρακολουθεί το σημείο καμπής που περιγράφει ο κόμβος.

Ο αλγόριθμος προϋποθέτει ότι όλα τα γεγονότα εστίας είναι γνωστά εξαρχής, ενώ τα γεγονότα κύκλου ανιχνεύονται κατά την εξέλιξή του. Σε κάθε στάδιο της σάρωσης, ο αλγόριθμος χειρίζεται ένα γεγονός από την ουρά και η τοπολογική δομή της παραλιακής γραμμής αλλάζει, εφόσον κάποια τριάδα τόξων εξαφανίζεται. Είναι απαραίτητο να διασφαλιστεί ότι το πιθανό γεγονός κύκλου που αφορά μια τέτοια τριάδα έχει όντως αποθηκευτεί στην ουρά Q. Αν τα δύο σημεία καμπής που αντιστοιχούν σε μια τριάδα διαδοχικών τόξων αποκλίνουν καθώς προχωρά η γραμμής σάρωσης ℓ, τότε δεν ορίζεται γεγονός κύκλου.

Ακόμη όμως και στην περίπτωση που τα σημεία καμπής συγκλίνουν, δεν είναι απαραίτητο ότι το γεγονός κύκλου όντως θα συμβεί. Είναι δυνατό η τριάδα των τόξων να εξαφανιστεί πριν τον χειρισμό του γεγονότος. Για παράδειγμα μπορεί να ανιχνευθεί μια καινούργια εστία από την γραμμή σάρωσης που θα δημιουργήσει ένα νέο εμβόλιμο παραβολικό τόξο στην τριάδα. Το γεγονός αυτό αποκαλείται *άκυρος υποψήφιος* (false alarm). Όταν ένα τόξο εξαφανίζεται, ελέγχονται όλες οι τριάδες συνεχόμενων τόξων όπου συμμετέχει. Αν στην ουρά Q υπάρχει ένα γεγονός κύκλου με δείκτη σε κάποια από αυτές, τότε το γεγονός διαγράφεται από την ουρά ως άκυρος υποψήφιος. Αποδεικνύεται ότι:

Λήμμα 5. *Κάθε κορυφή του διαγράμματος Voronoi ανιχνεύεται με ένα γεγονός κύκλου.*

Ο αλγόριθμος κατασκευής του διαγράμματος *Voronoi* έχει συνοπτικά ως εξής:

ΑΛΓΟΡΙΘΜΟΣ ΤΟΥ FORTUNE

1. Αρχικοποίησε :
   (α′) την ουρά γεγονότων Q με όλα τα γεγονότα εστίας.
   (β′) ένα κενό δέντρο T για την παραλιακή γραμμή.
   (γ′) μια κενή διπλοσυνδεδεμένη λίστα ακμών D.
2. Ενόσω η ουρά Q δεν είναι άδεια:
   (α′) Πάρε το γεγονός με την μεγαλύτερη y-συντεταγμένη.
   (β′) Αν πρόκειται για γεγονός της εστίας p_i τότε ΧΕΙΡΙΣΜΟΣ ΓΕΓΟΝΟΤΟΣ ΕΣΤΙΑΣ(p_i), αλλιώς ΧΕΙΡΙΣΜΟΣ ΓΕΓΟΝΟΤΟΣ ΚΥΚΛΟΥ(g), όπου g το φύλλο του δέντρου T που αντιπροσωπεύει το τόξο που θα εξαφανιστεί.

3. Οι εσωτερικοί κόμβοι που έχουν απομείνει στο δέντρο T αντιστοιχούν σε ημιευθείες του διαγράμματος, οπότε υπολόγισε το περιβάλλον παραλληλόγραμμο των εστιών και αναθεώρησε καταλλήλως την λίστα D.

4. Διάσχισε την διπλοσυνδεδεμένη λίστα ακμών D ώστε να κατασκευαστούν σωστά τα μη πεπερασμένα κελιά του διαγράμματος προς τα κάτω.

ΧΕΙΡΙΣΜΟΣ ΓΕΓΟΝΟΤΟΣ ΚΥΚΛΟΥ (g)

1. Διάγραψε το φύλλο g από το δέντρο T και αναδιάταξε καταλλήλως τους προγόνους του. Διάγραψε τα υποψήφια γεγονότα κύκλου από την ουρά Q όπου εμπλέκεται το g.

2. Ενημέρωσε την διπλοσυνδεδεμένη λίστα ακμών D με τη νέα κορυφή που αντιστοιχεί στο g και τις ακμές που καταλήγουν σ' αυτήν.

3. Έλεγξε τις τριάδες διαδοχικών τόξων που προκύπτουν και έχουν ως μεσαίο τον πρώην αριστερό και τον πρώην δεξιό γείτονα του g. Αν τα σημεία καμπής συγκλίνουν, τότε εισάγαγέ τα στην Q ως υποψήφια γεγονότα κύκλου.

ΧΕΙΡΙΣΜΟΣ ΓΕΓΟΝΟΤΟΣ ΕΣΤΙΑΣ (p_i)

1. Αν το δέντρο T είναι κενό, τότε το σημείο p_i μπορεί να εισαχθεί, αλλιώς συνέχισε με τα παρακάτω βήματα 2 έως 5.

2. Αναζήτησε στο T το παραβολικό τόξο που βρίσκεται κατακόρυφα πάνω από το p_i. Αν το σχετικό φύλλο του δέντρου έχει δείκτη στην ουρά Q, πρόκειται για άκυρο υποψήφιο που πρέπει να διαγραφεί από την Q.

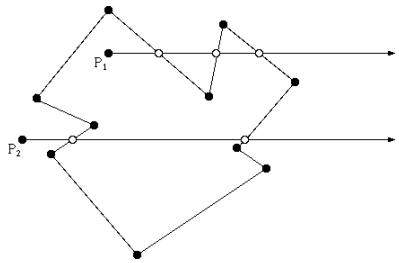
3. Αντικατάστησε το φύλλο του T με ένα υπόδενδρο με τρία φύλλα για την τριάδα των διαδοχικών παραβολικών τόξων, εκ των οποίων το μεσαίο αντιπροσωπεύει την εστία p_i.

4. Δημιούργησε δύο νέες ακμές του διαγράμματος στην διπλοσυνδεδεμένη λίστα D, οι οποίες θα ανιχνεύονται από τα δύο νέα σημεία καμπής της τριάδας τόξων.

5. Έλεγξε τις τριάδες διαδοχικών τόξων που προέκυψαν. Αν τα σημεία καμπής συγκλίνουν, τότε εισάγαγέ τα στην Q ως υποψήφια γεγονότα κύκλου

3.4 Ο αλγόριθμος Point in polygon

Ο έλεγχος αν ένα σημείο p βρίσκεται μέσα σε ένα απλό πολύγωνο P είναι μια αρκετά συνηθισμένη λειτουργία στις χωρικές εφαρμογές. Ο πιο συνηθισμένος αλγόριθμος είναι ο αλγόριθμος *σημείο εντός πολυγώνου* (point in polygon), ο οποίος βασίζεται στην εξής αρχή. Σχεδιάζουμε μια τυχαία ημιευθεία που αρχίζει από το σημείο p και μετράμε τον αριθμό των τομών της με τις πλευρές του πολυγώνου.

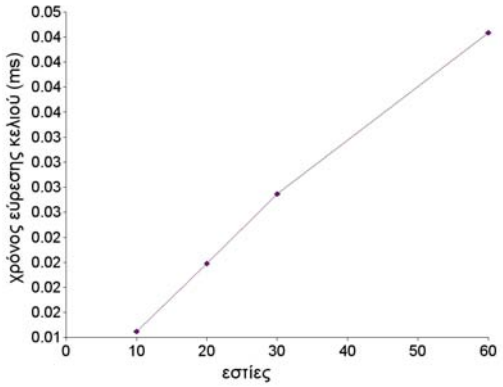


Σχήμα 3.7: Αλγόριθμος point in polygon : το σημείο P_2 είναι εκτός πολυγώνου, ενώ το P_1 εντός.

Αν ο αριθμός αυτός προκύψει άρτιος το p είναι εκτός πολυγώνου, ενώ αν είναι περιττός είναι εντός πολυγώνου. Αναλυτικά ο αλγόριθμος σε ψευδοκώδικα έχει ως εξής:

```
Point in polygon (P: πολύγωνο , p: σημείο)
begin
    if (p είναι πάνω σε μια πλευρά του P)
        then το p είναι μέσα στο P
    else
        count = 0
        l = ημιευθεία από το p
        for( i=1 το n)
        begin
            if (ΤΕΜΝΟΝΤΑΙ(πλευρά(i), l ) and
                not ΣΥΝΕΥΘΕΙΑΚΑ (πλευρά(i), l) ) then
                begin
                    if ( ένα τερματικό σημείο της πλευράς (i) είναι πάνω από το l)
                        then count = count + 1
                end
        end for
        if (count είναι περιττό)
            then το p είναι μέσα στο P
    end if
end
```

Ο έλεγχος αν είναι συνευθειακές η πλευρά του πολυγώνου και η ημιευθεία καθώς και ο έλεγχος για το αν ένα τερματικό σημείο της πλευράς είναι πάνω από την ημιευθεία γίνονται για να αντιμετωπιστούν ειδικές περιπτώσεις. Αν δεν υπήρχαν, θα είχαμε λανθασμένο αποτέλεσμα.



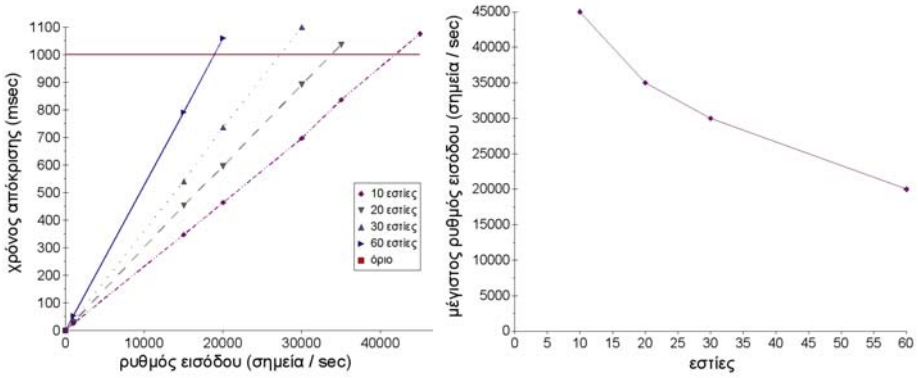
Σχήμα 3.8: Χρόνος απόκρισης ανά αριθμό εστιών

3.5 Πειραματικά αποτελέσματα

Σύμφωνα με την υλοποίηση που περιγράφηκε, δημιουργήθηκε το διάγραμμα *Voronoi* για τέσσερα σύνολα 10,20,30 και 60 εστιών. Σε κάθε ένα από αυτά εφαρμόσθηκε ο αλγόριθμος point in polygon για την εύρεση εγγύτερου γείτονα, για ένα σύνολο 100 κινούμενων αντικειμένων για 1000 διακριτές χρονικές στιγμές. Το σύνολο αυτό περιγράφει την κίνηση αυτοκινήτων στο οδικό δίκτυο της Αθήνας. Κάθε εγγραφή του περιγράφει την ταυτότητα του αντικειμένου *id*, τις συντεταγμένες του και το χρονόσημο αναφοράς. Για τις ανάγκες των πειραμάτων δεν λήφθηκαν υπόψιν τα χρονόσημα, αλλά το σύνολο θεωρήθηκε ως ένα πλήθος 100.000 σημειακών θέσεων. Με είσοδο κάθε μία από αυτές εφαρμόσθηκε ο αλγόριθμος point in polygon. Να σημειωθεί ότι δεν υπήρχε κάποια δομή ευρετηρίου πάνω στο διάγραμμα *Voronoi* και άρα ο αλγόριθμος εκτελούσε τυφλή αναζήτηση του κελιού. Τα πειράματα εκτελέστηκαν σε λειτουργικό σύστημα Ubuntu GNU/Linux 6.06 σε προσωπικό υπολογιστή Intel Celeron M 1600 (1.6 GHz) με μνήμη 266MHz.

Στην πρώτη δοκιμή μετρήθηκαν οι χρόνοι που κάνει ο αλγόριθμος point in polygon για την εύρεση του κελιού του διαγράμματος *Voronoi* στο οποίο βρίσκεται μία σημειακή θέση. Το πείραμα πραγματοποιήθηκε για τα τέσσερα σύνολα εστιών (10, 20, 30, 60), με είσοδο τις 100.000 σημειακές θέσεις. Ο χρόνος απάντησης του αλγόριθμου υπολογίστηκε ως ο μέσος όρος όλων των απαντήσεων. Για τα τέσσερα σύνολα εστιών, όπως φαίνεται και στο *Σχήμα 3.8*, προκύπτει μια γραμμικά αύξουσα σχέση. Αυτό ήταν αναμενόμενο, αφού ο αλγόριθμος είναι τυφλής αναζήτησης και άρα η αύξηση στον χρόνο απόκρισης είναι ανάλογη της αύξησης τους χώρου αναζήτησης.

Ως δεύτερη δοκιμή μετρήθηκε ο αριθμός των κινούμενων αντικειμένων που μπορεί να δεχτεί το σύστημα ανά δευτερόλεπτο, ώστε να μπορεί να ανταπεξέλθει χωρίς καθυστερήσεις. Έτσι δημιουργήθηκαν από το σύνολο των σημειακών θέσεων ομάδες με διαφορετικό αριθμό στοιχείων. Οι ομάδες αυτές τροφοδοτήθηκαν



Σχήμα 3.9: (α) Ρυθμοί εισόδου ανά αριθμό εστιών. (β) Μέγιστος ρυθμός εισόδου ανά αριθμό εστιών.

ως είσοδος στον αλγόριθμο point in polygon και μετρήθηκαν οι χρόνοι απόκρισης για τα τέσσερα διαγράμματα. Σκοπός ήταν να διαπιστωθεί για ποιόν αριθμό σημειακών θέσεων η απόκριση ήταν μεγαλύτερη του ενός δευτερολέπτου. Στο *Σχήμα 3.5(α)* φαίνεται ο χρόνος απόκρισης για μεταβλητό αριθμό εστιών καθώς και το όριο του ενός δευτερολέπτου. Η τομή των ευθειών με αυτό το όριο, δίνει και τον μέγιστο ρυθμό εισόδου που μπορεί να εξυπηρετήσει το σύστημα για κάθε σύνολο εστιών. Στα σημεία αυτά ο ρυθμός εισόδου είναι ίσος με τον ρυθμό εξόδου. Για ομάδες σημείων με μεγαλύτερο αριθμό το σύστημα εισάγει καθυστέρηση. Η διακύμανση του μέγιστου ρυθμού για τα τέσσερα σύνολα εστιών φαίνεται στο *Σχήμα 3.5(β)* και προκύπτει πάλι γραμμική σχέση. Και σε αυτή την περίπτωση αυτό ήταν αναμενόμενο, αφού στην ουσία η αύξηση των εστιών αυξάνει γραμμικά τον χώρο αναζήτησης του αλγόριθμου.

3.6 Ερευνητικά ζητήματα

Τα διαγράμματα *Voronoi* χρησιμοποιούνται ευρέως σε διάφορους τομείς της επιστήμης. Στις χωρικές βάσεις δεδομένων για παράδειγμα οι M. R. Kolahdouzan et al. [KS04] δίνουν δύο διαφορετικές τεχνικές αξιοποίησης των διαγραμμάτων *Voronoi* για την εύρεση των k εγγύτερων γειτόνων για αντικείμενα κινούμενα σε δίκτυα (π.χ. δρόμοι) . Η έρευνα γίνεται με έμφαση στα συστήματα πλοήγησης που χρησιμοποιούν GPS και σε δίκτυα μεταφορών για τον εγγύτερο γείτονα σε υπηρεσίες εντοπισμού.Στο [MDA02] χρησιμοποιούνται τα *Voronoi* για να αναπτυχθεί μια τεχνική που επιτρέπει την ακριβή απόδοση (rendering) τρισδιάστατων αντικειμένων κάτω από οποιεσδήποτε συνθήκες φωτισμού.Στο [Chi00] αναπτύσσεται μια τεχνική υλοποίησης κατανεμημένων υπηρεσιών για μικροκυματικά δίκτυα, αξιοποιώντας την χωρική πληροφορία των διαγραμμάτων *Voronoi*. Οι P.Labute et al. [LS05] βρίσκουν τα σημεία πρόσδεσης σε διαφορετικές δομές πρωτεϊνών αξιοποιώντας διαγράμματα *Voronoi*. Στο [Har95] υλοποιούνται διαγράμματα *Voronoi*

για να προταθεί μια λύση για την αναπαράσταση ποταμών σε συστήματα GIS. Στο [ZZP+03] μελετάται ο υπολογισμός κυττάρων *Voronoi* για κινούμενα αντικείμενα σε γεωμετρικά ρεύματα δεδομένων. Σε περιπτώσεις μεγάλων όγκων δυναμικά μεταβαλλόμενων σημειακών θέσεων δημιουργείται ένα γεωμετρικό ρεύμα δεδομένων (geometric data stream), τα στοιχεία του οποίου πρέπει να χρησιμοποιηθούν για online επεξεργασία. Στην περίπτωση αυτή, αντί για την κατασκευή ενός ολοκληρωμένου διαγράμματος *Voronoi*, επιλέγεται η λύση της ανανέωσης επιλεγμένων κελιών για συγκεκριμένα σημεία ενδιαφέροντος.

Ο F. Aurenhammer [Aur91] αναφέρει πολλές εφαρμογές των διαγραμμάτων *Voronoi*, σε διάφορους τομείς. Στον τομέα της πληροφορικής, δίνει ως παραδείγματα την συσχετιζόμενη αναζήτηση αρχείων (associative file searching), την χρονοδρομολόγηση πρόσβασης εγγραφών (scheduling record accesses) αλλά και την ανίχνευση συγκρούσεων (collision detection). Σε επίπεδο φυσικών επιστημών αναφέρει την μελέτη των περιοχών δράσης στην κρυσταλλογραφία (domains of action), τις ζώνες Wigner-Seitz, αλλά και τα μοντέλα Johnson-Mehl και Apollonius που βρίσκουν εφαρμογή στην χημεία, στην γεωλογία,στην βιολογία αντίστοιχα και την χρήση τους ως πολύγωνα Thiessen στην κλιματολογία.

3.7 Επεκτάσεις των διαγραμμάτων Voronoi

Η αρχική ιδέα των διαγραμμάτων *Voronoi* μπορεί να επεκταθεί, δίνοντας νέου τύπου διαγράμματα ικανά να ανταποκριθούν σε διαφορετικών απαιτήσεων προβλήματα. Η πιο προφανής επέκταση είναι αυτή σε πολυδιάστατους χώρους. Σε ευκλείδειο χώρο d διαστάσεων για n σημεία , η πολυπλοκότητα του διαγράμματος είναι $\Theta(n^{\lceil d/2 \rceil})$ και μπορεί να υπολογιστεί σε ασυμπτωτικά βέλτιστο χρόνο $O(\log n + n^{\lceil d/2 \rceil})$.

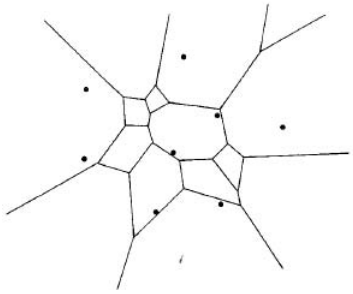
Πολλές παραλλαγές μπορούν να προκύψουν από την επιλογή της ιδιότητας σύμφωνα με την οποία θα κατασκευαστεί το διάγραμμα. Στα διαγράμματα *Voronoi* που μελετήθηκαν, η ιδιότητα αυτή ήταν η ευκλείδεια απόσταση. Αν ορίσουμε την απόσταση δύο σημείων p και q σύμφωνα με τον εξής τύπο:

$$dist_d(p,q) = \sqrt[d]{|p_x - q_x|^d + |p_y - q_y|^d} \tag{3.1}$$

με κατάλληλη επιλογή του d, παράγουμε διαφορετικά διαγράμματα. Για $d = 2$ είναι φανερό ότι παίρνουμε την ευκλείδεια απόσταση, ενώ για $d = 1$ παίρνουμε την απόσταση Manhattan. Αν στον υπολογισμό της απόστασης, συνυπολογίζεται αθροιστικά ή πολλαπλασιαστικά ένας παράγοντας βάρους για κάθε εστία τότε προκύπτουν τα *σταθμισμένα* (weighted) διαγράμματα *Voronoi* [Aur86].

3.7.1 Διαγράμματα Voronoi ανώτερης τάξης

Τα διαγράμματα *Voronoi ανώτερης τάξεως* (higher-order Voronoi diagrams) αποτελούν μια από τις σημαντικότερες γενικεύσεις του απλού διαγράμματος. Η



Σχήμα 3.10: Διάγραμμα Voronoi 2 εγγύτερων γειτόνων

έννοια της τάξης αναφέρεται στον αριθμό των σημείων που αποτελούν γεννήτορες του διαγράμματος. Ενώ στα απλά διαγράμματα *Voronoi* γεννήτορες αποτελούσαν τα σημεία του συνόλου $P = \{p_1, p_2, ..., p_n\}$, στα διαγράμματα ανώτερης τάξης θεωρούμε ότι οι γεννήτορες ανήκουν στο σύνολο όλων των δυνατών υποσυνόλων που αποτελούνται από k στοιχεία του P, δηλαδή του συνόλου $A^{(k)}(P) = \{\{p_{11}, \ldots, p_{1k}\}, \ldots, \{p_{l1}, \ldots, p_{lk}\}\}$, όπου $p_{ij} \in P$ και $l = n!/(k!/(n-k)!)$. Τα διαγράμματα ανώτερης τάξης έχουν μελετηθεί απο πολλούς ερευνητές ανάμεσα στους οποίους και οι Miles (1970), Shamos (1978) και Aurenhammer (1990).

Μπορούμε να διακρίνουμε δύο κατηγορίες διαγραμμάτων ανώτερης τάξης. Τα k-τάξεως διαγράμματα *Voronoi* (order-k Voronoi diagrams) και τα *διατεταγμένα* k-τάξεως διαγράμματα *Voronoi* (ordered order-k Voronoi diagrams). Η διαφορά τους έγκειται στο ότι στα διατεταγμένα διαγράμματα, η διάταξη με βάση την απόσταση των στοιχείων ενός συνόλου γεννήτορα λαμβάνεται υπόψιν κατά την δημιουργία του. Αντίθετα στα μη διατεταγμένα, ο παράγοντας αυτός δεν επηρεάζει το κελί. Στην συνέχεια θα αναφερθούμε μόνο στα μη διατεταγμένα k-τάξεως διαγράμματα *Voronoi*.

3.7.2 Διαγράμματα Voronoi k-τάξεως

Τα διαγράμματα *Voronoi* k-τάξεως (order-k Voronoi diagrams) λέγονται αλλιώς και διαγράμματα *Voronoi* k-*εγγύτερων γειτόνων*. Όπως και τα απλά διαγράμματα, αποτελούν έναν διαχωρισμό του επιπέδου σε πολύγωνα. Έχοντας το σύνολο των εστιών $P = \{p_1, p_2, ..., p_n\}$ στο $\Re^2$, τα πολύγωνα περιλαμβάνουν τα σημεία που βρίσκονται εγγύτερα στις k απο τις n εστίες.

Αν $A^{(k)}(P)$ είναι το σύνολο όλων των δυνατών υποσυνόλων k σημείων από το P έχουμε:

$$A^{(k)}(P) = \{{P_1}^{(k)}, \ldots, {P_i}^{(k)}, \ldots, {P_l}^{(k)}\} \tag{3.2}$$

όπου ${P_i}^{(k)} = \{p_{i1}, \ldots, p_{ik}\}, \quad p_{ij} \in P, \quad l = n!/(k!/(n-k)!)$

με το l να συμβολίζει τον αριθμό των δυνατών υποσυνόλων με k στοιχεία από το P.

Αν $d(p, p_{ij})$ είναι η ευκλείδεια απόσταση από το p στο p_{ij}, τα σημεία του συνόλου $P_i^{(k)}$ είναι οι k εγγύτεροι γείτονες του p, αν και μόνο αν οι αποστάσεις του p από τα $p_{i1}, \ldots, p_{ik}$ είναι μικρότερες ή ίσες από τις αποστάσεις του p από τα υπόλοιπα σημεία (π.χ. σημεία που ανήκουν στο $P \setminus \{p_{i1}, \ldots, p_{ik}\}$). Έτσι το σύνολο $V(P_i^k)$ των σημείων που ανατίθενται στο $\{p_{i1}, \ldots, p_{ik}\}$ γράφεται ως εξής:

$$V(P_i^{(k)}) = \left\{ p \middle| d(p, p_{im}) \leq d(p, p_j), \text{ με } m = \{1, \ldots, k\} \text{ και } p_j \in P \setminus P_i^{(k)} \right\} \quad (3.3)$$

Μπορούμε να δώσουμε έναν ισοδύναμο ορισμό αν θεωρήσουμε ότι ο πιο απομακρυσμένος από τους k γείτονες του p, απέχει λιγότερο από το p σε σύγκριση με τα υπόλοιπα σημεία. Δηλαδή :

$$V(P_i{}^{(k)}) = \left\{ p \middle| \max_{P_h} \left\{ d(p, p_h) | p_h \in P_i^{(k)} \right\} \leq \min_{P_i} \left\{ d(p, p_j) | p_j \in P \setminus P_i^{(k)} \right\} \right\} \quad (3.4)$$

Το σύνολο $V(P_i{}^{(k)})$ ονομάζεται *πολύγωνο Voronoi k-τάξεως* που αντιστοιχεί στο σύνολο $P_i^{(k)}$. Το σύνολο των πολυγώνων *Voronoi k-τάξεως* $\mathcal{V}(A^k(P), d, \Re_m) = \mathcal{V}^{(k)} = \left\{ V(P_1^{(k)}), \ldots, V(P_n^{(k)}) \right\}$, ονομάζεται *διάγραμμα Voronoi k-τάξεως* που παράγεται από το σύνολο P. Συχνά το $\mathcal{V}^{(k)}$ στην βιβλιογραφία αναφέρεται και ως *γενικευμένο διάγραμμα Voronoi* (generalized Voronoi diagram) [OBS94].

Αν έχουμε δύο σημεία p_i και p_j η μεσοκάθετος του ευθύγραμμου τμήματος που ορίζουν δημιουργεί δύο ημιεπίπεδα. Το ημιεπίπεδο που αντιστοιχεί στο σημείο p_i το συμβολίζουμε ως $H(p_i, p_j)$. Σύμφωνα με αυτόν τον τύπο το σύνολο των σημείων που ικανοποιεί την (3.3) δίνεται και από τον τύπο:

$$H(p_{i1}, p_j) \bigcap \ldots \bigcap H(p_{ik}, p_j) \quad (3.5)$$

Έτσι το κελί *Voronoi* k-τάξεως μπορεί να οριστεί και ως εξής:

$$V(p_i^{(k)}) = \bigcap_{p_j \in P \setminus P_i^{(k)}} [H(p_{i1}, p_j) \cap \ldots \cap H(p_{ik}, p_j)] \quad (3.6)$$

Το κελί *Voronoi* του κανονικού διαγράμματος *Voronoi* μπορεί να οριστεί ως $V(p_{1j} | P \setminus \{p_j\})$. Η (3.6) τότε γράφεται:

$$V(P_i^{(k)}) = \bigcap_{h=1}^{k} V\left(p_{ih} \middle| [P \setminus P_i^{(k)}] \cup \{p_{ih}\} \right) \quad (3.7)$$

Από την σχέση (3.7) προκύπτει ότι το σύνολο $V(P_i^{(k)})$, που αντιπροσωπεύει το κελί *Voronoi* k-τάξεως μπορεί να είναι κενό. Αν δεν είναι κενό τότε μπορεί να

είναι είτε σημείο είτε περιοχή. Όταν είναι σημείο, τότε το $V(P_i^{(k)})$ δεν αποτελεί τριγωνισμό του επιπέδου. Αν είναι περιοχή, τότε είναι κυρτό πολύγωνο αφού προκύπτει από τομή κυρτών πολυγώνων. Συνολικά προκύπτει το εξής:

Λήμμα 6. *Το σύνολο $V(P_i^{(k)})$ που δίνεται από την εξίσωση (3.4) μπορεί να είναι κενό, ένα σημείο ή μια περιοχή. Υποθέτοντας ότι οι εστίες δεν βρίσκονται πάνω στον ίδιο κύκλο, ένα μη κενό $V(P_i^{(k)})$ είναι κυρτό πολύγωνο.*

Αποδεικνύεται επίσης ότι ισχύουν τα εξής :

Λήμμα 7. *Ένα μη κενό κελί Voronoi k-τάξεως περιέχει $0, 1, \ldots, k$ σημεία του P.*

Λήμμα 8. *Εάν δεν υπάρχουν σημεία του P πάνω στον ίδιο κύκλο, για κάθε κορυφή q_i ενός διαγράμματος Voronoi k-τάξεως, υπάρχει ένα μοναδικός κύκλος με κέντρο το q_i που περνάει από τρία σημεία του P και περικλείει $k-2$ ή $k-1$ σημεία του P.*

Λήμμα 9. *Έστω $n_v^{(k)}, n_e^{(k)}, n_f^{(k)}, n_u^{(k)}$ ο αριθμός κορυφών,πλευρών,k-τάξεως κελιά Voronoi και k-τάξεως ανοιχτά κελιά αντίστοιχα. Τότε αν δεν υπάρχουν σημεία του P πάνω στον ίδιο κύκλο, ισχύουν οι ακόλουθες σχέσεις:*

$$n_v^{(k)} = 2(n_f^{(k)} - 1) - n_u^{(k)}$$
$$n_e^{(k)} = 3(n_f^{(k)} - 1) - n_u^{(k)}$$

Λήμμα 10. *Εάν το P δεν έχει εστίες που βρίσκονται πάνω στον ίδιο κύκλο, ο αριθμός $n_f^{(k)}$ των μη κενών κελιών Voronoi δίνεται από την:*

$$n_f^{(k)} = (2k-1)n - (k^2-1) - \sum_{i=1}^{k} n_u^{(i-1)}$$

όπου $n_u^{(i-1)}$ είναι ο αριθμός των ανοιχτών $(i-1)$-τάξεως κελιών Voronoi. Συνεπώς το κόστος του $n_f^{(k)}$ είναι $O(k(n-k))$.

3.8 Αξιολόγηση της μεθόδου

Η λύση που προτείνεται για το πρόβλημα της αναζήτησης του εγγύτερου γείτονα εμφανίζει σημαντικά προτερήματα. Τα διαγράμματα *Voronoi* προσφέρουν σχετική ευκολία κατασκευής αλλά και μεγάλη ταχύτητα απόκρισης σε ερωτήματα εγγύτερου γείτονα. Η χρήση της τεχνικής *filter and refinement* με αξιοποίηση των διαγραμμάτων, όπως φάνηκε από τις πειραματικές μετρήσεις δίνει πάρα πολύ καλά

αποτελέσματα ακόμα και με την χρήση ενός αλγόριθμου τυφλής αναζήτησης (point in polygon). Αν εφαρμοστεί κάποιας μορφής ευρετήριο, πάνω στο διάγραμμα, ο όγκος και ο ρυθμός της εισόδου μπορεί να κλιμακωθεί χωρίς επιπτώσεις στην απόδοση του συστήματος. Τα αποτελέσματα αυτά αποτελούν μια ισχυρή ένδειξη ότι η λύση που προτείνεται μπορεί να ανταπεξέλθει σε πραγματικές συνθήκες εφαρμογής. Φυσικά αυτό είναι εμφανές και από τον μεγάλο αριθμό εφαρμογών διαφόρων τομέων, στις οποίες έχουν βρει εφαρμογή τα διαγράμματα *Voronoi*.

Υπάρχουν όμως περιπτώσεις όπου η τήρηση ολόκληρου του διαγράμματος Voronoi αποτελεί δύσκολο εγχείρημα, όπως για παράδειγμα όταν οι εστίες είναι κινούμενες, όπου και απαιτείται ο εξαρχής υπολογισμός του διαγράμματος σε κάθε αλλαγή σημειακής θέσης μιας εστίας. Πολλές φορές πάλι η πληροφορία που δίνει το ολόκληρο το διάγραμμα είναι πλεονάζουσα, λ.χ. αν ενδιαφερόμαστε για την μελέτη ενός και μόνο συγκεκριμένου κόμβου. Σε αυτές τις περιπτώσεις είναι πιο συμφέρουσα η μελέτη μεμονωμένων κελιών Voronoi, τα οποία αποτελούν και το αντικείμενο του επόμενου κεφαλαίου.

Κεφάλαιο 4

Προσεγγιστικά κελιά Voronoi k-τάξεως

Εισαγωγή

Η χρησιμότητα και η αποδοτικότητα των διαγραμμάτων Voronoi, όπως παρουσιάστηκαν και στο προηγούμενο κεφάλαιο, έχουν οδηγήσει στην ευρεία αποδοχή τους ως εργαλείο σε διάφορες εφαρμογές. Υπάρχουν όμως κατηγορίες εφαρμογών, όπου το μέγεθος των εμπλεκόμενων παραμέτρων σε συνδυασμό με την μικρή αποθηκευτική δυνατότητα και την ανάγκη για γρήγορη απόκριση, καθιστούν δύσκολη την χρήση τους, αν και τα οφέλη της γεωμετρικής προσέγγισης θα ήταν πολλά. Η λύση που προτείνεται σε τέτοιες περιπτώσεις, είναι να γίνουν συμβιβασμοί στις απαιτήσεις που τίθενται στο σύστημα, ώστε αυτό να μπορεί να αντεπεξέλθει στις ανάγκες του προβλήματος. Η ανάγκη αυτή συνήθως απορρέει από τους περιορισμούς που θέτει το τεχνολογικό υπόβαθρο στο οποίο βασίζεται το σύστημα. Για παράδειγμα οι κατανεμημένες (distributed) εφαρμογές, χρησιμοποιώντας κόμβους περιορισμένων δυνατοτήτων για να μειώσουν το κόστος, βασίζουν το λειτουργικό τους πλαίσιο στην ισχύ του συστήματος που προκύπτει συνολικά. Στις εφαρμογές πραγματικού χρόνου (real time), όπως αναφέρθηκε και στο κεφάλαιο των ρευμάτων δεδομένων, η γρήγορη απόκριση αποτελεί την κρίσιμη παράμετρο και είναι αποδεκτό να γίνονται υποχωρήσεις στην ακρίβεια.

Υπάρχουν βεβαίως και κατηγορίες εφαρμογών όπου η κατασκευή και τήρηση ολόκληρου του διαγράμματος είναι είτε περιττή (λ.χ. όταν μας ενδιαφέρει μόνο η τοπική και όχι η συνολική πληροφορία) είτε αδύνατη λόγω απουσίας των παραμέτρων ολόκληρου του συστήματος. Το πλαίσιο αυτό των εφαρμογών, οδήγησαν στην μελέτη μεμονωμένων κελιών του διαγράμματος *Voronoi*, διατηρώντας έτσι τα πλεονεκτήματα που προσφέρει η χρήση τους και ικανοποιώντας τις απαιτήσεις επεεργασίας. Σε περιπτώσεις δε που η απόκριση πρέπει να είναι άμεση, όπως σε εφαρμογές πραγματικού χρόνου, τα κελιά μπορούν να δημιουργούνται προσεγγιστικά με ένα γνωστό και υπολογίσιμο σφάλμα.

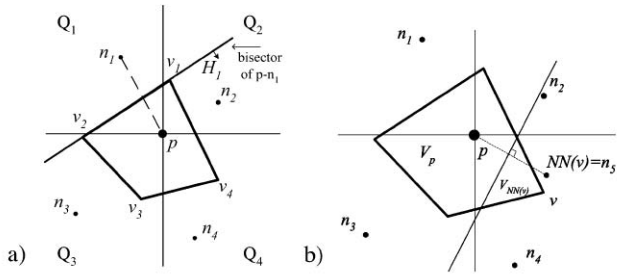
Ένα χαρακτηριστικό παράδειγμα μιας εφαρμογής αυτής της κατηγορίας, είναι ένα σύστημα αισθητήρων καταγραφής πληροφοριών. Υποθέτοντας ότι το σύστημα αποτελείται από πολλούς μικρούς αυτόνομους αισθητήρες, από τους οποίους ζητείται να καταγράψουν πληροφορίες για την θερμοκρασία ενός χώρου, η υιοθέτηση των γεωμετρικών χαρακτηριστικών του διαγράμματος Voronoi κατά τον υπολογισμό των χωρικών μέσων όρων της θερμοκρασίας αποδεικνύεται πολύ χρήσιμη και αποδοτική. Σε ένα πραγματικό περιβάλλον όμως, οι αισθητήρες υπόκεινται σε περιορισμούς τόσο επεξεργαστικής ισχύος όσο και αυτονομίας, με αποτέλεσμα να είναι απαραίτητο να γίνουν κάποιοι συμβιβασμοί στην υλοποίηση. Την λύση μπορεί να δώσει η υλοποίηση μεμονωμένων κελιών σε κάθε κόμβο, βασιζόμενη όχι στην γνώση των μετρήσεων όλων των κόμβων, αλλά σε αυτήν των πλησιέστερων κόμβων.

Η δυνατότητα των μεμονωμένων κελιών να ανανεώνουν το σχήμα τους με μεγαλύτερη ευκολία σε σχέση με το διάγραμμα, τα καθιστά πολύ ελκυστικές λύσεις σε προβλήματα αναζήτησης εγγύτερων γειτόνων, όπου τα αντικείμενα ενδιαφέροντος είναι κινούμενα. Για παράδειγμα μια εφαρμογή εύρεσης των τριών εγγύτερων ασθενοφόρων για ένα νοσοκομείο ή για έναν συνδρομητή μιας αντίστοιχης υπηρεσίας μέσω GPS. Η υιοθέτηση της χρήσης των προσεγγιστικών κελιών μπορεί να προκύψει σε εφαρμογές όπου η είσοδος είναι ένα ρεύμα δεδομένων υψηλού ρυθμού, όπως αυτό της κίνησης μεμονωμένων αυτοκινήτων στο οδικό δίκτυο μιας πόλης. Το ποσοστό λάθους που περιέχουν οι απαντήσεις είναι αποδεκτό στα πλαίσια των απαιτήσεων του συστήματος, καθώς καλύπτεται από την επίτευξη απόκρισης σε πραγματικό χρόνο.

Στο κεφάλαιο αυτό παρουσιάζεται ένας πρωτότυπος αλγόριθμος κατασκευής *προσεγγιστικών κελιών Voronoi k-τάξεως* για ένα ρεύμα κινούμενων αντικειμένων. Δίνονται δύο παραλλαγές του αλγορίθμου, ανάλογα με το αν η ανανέωση των σημειακών θέσεων του ρεύματος είναι σύγχρονη ή ασύγχρονη. Στην περίπτωση της ασύγχρονης ανανέωσης, χρησιμοποιείται χρονικά κυλιόμενο παράθυρο για την μείωση του επεξεργαστικού φόρτου του αλγορίθμου. Ο αλγόριθμος ακολουθεί την συλλογιστική που προτάθηκε από τους Sharifzadeh και Shahabi που στο [SS06] προτείνουν έναν αλγόριθμο που κατασκευάζει προσεγγιστικά κελιά *Voronoi* για έναν εγγύτερο γείτονα.

4.1 Προκαταρκτικά

Οι Sharifzadeh και Shahabi [SS06] δίνουν ένα πολύ καλό παράδειγμα ενός δικτύου αισθητήρων, όπου χρησιμοποιούν την έννοια του μεμονωμένου κελιού Voronoi, για τον υπολογισμού μιας συναθροιστικής συνάρτησης. Η εφαρμογή της ιδέας των κελιών ακολουθεί δύο μονοπάτια, αφού μελετούν την περίπτωση όπου κάθε κόμβος έχει γνώση των τιμών όλων των υπόλοιπων, όπως και την περίπτωση όπου η γνώση περιορίζεται σε μια τοπική γειτονιά. Ο τρόπος λειτουργίας του αλγόριθμου που προτείνουν και στα δύο μοντέλα είναι ο ίδιος. Η περίπτωση της τοπικής γνώσης διαφοροποιείται στο γεγονός ότι το σύνολο των εστιών N με γνωστή τιμή αλλάζει από βήμα σε βήμα, καθώς κόμβοι προσθαφαιρούνται σε αυτό.

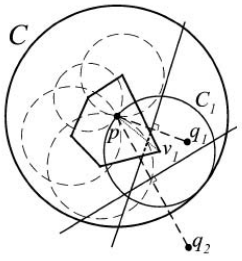


Σχήμα 4.1: (α) Προσεγγιστικό κελί Voronoi (β) Εκλέπτυνση κελιού

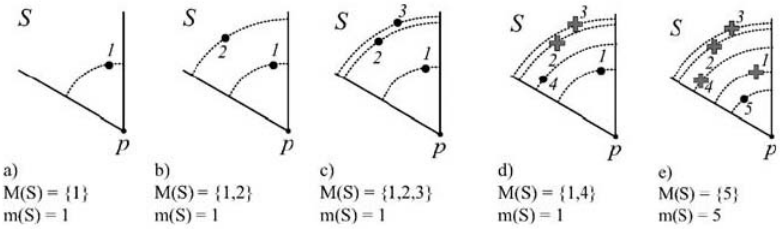
Πέρα από αυτή την διαφορά ο αλγόριθμος εξελίσσεται σε δύο στάδια. Πρώτα δημιουργείται ένα πολύ προσεγγιστικό κελί και ύστερα αυτό το κελί εκλεπτύνεται σταδιακά, μέχρι να επιτύχουμε το ακριβές κελί.

Στο σχήμα 4.1(α) φαίνεται το κελί που προκύπτει από το αρχικό στάδιο του αλγόριθμου. Το κελί είναι η τομή των μεσοκαθέτων που ορίζουν τέσσερα σημεία του επιπέδου με το σημείο ερωτήσεως p. Τα τέσσερα αυτά σημεία n_1, n_2, n_3, n_4 όπως φαίνεται και στο σχήμα είναι τα εγγύτερα στο p και ανήκει το καθένα σε ένα τεταρτημόριο. Τα τεταρτημόρια ορίζονται με αρχή των αξόνων το p. Στο στάδιο της εκλέπτυνσης (Σχήμα 4.1(β)), για κάθε κορυφή του κελιού βρίσκεται το εγγύτερο σημείο $NN(v)$ σε αυτήν και σχηματίζεται η μεσοκάθετος του $\overline{pNN(v)}$. Η πιθανή τομή της με το κελί δίνει την νέα μορφή του. Η διαδικασία επαναλαμβάνεται μέχρι να δημιουργηθεί το ακριβές κελί, όταν εξαντληθεί το σύνολο N. Μια επέκταση που προτείνεται, είναι να μην εξετάζονται σημεία του N που βρίσκονται εκτός μιας *περιοχής επιρροής* γύρω από το p. Η περιοχή επιρροής ορίζεται ως η ένωση των κύκλων που έχουν ως κέντρο τις κορυφές v_i του κελιού και ακτίνα ίση με $\overline{pv_i}$ (Σχήμα 4.2). Οποιοδήποτε σημείο βρίσκεται εκτός της περιοχής αυτής αποδεικνύεται ότι δεν μπορεί να επηρεάσει την μορφή του κελιού και άρα δεν εξετάζεται από τον αλγόριθμο.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η επέκταση του αλγόριθμου, ώστε να υπολο-



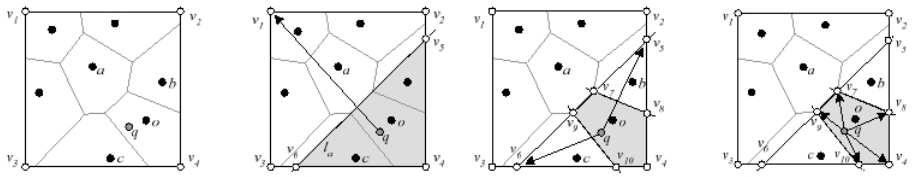
Σχήμα 4.2: Το σημείο p_1 βρίσκεται εντός της περιοχής επιρροής ενώ το p_2 όχι και άρα δεν θα εξεταστεί.



Σχήμα 4.3: Αλγόριθμος AVC-SW. Ένας κυκλικός τομέας για τέσσερα χρονόσημα.

γίζει το *προσεγγιστικό* πλέον κελί Voronoi εγγύτερου γείτονα σε μοντέλο παραθύρου (αλγόριθμος AVC-SW). Η έννοια του χρονικού παραθύρου χρησιμοποιείται τόσο για να μειώσει τον φόρτο από την επεξεργασία όλων των σημείων, όσο και για να αναδείξει την σημασία των πιο πρόσφατων δεδομένων. Θέτοντας ως εκκίνηση και πάλι το παράδειγμα του δικτύου αισθητήρων, τονίζουν ότι οι πληροφορίες που ελήφθησαν πρόσφατα είναι οι πιο σημαντικές. Έτσι θεωρούν ότι κάθε σημείο έχει μια συγκεκριμένη χρονική ισχύ, καθοριζόμενη από το χρονόσημό της. Στο σύστημα διατηρούνται μόνο τα σημεία που βρίσκονται εντός του χρονικού διαστήματος που ορίζει το παράθυρο, δηλαδή όσα είναι ακόμα ενεργά στο διάστημα αυτό. Η απόρριψη ενός σημείου γίνεται μόνο αν δεν μπορεί να επηρεάσει την μορφή του κελιού σε καμία από τις επόμενες χρονικές στιγμές που θα είναι ενεργό. Έτσι, αν και ένα σημείο n μπορεί να μην επηρεάζει αρχικά το κελί, αν κάποια άλλα σημεία ακυρωθούν (όταν λήξει η ισχύς τους) είναι δυνατόν η μεσοκάθετος που ορίζει τελικά να τροποποιεί την μορφή του κελιού.

Αρχικά το σύστημα διαμερίζει τον χώρο γύρω από το σημείο ερωτήσεως p σε λ τομείς, χρησιμοποιώντας ισάριθμα διανύσματα που ξεκινούν από το p. Η μεταξύ τους γωνία είναι ίση με $\theta = 2\pi/\lambda$. Τα σημεία που καταφθάνουν αντιστοιχίζονται σε κάποιο τομέα ανάλογα με την σχετική τους θέση ως προς το p. Στον σχηματισμό του κελιού συνεισφέρει μόνο το εγγύτερο στο p σημείο κάθε τομέα. Όταν ένα καινούργιο σημείο n καταφθάνει στο σύστημα, κάποια ήδη αποθηκευμένα σημεία μπορεί να ακυρωθούν. Αυτό εξαρτάται από την απόσταση του n από το p σε σχέση με τις αποστάσεις των ήδη αποθηκευμένων σημείων. Αν ο τομέας είναι άδειος τότε το n εισάγεται σε αυτόν και η μεσοκάθετος που ορίζει με το p συνεισφέρει στο κελί. Αν όμως υπάρχουν άλλα σημεία στον τομέα, τότε ακυρώνονται όλα τα σημεία που βρίσκονται πιο μακριά από το p σε σχέση με το n και έχουν μικρότερο χρονόσημο. Τα σημεία αυτά δεν είναι δυνατό να επηρεάσουν τον σχηματισμό του τομέα σε κανένα επόμενο χρονόσημο αφού πάντα θα καλύπτονται από το n. Θεωρώντας ότι το εγγύτερο σημείο ενός τομέα S στο p συμβολίζεται με $m(S)$ και όλα τα υπόλοιπα του τομέα εκτός αυτού με $M(S)$, στο σχήμα 4.3 απεικονίζεται η εξέλιξη του αλγόριθμου για τέσσερα χρονόσημα. Στο 4.3(a) έχουμε την εισαγωγή του σημείου 1 ($t = 1$). Τα σημεία 2 και 3 καταφθάνουν στο χρονόσημο $t = 2$ (4.3(a) και 4.3(b)). Επειδή και τα δύο βρίσκονται πιο μακριά από το p σε σχέση με το 1, δεν το ακυρώνουν. Αντίθετα στο χρονόσημο $t = 3$ το σημείο 4 όταν εισέλθει στον τομέα ακυρώνει τα 2 και 3, ευρισκόμενο εγγύτερα στο p, όπως φαίνεται στο



Σχήμα 4.4: Ερωτήματα TPNN

4.3(c). Τέλος στο 4.3(d) το σημείο 5 ακυρώνει όλα τα υπόλοιπα. Η ιδέα αυτή αποτέλεσε και την βάση για την σύλληψη του αλγόριθμου που προτείνεται στην συνέχεια για τον υπολογισμό του προσεγγιστικού κελιού k-εγγύτερων γειτόνων.

Οι Zhang, Zhu και Papadias [ZZP+03] κατασκευάζουν το ακριβές κελί *Voronoi* k-τάξεως για στατικά σύνολα σημειακών θέσεων, χρησιμοποιώντας μια διαφορετική προσέγγιση. Χρησιμοποιώντας τομές ημιεπιπέδων σε μια τοπική γειτονιά γύρω από το σημείο ερωτήσεως, δημιουργούν σταδιακά το ακριβές κελί *Voronoi*. Ο αλγόριθμος κάνει χρήση των ερωτημάτων *εγγύτερου γείτονα με χρονική παράμετρο* (TPNN). Δοθέντος ενός συνόλου στατικών σημείων και ενός κινούμενου q, τα ερωτήματα επιστρέφουν για μια συγκεκριμένη κατεύθυνση κίνησης, το εγγύτερο στατικό σημείο r στο q προς αυτή την κατεύθυνση. Υποθέτουμε ότι γνωρίζουμε τον εγγύτερο γείτονα o του q, ο οποίος εξαιρείται από τα ερωτήματα TPNN. Στην αρχή το κελί αρχικοποιείται ως ένα παραλληλόγραμμο που περιέχει όλα τα ακίνητα σημεία N (σχήμα 4.4(α)). Επιλέγεται τυχαία μια κορυφή v του κελιού και ορίζεται ως κατεύθυνση του ερωτήματος TPNN το διάνυσμα $\overrightarrow{qv}$. Το ερώτημα επιστρέφει το εγγύτερο στατικό σημείο n και η μεσοκάθετος που ορίζει το $\overline{qn}$, χρησιμοποιείται για να ενημερώσουμε το κελί (σχήμα 4.4(β)). Ο αλγόριθμος συνεχίζει επαναληπτικά με ένα νέα ερωτήματα TPNN, επιλέγοντας τυχαία ανάμεσα από τις κορυφές του κελιού αυτή που θα ορίσει την κατεύθυνση (σχήμα 4.4(γ)). Η διαδικασία τερματίζει όταν δεν βρίσκονται νέα σημεία για να ενημερωθεί το κελί (σχημα 4.4(δ)). Ο αλγόριθμος αξιοποιεί το κελί που υπολογίσθηκε, ως περιοχή εγκυρότητας των εγγύτερων γειτόνων που βρέθηκαν. Συνυπολογίζοντας παραμέτρους όπως η ταχύτητα κίνησης του σημείου ερωτήσεως προβλέπει τις επόμενες χρονικές στιγμές κατά τις οποίες αυτό θα βρίσκεται μέσα στο κελί και άρα δεν υπάρχει ανάγκη επανυπολογισμού των γειτόνων.

Ο αλγόριθμος υπολογίζει το κελί για έναν εγγύτερο γείτονα, αλλά επεκτείνεται και για κελιά k εγγύτερων γειτόνων, διαφοροποιώντας ελάχιστα την πορεία του. Στην περίπτωση των k γειτόνων, είναι γνωστοί από την αρχή οι k εγγύτεροι γείτονες του q $(o_1, \ldots, o_k)$. Τα ερωτήματα που τίθενται πλέον είναι ερωτήματα TPkNN, που επιστρέφουν το εγγύτερο σημείο r προς την ζητούμενη κατεύθυνση, μαζί με τον εγγύτερο γείτονα o_i που ακυρώνεται αν το q περάσει την μεσοκάθετο του $\overline{qo_i}$. Η μεσοκάθετος αυτή αξιοποιείται όπως και πριν, ενώ ερωτήματα TPkNN τίθενται μέχρις ότου πάψουν να επιστρέφονται καινούργιες απαντήσεις.

4.2 Αλγόριθμος υπολογισμού προσεγγιστικού κελιού Voronoi k τάξης

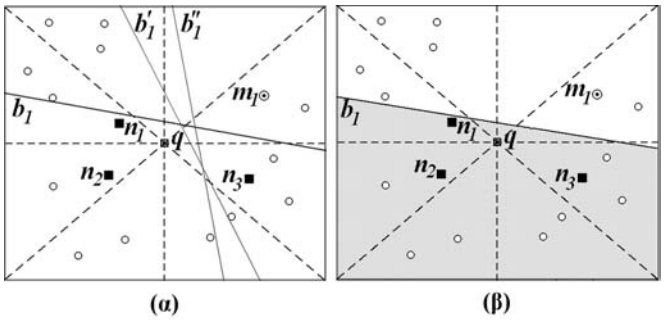
Ο αλγόριθμος υπολογίζει το προσεγγιστικό κελί Voronoi k-τάξεως για ένα ρεύμα αντικειμένων. Σκοπός είναι να μπορεί να παρέχει απαντήσεις για σύνολα αντικειμένων που καταφθάνουν με μεγάλο ρυθμό. Για αυτό το σκοπό επιλέχτηκε η προσέγγιση του κελιού Voronoi, καθώς σε εφαρμογές πραγματικού χρόνου η απόκριση αποτελεί πιο σημαντική παράμετρο από την ακρίβεια της απάντησης. Το προσεγγιστικό κελί δημιουργείται για κάθε χρονόσημο και για ένα σημείο ερωτήσεως. Ο σχεδιασμός του αλγόριθμου αποκλείει την εφαρμογή του για περισσότερα σημεία ερωτήσεως ,αφού κάθε κατάτμηση είναι ανεξάρτητη από τις υπόλοιπες. Η επεξεργασία πολλαπλών ερωτημάτων μπορεί να γίνει με ξεχωριστή εκτέλεση του αλγορίθμου για κάθε σημείο ερωτήσεως, είτε σε πολλαπλούς επεξεργαστές είτε σε πολλαπλά νήματα εκτέλεσης (threads).

4.2.1 Αλγόριθμος ταυτόχρονης ανανέωσης

Θεωρούμε ένα σύνολο $P = \{p_1, p_2, \ldots, p_m\}$ από N κινούμενα αντικείμενα στον δισδιάστατο χώρο. Στον ίδιο χώρο κινείται και το σημείο ερωτήσεως Q (query point). Ο αλγόριθμος στηρίζεται στην ιδέα της ισομερούς διαμέρισης του επιπέδου σε κυκλικούς τομείς με κέντρο το σημείο ερωτήσεως Q. Έτσι ο δισδιάστατος χώρος διαμερίζεται σε λ τομείς, χρησιμοποιώντας λ διανύσματα που ξεκινούν από το Q και έχουν ίση γωνιακή απόσταση μεταξύ τους $\theta = 2\pi/\lambda$. Το σύνολο των τομέων ονομάζεται S και ορίζεται ως $S = \{S_1, S_2, \ldots, S_\lambda\}$. Το εύρος των τομέων είναι σημαντικό χαρακτηριστικό του αλγόριθμου και επηρεάζει σημαντικά την ακρίβεια του παραγόμενου κελιού.

Αφού δημιουργηθούν οι τομείς, το σύστημα είναι έτοιμο να δεχθεί τις σημειακές θέσεις των κινούμενων αντικειμένων. Ο αλγόριθμος υλοποιεί μια μέθοδο *αποβολής φόρτου* (load shedding), η οποία επιτρέπει σε κάθε κυκλικό τομέα να αποθηκεύει το πολύ τις $k + 1$ εγγύτερες σημειακές θέσεις στο Q. Ο αριθμός αυτός είναι το ελάχιστο όριο για το οποίο μπορούμε να είμαστε σίγουροι ότι στο τέλος του τρέχοντος χρονόσημου (timestamp), θα υπάρχουν στο σύστημα οι k εγγύτεροι γείτονες που ζητήθηκαν και μια σημειακή θέση που θα δημιουργήσει το κελί. Έτσι για κάθε μία σημειακή θέση που διαβάζεται από το ρεύμα εισόδου, εξετάζεται η θέση της στο επίπεδο και αντιστοιχίζεται σε έναν από τους κυκλικούς τομείς. Εάν ο τομέας έχει λιγότερες από $k+1$ αποθηκευμένες σημειακές θέσεις, τότε εισάγεται στην λίστα σημείων του. Σε αντίθετη περίπτωση μόνο αν βρίσκεται εγγύτερα στο Q από την πιο απομακρυσμένη σημειακή θέση της λίστας αποθηκεύεται παίρνοντας την θέση της, αλλιώς απορρίπτεται. Η διαδικασία αυτή γίνεται για όλες τις επόμενες σημειακές θέσεις και συνεχίζεται μέχρι να ανιχνευθεί χρονόσημο διαφορετικό από το τρέχον (*function UpdateSectors*).

Στο τέλος κάθε χρονοσήμου ξεκινάει η διαδικασία κατασκευής του κελιού με την εύρεση των k εγγύτερων γειτόνων του Q από το σύνολο των ενεργών αντικειμένων στο σύστημα. Για λόγους απλότητας η εύρεση ακολουθεί τυφλή αναζήτηση



Σχήμα 4.5: (α) Εύρεση πιο περιοριστικής μεσοκαθέτου. (β) Περικοπή κελιού από μεσοκάθετο.

στο σύνολο των αντικειμένων. Στη συνέχεια για κάθε τομέα εντοπίζεται και μαρκάρεται το *σημείο ελέγχου*, που αποτελεί την εγγύτερη στο σημείο ερωτήσεως σημειακή θέση που δεν έχει σημαδευτεί στο προηγούμενο βήμα ως εγγύτερος γείτονας του Q (*function InitPointLists*).

Τα κινούμενα αντικείμενα θεωρούμε ότι μετακινούνται σε έναν περιορισμένο χώρο, του οποίου γνωρίζουμε τις συντεταγμένες. Δημιουργούμε λοιπόν την πρώιμη μορφή του κελιού ως ένα παραλληλόγραμμο που περιλαμβάνει όλη την περιοχή μελέτης. Στη συνέχεια αξιοποιούμε τα σημεία ελέγχου που βρέθηκαν προηγούμενα για να δημιουργήσουμε το προσεγγιστικό κελί Voronoi. Αυτό γίνεται εξετάζοντας την περιοχή μελέτης και βρίσκοντας τις μεσοκαθέτους που ορίζουν τα ευθύγραμμα τμήματα που ενώνουν το σημείο ελέγχου κάθε τομέα, με κάθε έναν από τους εγγύτερους γείτονες. Από αυτές τις μεσοκαθέτους για κάθε τομέα επιλέγεται αυτή που βρίσκεται σε μικρότερη απόσταση από το Q. Η μεσοκάθετος αυτή ονομάζεται *πιο περιοριστική μεσοκάθετος* για το σημείο ελέγχου. Για να βρεθεί, υπολογίζεται η ακτίνα όλων των κύκλων με κέντρο το σημείο ερωτήσεως Q που εφάπτονται στις μεσοκαθέτους. Η μεσοκάθετος που εφάπτεται στον κύκλο με την μικρότερη ακτίνα είναι η πιο περιοριστική. Η μεσοκάθετος αυτή χρησιμοποιείται στην συνέχεια για να σχηματιστεί το κελί (σχήμα 4.5(α)), βρίσκοντας την τομή της με το υπάρχον σχήμα του κελιού που αρχικά είναι παραλληλόγραμμο (σχήμα 4.5(β)). Η διαδικασία αυτή όταν γίνει για όλους τους τομείς, με αντι-ωρολογιακή φορά, μας δίνει το προσεγγιστικό κελί για το τρέχον χρονόσημο (*function CreateCell*).

Ο αλγόριθμος όπως περιγράφηκε, βρίσκει εφαρμογή στην περίπτωση όπου τα κινούμενα αντικείμενα ανανεώνουν τις σημειακές τους θέσεις ταυτόχρονα σε κάθε χρονόσημο. Το σημείο ερωτήσεως Q θεωρείται ακίνητο. Στην περίπτωση που είναι κινητό, τότε ο αλγόριθμος σε κάθε μετακίνηση του Q πρέπει να προσαρμόζει την κατάτμηση του χώρου στην νέα σημειακή του θέση. Ακολουθεί η περιγραφή του σε μορφή ψευδοκώδικα.

Αλγόριθμος σύγχρονης ανανέωσης

Function *UpdateSectors* (point q, int λ, int k, interval $\Delta\tau$)

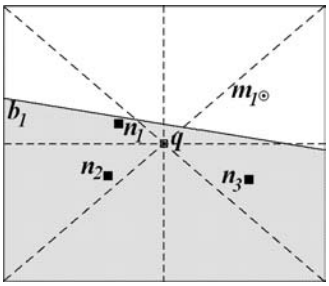
1. Με κέντρο το σημείο ερωτήσεως q χώρισε το ευκλείδιο επίπεδο E σε λ ισογωνιακούς τομείς $S_1, S_2, \ldots, S_\lambda$
2. $L(S_i) \leftarrow \emptyset$
3. **while** ($o_j \leftarrow$ επόμενη σημειακή θέση στο P) $\neq$ NULL **do**
4. $S_i \leftarrow$ τομέας τ.ω. o_j ανήκει στον S_i.
5. **if** $|L(S_i)| < k+1$ **then**
6. Βάλε την o_j με αύξουσα σειρά στην $L(S_i)$
7. **else** $v \leftarrow (k+1)$-στή σημειακή θέση $\in L(S_i)$
8. **if** $dist(o_j, q) < dist(v, q)$ **then**
9. Εισήγαγε την o_j με αύξουσα σειρά στην $L(S_i)$
10. Αφαίρεσε την v από την $L(S_i)$
11. **return** $L(S_i)$, $i = 1, \ldots \lambda$

Function *InitPointLists* (point q, int λ, int k, points L)

1. $kNNList \leftarrow \emptyset$, $MList \leftarrow \emptyset$
2. **for each** τομέα S_i, $i = 1, \ldots \lambda$
3. **while** ($p \leftarrow$ επόμενη σημειακή θέση $\in L(S_i)$) $\neq$ NULL **do**
4. **if** $|kNNList| < k$ **then**
5. $kNNList \leftarrow kNNList \cup p$
6. **else**
7. **while** ($r \leftarrow$ επόμενη σημειακή θέση $\in kNNList$) $\neq$ NULL **do**
8. **if** $dist(p, q) < dist(r, q)$ **then**
9. Εισήγαγε την p πριν από την r στην $kNNList$
10. Ολίσθησε όλες τις επόμενες θέσεις στην $kNNList$
11. Σημείωσε την p ως επιλεγμένη στην $L(S_i)$
12. **if** ($v \leftarrow (k+1)$-στό σημείο $\in kNNList$) $\neq$ NULL
13. **then** Αφαίρεσε την v από την $kNNList$
14. Σημείωσε την v ως μη επιλεγμένη στην $L(S_j)$
15. **else break**
16. **for each** τομέα S_i, $i = 1, \ldots \lambda$
17. $m(S_i) \leftarrow$ την πρώτη *μη επιλεγμένη* σημειακή θέση στην $L(S_i)$
18. $MList \leftarrow MList \cup m(S_i)$
19. **return** $kNNList$, $MList$

Function *CreateCell* (point q, points $kNNList$, points $MList$)

1. $V_q \leftarrow$ επίπεδο κίνησης E
2. **for each** σημειακή θέση $m(S_i) \in MList$



Σχήμα 4.6: Αρχικό κελί για $k = 3$.

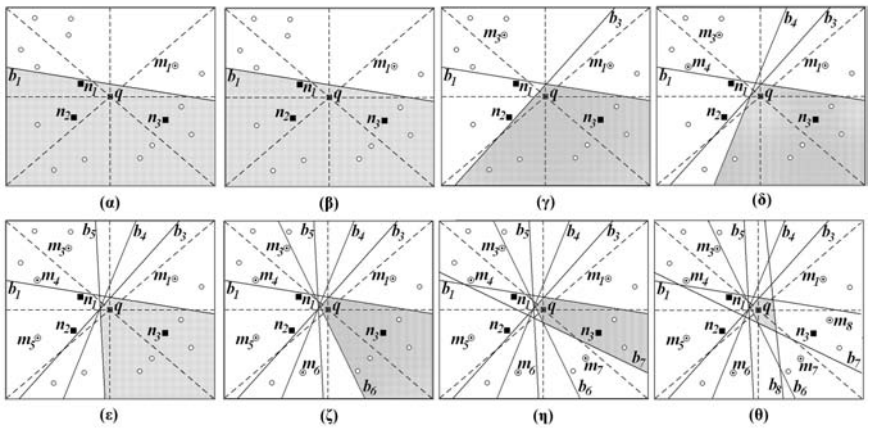
3. $B_i \leftarrow \emptyset$
4. **for each** σημειακή θέση $n_j \in kNNList$
5. Εισήγαγε στην B_i την μεσοκάθετο του ευθύγραμμου τμήματος με άκρα τις $m(S_i)$ και n_j
6. $b_i \leftarrow$ επέλεξε την μεσοκάθετο από το B_i που βρίσκεται εγγύτερα στο q
7. Εκλέπτυνε το V_q με το b_i
8. **return** V_q

4.2.2 Μοντέλο χρονικά κυλιόμενου παραθύρου

Μια επέκταση με ιδιαίτερο ενδιαφέρον είναι αυτή της εφαρμογής ενός χρονικά κυλιόμενου παραθύρου στο σύστημα. Σε πολλές εφαρμογές πραγματικού χρόνου τα αντικείμενα που το σύστημα καλείται να παρακολουθήσει, δεν ανανεώνουν τις σημειακές τους θέσεις σύγχρονα, όπως για παράδειγμα αν το σύστημα τροφοδοτείται με στοιχεία κίνησης αυτοκινήτων στο οδικό δίκτυο μιας πόλης. Σε τέτοιες εφαρμογές όμως ιδιαίτερη αξία έχουν τα πιο πρόσφατα δεδομένα και η διατήρηση του συνόλου των δεδομένων είναι περιττή. Η εφαρμογή ενός χρονικά κυλιόμενου παραθύρου (λ.χ. θέσεις των αυτοκινήτων τα τελευταία 5 λεπτά) επιτρέπει στο σύστημα αφενός να διαχειριστεί τα δεδομένα αποτελεσματικότερα, ενώ ταυτόχρονα μειώνεται και ο φόρτος του δικτύου που αναλαμβάνει την μετάδοση των στοιχείων.

Δεχόμαστε ότι τα κινούμενα αντικείμενα έχουν σημειακές θέσεις με χρονική διάρκεια ισχύος μεγαλύτερη του ενός χρονοσήμου και ότι αυτές δεν ανανεώνονται για όλα τα αντικείμενα ταυτόχρονα. Έτσι στο εύρος των χρονοσήμων που καλύπτει το παράθυρο, υπάρχουν σημειακές θέσεις με διαφορετική υπολειπόμενη χρονική ισχύ, σενάριο που πλησιάζει περισσότερο τις συνθήκες μιας πραγματικής εφαρμογής. Για λόγους απλότητας γίνεται επίσης η παραδοχή ότι όλες οι σημειακές θέσεις έχουν χρονική ισχύ $\Delta\tau$ χρονόσημα, ίση με το εύρος του παραθύρου. Ένα δεύτερο χαρακτηριστικό είναι η αλλαγή του μέγιστου αριθμού σημειακών θέσεων που αποθηκευόνται σε κάθε τομέα. Η λίστα σημειακών θέσεων κάθε τομέα δεν μπορεί να διατηρεί για κάθε χρονόσημο εντός παραθύρου αριθμό εγγραφών μεγαλύτερο από $k + 1$. Ο περιορισμός αυτός μας εξασφαλίζει την ύπαρξη του κελιού για κάθε χρονόσημο, αλλά ταυτόχρονα μειώνει σημαντικά τον αποθηκευτικό και υπολογιστικό φόρτο.

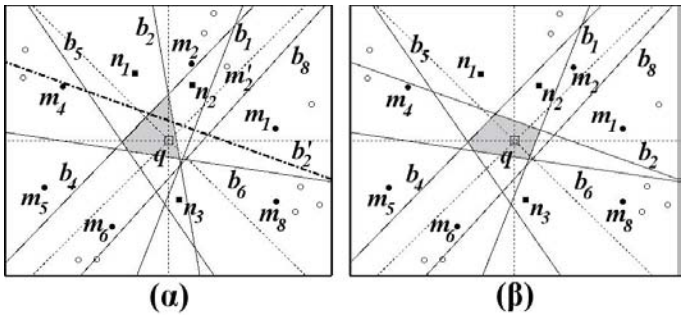
Ο αλγόριθμος χωρίζεται σε δύο φάσεις, στην φάση αρχικοποίησης και στην φάση ανανέωσης. Την φάση αρχικοποίησης προηγείται η εισαγωγή ενός συγκεκριμένου



Σχήμα 4.7: Δημιουργία κελιού.

αριθμού σημειακών θέσεων στο σύστημα (*function UpdateSectors*). Συγκεκριμένα ο ελάχιστος αριθμός αυτός είναι $k+1$ σημειακές θέσεις, δηλαδή ίσος με αυτόν των εγγύτερων γειτόνων που αναζητούμε συν ένα. Η επιπρόσθετη σημειακή θέση απαιτείται για να χρησιμοποιηθεί στην δημιουργία ενός αρχικού κελιού των k υπολοίπων σημειακών θέσεων. Προφανώς αυτή η μία σημειακή θέση επιλέγεται να είναι η πιο απομακρυσμένη από τις $k+1$. Για να επιτευχθεί καλύτερη απόκριση είναι δυνατή η εισαγωγή ένος μεγαλύτερου αριθμού σημειακών θέσεων. Στην *φάση της αρχικοποίησης*, δημιουργείται ένα αρχικό κελί (σχήμα 4.6) το οποίο είναι η βάση για την δημιουργία του τελικού. Αυτό γίνεται χρησιμοποιώντας τις σημειακές θέσεις που έχουν εισέλθει ως τώρα στο σύστημα ως εξής. Αρχικά, γίνεται αναζήτηση σε όλους τους κυκλικούς τομείς και σημαδεύονται ως οι k εγγύτεροι γείτονες του Q, τα k πλησιέστερα σημεία σε αυτό. Αυτό μπορεί να γίνει με αλγόριθμο τυφλής αναζήτησης λόγω του μικρού αριθμού των σημειακών θέσεων. Ύστερα, για κάθε τομέα βρίσκεται και σημαδεύεται το *σημείο ελέγχου* του (*function InitPointLists*) και για τα ενεργά σημεία του συστήματος δημιουργείται το κελί με την ίδια διαδικασία που ακολουθήθηκε στον αλγόριθμο ταυτόχρονης ανανέωσης (*function CreateCell*). Δηλαδή η πρώιμη μορφή του κελιού ορίζεται ως ένα παραλληλόγραμμο που περιλαμβάνει όλη την περιοχή μελέτης, το οποίο εκλεπτύνεται διαδοχικά για κάθε κυκλικό τομέα από την *πιο περιοριστική μεσοκάθετο* του σημείου ελέγχου του (σχήμα 4.7).

Στη συνέχεια αρχίζει η *φάση της ανανέωσης* χωριστά για κάθε διακριτό χρονόσημο, όπου το σύστημα δέχεται και πάλι νέες σημειακές θέσεις. Αφού βρεθεί ο τομέας όπου αντιστοιχεί μια καινούργια σημειακή θέση p, εξετάζεται αν αυτή πληροί τις προϋποθέσεις εισαγωγής. Αν για το τρέχον χρονόσημο η λίστα σημειακών θέσεων δεν είναι πλήρης, η p εισάγεται εκεί. Αν όμως η λίστα είναι πλήρης, τότε γίνεται έλεγχος αν η απόρριψη της σημειακής θέσης p θα επηρεάσει την μορφή του κελιού στα επόμενα χρονόσημα. Αυτό συμβαίνει μόνο αν η p βρίσκεται εγγύτερα στο Q από κάποια άλλη ήδη αποθηκευμένη σημειακή θέση r και ταυτόχρονα η r έχει χρονόσημο μικρότερο από την p. Στην περίπτωση που βρεθούν πολλαπλές σημειακές θέσεις που πληρούν τα κριτήρια, επιλέγεται η πιο απομακρυσμένη. Με την πλήρωση των συνθηκών η παλιά σημειακή θέση r αφαιρείται από το σύστημα για όλα τα χρονόσημα και η καινούργια αποθηκεύεται στην λίστα.



Σχήμα 4.8: Φάση ανανέωσης : (α) Το σημείο m_2 ακυρώνεται και το m_2' γίνεται το νέο σημείο ελέγχου (β) Η μορφή του προσεγγιστικού κελιού ανανεώνεται από τις νέες μεσοκαθέτους, για τους εγγύτερους γείτονες n_1, n_2, n_3.

Εφόσον δεν ισχύουν οι συνθήκες η σημειακή θέση p απορρίπτεται αφού αποκλείεται να επηρεάσει την μορφή του κελιού (*function UpdateSectors*).

Η διαδικασία συνεχίζεται μέχρι να ανιχνευθεί χρονόσημο διαφορετικό από το τρέχον τ, οπότε και γίνεται η ανανέωση του υπάρχοντος κελιού (*function MaintainCell*). Για κάθε τομέα πρέπει να ελεγχθεί αν έχει αλλάξει το σημείο ελέγχου και αν κάποια σημειακή θέση του τομέα βρίσκεται πλησιέστερα στο Q από τους τωρινούς εγγύτερους γείτονες. Εάν βρεθεί ένας καινούργιος εγγύτερος γείτονας τότε ο αλγόριθμος επανεκκινεί από την φάση αρχικοποίησης, με τις υπάρχουσες σημειακές θέσεις. Δηλαδή το κελί ξαναπαίρνει το σχήμα του περιβάλλοντος παραλληλογράμμου και βρίσκονται οι τομές του με τις νέες πιο περιοριστικές μεσοκαθέτους κάθε τομέα. Αν έχει αλλάξει μόνο το σημείο ελέγχου τότε είναι απαραίτητο να γίνει ανανέωση της πλευράς του κελιού που αντιστοιχεί στον τομέα αυτό. Για να επιτευχθεί αυτό χρειάζεται να αναζητηθεί και να αφαιρεθεί (αν υπάρχει) η πλευρά αυτή από το κελί. Το κελί επεκτείνεται και κλείνει είτε με την τομή των γειτονικών πλευρών (αν είναι συγκλίνουσες) είτε στο περιβάλλον παραλληλόγραμμο (αν είναι αποκλίνουσες). Στην συνέχεια το κελί εκλεπτύνεται βρίσκοντας την τομή του με την πιο περιοριστική μεσοκάθετο που ορίζεται από το νέο σημείο ελέγχου (σχήμα 4.8).

Πριν την αρχή του επόμενου χρονοσήμου $\tau+1$, πραγματοποιείται ολίσθηση του χρονικού παραθύρου με την αφαίρεση όλων των σημειακών θέσεων που έχουν λήξει. Στην περίπτωση που το σημείο ερωτήσεως Q είναι ακίνητο, ο αλγόριθμος επαναλαμβάνεται μέχρι το πέρας του ρεύματος εισόδου. Αν και το Q αλλάζει σημειακή θέση (λογικά με ρυθμό κατά πολύ μικρότερο από αυτόν των κινούμενων αντικειμένων) τότε σε κάθε αλλαγή της πρέπει να εκκινηθεί ο αλγόριθμος από την αρχή. Αυτό γίνεται με την αντιγραφή των ενεργών σημειακών θέσεων σε μια προσωρινή λίστα, την δημιουργία νέας κατάτμησης για την νέα θέση του Q και την αναδιανομή των ενεργών σημειακών θέσεων σε αυτή. Ο αλγόριθμος συνεχίζει με την ανάγνωση των σημειακών θέσεων για το τρέχον χρονόσημο και την δημιουργία του αρχικού κελιού (*function UpdateQueryPoint*).

Αλγόριθμος ασύγχρονης ανανέωσης με χρήση χρονικού παραθύρου

Function *UpdateSectors* (point q, int λ, int k, interval $\Delta\tau$)

1. Με κέντρο το σημείο ερωτήσεως q χώρισε το ευκλείδιο επίπεδο E σε λ ισογωνιακούς τομείς $S_1, S_2, \ldots, S_\lambda$
2. **for each** $L(S_i)$, $i = 1, \ldots \lambda$
3. **for each** $p \in L(S_i)$
4. **if** $p.\tau \leq \tau_{NOW} - \Delta\tau$ **then**
5. Αφαίρεσε το p από την $L(S_i)$
6. **while** ($o_j \leftarrow$ η επόμενη σημειακή θέση με χρονόσημο τ_{NOW} στο P) $\neq$ NULL **do**
7. $S_i \leftarrow$ τομέας τ.ω. o_j βρίσκεται μέσα στο S_i.
8. **if** $|\{m \in L(S_i)$ με $m.\tau = \tau_{NOW}\}| < k + 1$ **then**
9. Εισήγαγε την o_j με αύξουσα σειρά στην $L(S_i)$
10. **else** $v \leftarrow (k+1)$-στή σημειακή θέση στην $\in L(S_i)$
11. **if** $dist(o_j, q) < dist(v, q)$ **and** $v.\tau \leq \tau_{NOW}$ **then**
12. Εισήγαγε την o_j με αύξουσα σειρά στην $L(S_i)$
13. Αφαίρεσε την v από την $L(S_i)$
14. **return** $L(S_i)$, $i = 1, \ldots \lambda$

Function *InitPointLists* (point q, int λ, int k, points L)

1. $kNNList \leftarrow \emptyset$, $MList \leftarrow \emptyset$
2. **for each** τομέα S_i, $i = 1, \ldots \lambda$
3. **while** ($p \leftarrow$ επόμενη σημειακή θέση $\in L(S_i)$) $\neq$ NULL **do**
4. **if** $|kNNList| < k$ **then**
5. $kNNList \leftarrow kNNList \cup p$
6. **else**
7. **while** ($r \leftarrow$ επόμενη σημειακή θέση $\in kNNList$) $\neq$ NULL **do**
8. **if** $dist(p, q) < dist(r, q)$ **then**
9. Εισήγαγε την p πριν από την r στην $kNNList$
10. Ολίσθησε όλες τις επόμενες θέσεις στην $kNNList$
11. Σημείωσε την p ως επιλεγμένη στην $L(S_i)$
12. **if** ($v \leftarrow (k+1)$-στό σημείο $\in kNNList$) $\neq$ NULL
13. **then** Αφαίρεσε την v από την $kNNList$
14. Σημείωσε την v ως μη επιλεγμένη στην $L(S_j)$
15. **else break**
16. **for each** τομέα S_i, $i = 1, \ldots \lambda$
17. $m(S_i) \leftarrow$ την πρώτη *μη επιλεγμένη* σημειακή θέση στην $L(S_i)$
18. $MList \leftarrow MList \cup m(S_i)$
19. **return** $kNNList$, $MList$

Function *CreateCell* (point q, points $kNNList$, points $MList$)

1. $V_q \leftarrow$ επίπεδο κίνησης E
2. **for each** σημειακή θέση $m(S_i) \in MList$
3. $B_i \leftarrow \emptyset$
4. **for each** σημειακή θέση $n_j \in kNNList$
5. Εισήγαγε στην B_i την μεσοκάθετο του ευθύγραμμου τμήματος με άκρα τις $m(S_i)$ και n_j
6. $b_i \leftarrow$ επέλεξε την μεσοκάθετο από το B_i που βρίσκεται εγγύτερα στο q
7. Εκλέπτυνε το V_q με το b_i
8. **return** V_q

Function *MaintainCell* (point q, points $kNNList$, points $MList$)

1. **if** κάποια από τις σημειακές θέσεις στην $kNNList$ έχει αλλάξει **then**
2. Κάλεσε την *CreateCell*(q, $kNNList$, $MList$)
3. **else**
4. **for each** τομέα S_i, $i = 1, \ldots \lambda$
5. **if** $m(S_i) \neq m'(S_i) \in MList$ **then**
6. $B_i \leftarrow \emptyset$
7. **for each** σημειακή θέση $n_j \in kNNList$
8. Εισήγαγε στην B_i την μεσοκάθετο του ευθύγραμμου τμήματος με άκρα τις $m'(S_i)$ και n_j
9. $b'_i \leftarrow$ επέλεξε την μεσοκάθετο από το B_i που είναι εγγύτερη στο q
10. Αφαίρεσε την υπάρχουσα μεσοκάθετο b_i που δημιουργείται από το $m(S_i)$
11. Εκλέπτυνε το υπάρχον V_q με την b'_i
12. **return** V_q

Function *UpdateQueryPoint* (point q, point q_{new}, int λ, points $BackupList$, points L)

1. $kNNList \leftarrow \emptyset$, $BackupList \leftarrow \emptyset$
2. **for each** τομέα S_i, $i = 1, \ldots \lambda$
3. **while** ($p \leftarrow$ επόμενη σημειακή θέση $\in L(S_i)$) $\neq$ NULL **do**
4. $BackupList \leftarrow BackupList \cup p$
5. Αφαίρεσε την p από την $L(S_i)$
6. Κάλεσε την InitPointLists(q_{new}, k, L)
7. **while** ($p \leftarrow$ επόμενη σημειακή θέση $\in BackupList$) $\neq$ NULL **do**
8. $S_i \leftarrow$ τομέας τ.ω. o_j βρίσκεται μέσα στο S_i.
9. **if** $|L(S_i)| < k + 1$ **then**
10. Εισήγαγε την o_j με αύξουσα σειρά στην $L(S_i)$
11. **else** $v \leftarrow (k+1)$-στή σημειακή θέση στην $\in L(S_i)$
12. **if** $dist(o_j, q) < dist(v, q)$ **and** $v.\tau \leq \tau_{NOW}$ **then**
13. Εισήγαγε την o_j με αύξουσα σειρά στην $L(S_i)$
14. Αφαίρεσε την v από την $L(S_i)$
15. **return** $L(S_i)$, $i = 1, \ldots \lambda$

4.3 Ιδιότητες προσεγγιστικών κελιών Voronoi k-τάξεως

Λήμμα 1. Το προσεγγιστικό κελί $V'(p)$ περιέχει πλήρως το ακριβές κελί $V(p)$.

Απόδειξη. Το $V'(p)$ κατασκευάζεται από ένα υποσύνολο των μεσοκαθέτων που κατασκευάζουν το $V(p)$. Έστω ένα εσωτερικό σημείο $q \in V(p)$ με $q \notin V'(p)$. Τότε, πρέπει να υπάρχει ένα σημείο τομής των q μεσοκαθέτων (αυτό που καθορίζει μέρος του $V'(p)$) που δεν έχει συνυπολογιστεί στην κατασκευή του $V(p)$. Αυτό έχει σαν αποτέλεσμα, η κορυφή q να βρίσκεται εγγύτερα στο p από ότι αυτές που ορίζουν το $V(p)$, το οποίο αποτελεί άτοπο. □

Λήμμα 2. Το προσεγγιστικό κελί $V'(p)$ είναι κυρτό πολύγωνο.

Απόδειξη. Το $V'(p)$ κατασκευάζεται λαμβάνοντας υπόψιν $\lambda \geq 2$ ημιεπίπεδα που ορίζονται από τις αντίστοιχες μεσοκαθέτους. Εφόσον τα ημιεπίπεδα αυτά είναι κυρτά πολύγωνα και η τομή κυρτών πολυγώνων είναι κυρτή, η πρόταση ισχύει. □

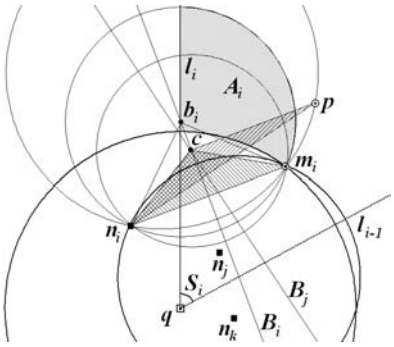
Λήμμα 3. Το προσεγγιστικό κελί $V'(p)$ μπορεί να περικλείει περισσότερα από k σημεία του συνόλου P των κινούμενων αντικειμένων.

Το φαινόμενο εξαρτάται από την σχετική θέση των σημείων ελέγχου σε σύγκριση με τις k εγγύτερες σημειακές θέσεις που οριοθετούν το κελί, όπως επίσης και με την επιλεγμένη γωνία θ των τομέων.

Η μεσοκάθετος B_i που αντιστοιχεί στο σημείο ελέγχου m_i του τομέα S_i τέμνει τουλάχιστον έναν από τους ημιάξονες l_{i-1} και l_i που οριοθετούν τον τομέα. Ονομάζουμε κάθε τέτοια τομή, *σημείο βάσης* b_i της μεσοκαθέτου B_i σε σχέση με τον ημιάξονα l_i. Έστω ένα κύκλος C_i με κέντρο το σημείο βάσης b_i με ακτίνα ίση με $|b_i m_i|$. Το τμήμα του κύκλου που βρίσκεται εντός του κυκλικού τομέα S_i έχει μια επικάλυψη με την κυκλική περιοχή με κέντρο το σημείο ερωτήσεως Q και ακτίνα $|qm_i|$. Αυτή η περιοχή *ακύρωσης* A_i οριοθετείται από:

1. τον άξονα l_i (αυτόν που ορίζει τον κυκλικό τομέα S_i τεμνόμενος με την μεσοκάθετο B_i στο σημείο βάσης b_i,
2. την χορδή x_i του κύκλου $O(q, |qm_i|)$ που βρίσκεται εντός του S_i, και
3. την περιφέρεια του κύκλου $C_i(b_i, |b_i m_i|)$.

Οι σημειακές θέσεις που βρίσκονται μέσα σε αυτή την κοινή περιοχή μπορούν δυνητικά να εισάγουν μεσοκαθέτους που τέμνουν την B_i εντός του τομέα S_i και μπορεί να είναι εγγύτερες της B_i στο σημείο ερωτήσεως Q. Αυτό έχει ως αποτέλεσμα οι μεσοκάθετοι αυτές να μπορούν να επηρεάσουν το προσεγγιστικό κελί *Voronoi* που επιστρέφει ο αλγόριθμος. Από την άλλη, είναι σίγουρο ότι οποιαδήποτε σημειακή θέση που βρίσκεται εκτός αυτών (το πολύ δύο) των περιοχών



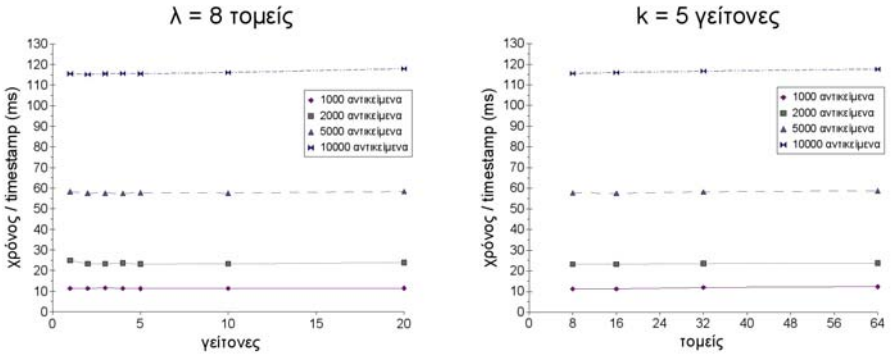
Σχήμα 4.9: Σημειακή θέση που ακυρώνει μια μεσοκάθετο.

ακύρωσης, που αντιστοιχούν στα σημεία βάσης b_{i-1} και b_i δεν μπορούν να επηρεάσουν την μεσοκάθετο B_i εντός του τομέα S_i. Εντούτοις, κάθε σημειακή θέση p εντός του τομέα S_i με $|qp| > |qm_i|$ μπορεί πιθανώς να ακυρώσει την μεσοκάθετο B_i αν κατασκευάζει την πιο περιοριστική μεσοκάθετό του (ΠΠΜ) B_j με διαφορετικό $n_j \neq n_i$ από τους k-εγγύτερους γείτονες.

Λήμμα 4. Μια σημειακή θέση p μπορεί να ακυρώσει μερικώς την μεσοκάθετο B_i που σχηματίζει το σημείο ελέγχου m_i του κυκλικού τομέα S_i με έναν από του k εγγύτερους γείτονες n_i, αν το p βρεθεί μέσα σε κάποια από τις περιοχές ακύρωσης που ορίζονται σε σχέση με τα σημεία τομής της μεσοκαθέτου B_i και των δύο αξόνων που ορίζουν τον τομέα S_i.

Απόδειξη. Θεωρούμε ότι η σημειακή θέση p βρίσκεται έξω από την περιοχή ακύρωσης A_i που αντιστοιχεί στο σημείο βάσης b_i. Προφανώς, $|pb_i| > |b_i m_i|$. Σχηματίζουμε την πιο περιοριστική μεσοκάθετο B_j που αντιστοιχεί στο ευθύγραμμο τμήμα $\overline{pn_i}$, όπου n_i είναι ο εγγύτερος γείτονας ενδιαφέροντος. Εφόσον τα ευθύγραμμα τμήματα $\overline{m_i n_i}$ και $\overline{pn_i}$ τέμνονται στο σημείο n_i, οι αντίστοιχες μεσοκάθετοι B_i και B_j επίσης θα τέμνονται στο σημείο r.

Τώρα υποθέτουμε ότι το r βρίσκεται εντός του κυκλικού τομέα S_i. Εφόσον το r ανήκει και στις δύο μεσοκαθέτους B_i και B_j, ισχύει ότι $|rn_i| = |rm_i|$ και $|rn_i| = |rp|$, άρα τα σημεία n_i, m_i, p βρίσκονται πάνω στον ίδιο κύκλο. Έτσι η χορδή $\overline{n_i m_i}$ ανήκει στην περιφέρεια του μοναδικού κύκλου που περνάει επίσης από το p. Επομένως οι κύκλοι $O_1(b_i, |b_i m_i|)$ και $O_2(r, |rm_i|)$ τέμνονται και έχουν μια κοινή χορδή την $\overline{n_i m_i}$. Εφόσον $|rp| = |rn_i|$,$|rn_i| = |rm_i|$ και $|rm_i| < |b_i m_i|$, ένα τόξο a του κύκλου O_2 βρίσκεται εκτός του κύκλου O_1 ενώ το συμπληρωματικό του τόξο a' βρίσκεται εντός του O_1. Αλλά σε ποιο από τα δύο τόξα μπορεί να ανήκει το σημείο p; Προφανώς το p δεν μπορεί να ανήκει στο a' και άρα πρέπει να ανήκει στο a. Αλλά το τόξο αυτό βρίσκεται πλήρως εντός του κύκλου $O(q, |qm_i|)$, το οποίο σημαίνει ότι το p πρέπει να είναι το σημείο ελέγχου του κυκλικού τομέα S_i αντί για το m_i (άτοπο). □



Σχήμα 4.10: Χρονική απόκριση ανά χρονόσημο συναρτήσει (α) του N και για $\lambda = 8$ και (β) του λ και για $k = 5$

4.4 Πειραματικά αποτελέσματα

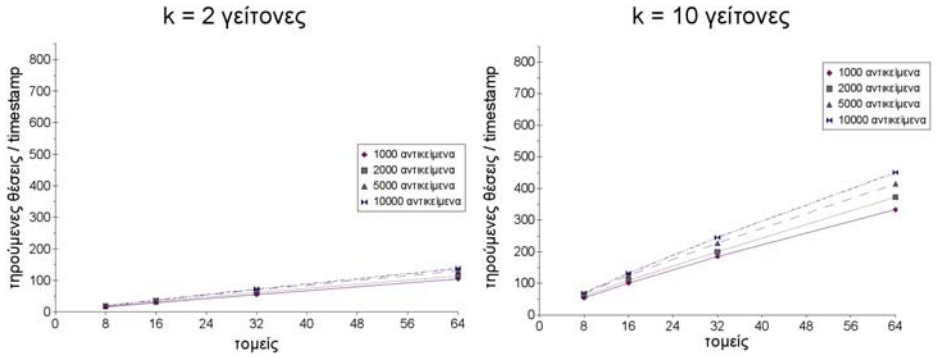
Στην ενότητα αυτή αξιολογούνται πειραματικά οι δύο μέθοδοι για την δημιουργία προσεγγιστικών κελιών *Voronoi* k τάξεως. Όλες οι δομές και οι αλγόριθμοι υλοποιήθηκαν σε γλώσσα C++ και τα πειράματα εκτελέστηκαν σε λειτουργικό σύστημα Ubuntu GNU/Linux 6.06 σε προσωπικό υπολογιστή Intel Celeron M 1600 (1.6 GHz) με μνήμη 266MHz.

4.4.1 Ταυτόχρονη ανανέωση σημειακών θέσεων

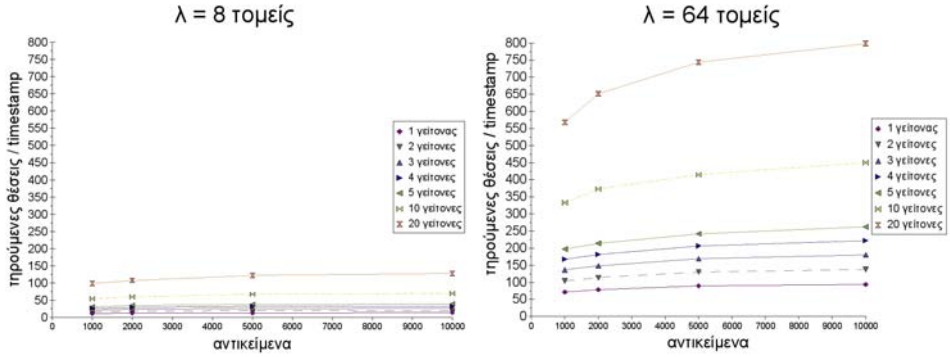
Για την παραλλαγή του αλγόριθμου κατασκευής προσεγγιστικού κελιού Voronoi, χρησιμοποιήθηκαν τέσσερα πειραματικά σύνολα δεδομένων αποτελούμενα από 1000, 2000, 5000 και 10000 κινούμενα αντικείμενα αντίστοιχα. Κάθε σύνολο περιέχει τις σημειακές θέσεις των αντικειμένων του για 200 χρονόσημα. Οι τροχιές των συνόλων προσομοιώνουν την κυκλοφοριακή κίνηση στο οδικό δίκτυο της Αθήνας. Για όλα τα αντικείμενα έχουμε μια διαφορετική σημειακή θέση σε κάθε χρονόσημο. Κάθε εγγραφή του συνόλου δεδομένων έχει την μορφή $< t, id, x, y >$ όπου t είναι το χρονόσημο αναφοράς, id η ταυτότητα του αντικειμένου και x, y οι συντεταγμένες της σημειακής του θέσης. Τα πειραματικά σύνολα των 1000, 2000 και 5000 αντικειμένων προέκυψαν από το σύνολο με τα 10000 αντικείμενα, επιλέγοντας με χρήση modulo αντικείμενα με βάση το id τους. Έγιναν πειράματα για τρεις παραμέτρους του αλγορίθμου:

- Πλήθος κινούμενων αντικειμένων N (1000, 2000, 5000, 10000).
- Αριθμός κυκλικών τομέων λ (8, 16, 32, 64).
- Αναζητούμενοι εγγύτεροι γείτονες k (1, 2, 3, 5, 10, 20).

Πρώτα μετρήθηκε ο χρόνος που χρειάζεται κατά μέσο όρο ο αλγόριθμος σε κάθε χρονόσημο για ένα σημείο ερωτήσεως, για να εισάγει τις σημειακές θέσεις των



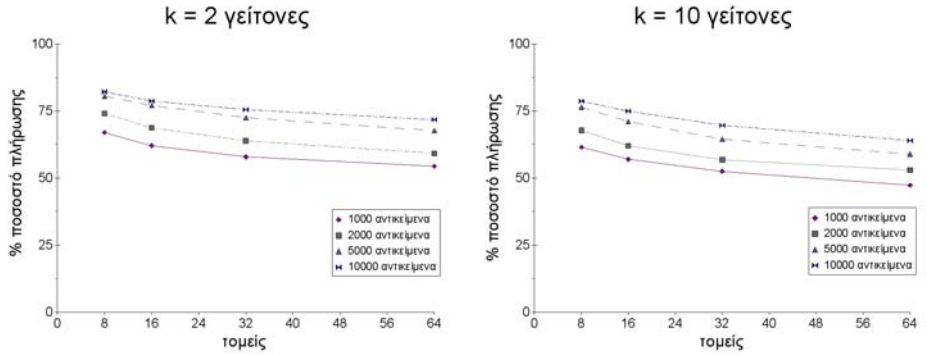
Σχήμα 4.11: Μέσος όρος αποθηκευμένων σημειακών θέσεων ανά χρονόσημο συναρτήσει του λ για (α) $k = 2$ και (β) για $k = 10$.



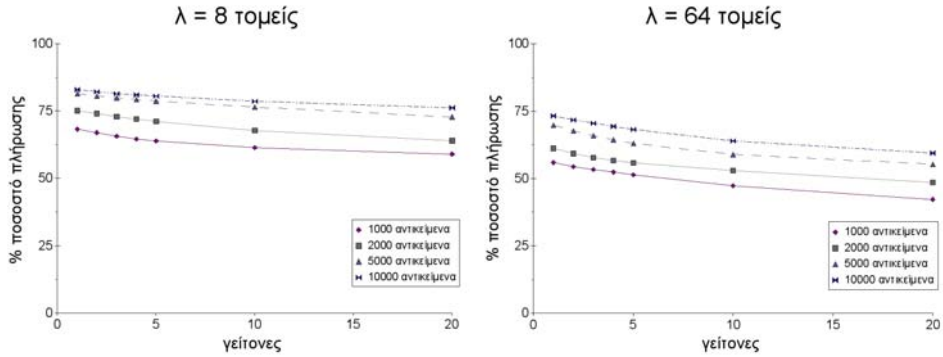
Σχήμα 4.12: Μέσος όρος αποθηκευμένων σημειακών θέσεων ανά χρονόσημο συναρτήσει του N για (α) $\lambda = 8$ και (β) για $\lambda = 64$

αντικειμένων στους κυκλικούς τομείς και για να κατασκευάσει το προσεγγιστικό κελί. Όπως φαίνεται στα σχήματα 4.10 (α) και (β) ο χρόνος αυτός βρέθηκε περίπου σταθερός για δεδομένο αριθμό κινούμενων αντικειμένων. Επομένως η χρονική απόκριση του συστήματος είναι ανεξάρτητη τόσο από τον αριθμό των κυκλικών τομέων λ , όσο και από τον αριθμό k των εγγύτερων γειτόνων. Πρακτικά είναι μια γραμμική συνάρτηση του πλήθους των αντικειμένων.

Γνωρίζοντας ότι ο θεωρητικά επιτρεπόμενος αποθηκευμένος αριθμός σημειακών θέσεων στο σύστημα είναι ίσος με $\lambda * (k + 1)$, μετρήθηκε ο πειραματικός μέσος όρος τους ανά χρονόσημο. Στα σχήματα 4.11(α) και (β) φαίνεται ότι η γραμμική σχέση που δίνει το θεωρητικό μέγιστο επιβεβαιώνεται, καθώς με αύξηση των τομέων παρατηρείται αύξηση και του αριθμού των αποθηκευμένων σημείων. Στα σχήματα 4.12(α) και (β) περιγράφεται η εξέλιξη του αριθμού των σημειακών θέσεων συναρτήσει του N, για δεδομένο αριθμό κυκλικών τομέων ($\lambda = 8, 64$ αντίστοιχα) και για διάφορες τιμές του k. Παρατηρούμε ότι ο αριθμός των αποθηκευμένων σημειακών θέσεων υπολείπεται σημαντικά του μέγιστου επιτρεπτού.



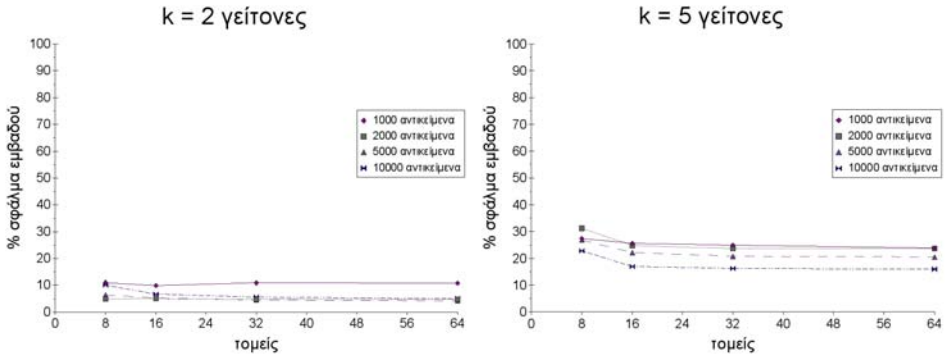
Σχήμα 4.13: Ποσοστό πλήρωσης συναρτήσει του λ για (α) $k = 2$ και (β) $k = 10$



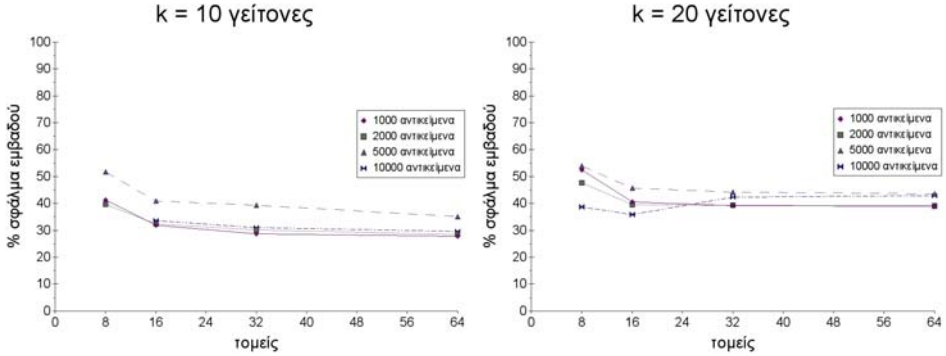
Σχήμα 4.14: Ποσοστό πλήρωσης συναρτήσει του k για (α) $\lambda = 8$ και (β) $\lambda = 64$

Αυτό οφείλεται στην δυνατότητα απόρριψης σημείων από τον αλγόριθμο. Το ποσοστό των σημείων που θα απορριφθεί όμως, εξαρτάται από την κατανομή τους και συγκεκριμένα την σχετική τους θέση σε σχέση με το σημείο ερωτήσεως. Στα σχήματα 4.13(α) και (β) φαίνεται η ποσοστιαία πληρότητα του συστήματος, δηλαδή το επί τοις εκατό ποσοστό του θεωρητικού μέγιστου που παρατηρείται στα πειραματικά αποτελέσματα. Λαμβάνοντας υπόψιν και τα σχήματα 4.14(α) και (β) συμπεραίνεται ότι το σύστημα έχει μια μέση πληρότητα $55 - 80\%$ για όλες τις περιπτώσεις.

Το πιο σημαντικό μέγεθος για τον αλγόριθμο είναι η ποιότητα της προσέγγισης του κελιού που επιτυγχάνει. Το μέγεθος που επιλέχθηκε για να υπολογιστεί αυτό, είναι η ποσοστιαία διαφορά του εμβαδού του προσεγγιστικού κελιού με αυτό του ακριβούς. Όπως φαίνεται και στα σχήματα 4.15(α) και (β) το ποσοστό σφάλματος κυμαίνεται μεταξύ $5 - 35\%$ για 2 και 5 εγγύτερους γείτονες, ενώ για μεγάλες τιμές του k (Σχήμα 4.16(α) και (β)) το ποσοστιαίο σφάλμα είναι μεταξύ $30 - 55\%$. Συμπεραίνεται ότι ο αριθμός των εγγύτερων γειτόνων είναι αυτός που επηρεάζει στον μεγαλύτερο βαθμό την ακρίβεια του προσεγγιστικού κελιού. Η αύξηση του k αυξάνει τον αριθμό των μεσοκαθέτων που παραλείπονται για κάθε σημείο ελέγχου,



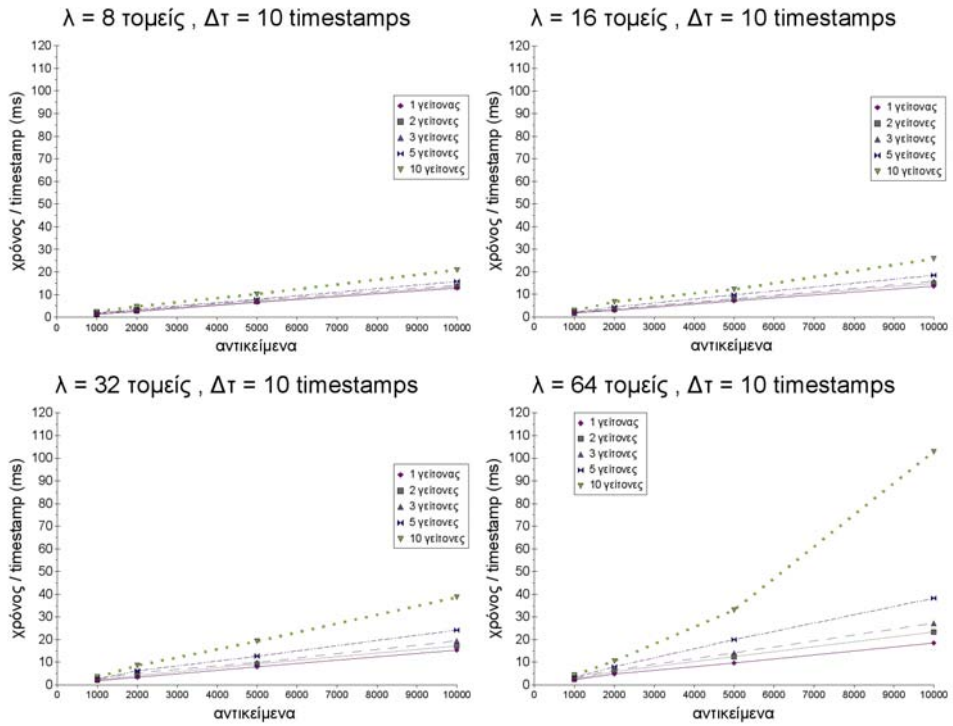
Σχήμα 4.15: Ποσοστιαίο σφάλμα εμβαδού συναρτήσει του λ για (α) $k = 2$ και (β) $k = 5$.



Σχήμα 4.16: Ποσοστιαίο σφάλμα εμβαδού συναρτήσει του λ για (α) $k = 10$ και (β) $k = 20$.

κατά την δημιουργία του κελιού. Τα δύο σχήματα δείχνουν καθαρά την αύξηση στο σφάλμα που έχουμε για μεγάλο αριθμό εγγύτερων γειτόνων. Για μικρές τιμές του k η αύξηση στον αριθμό των τομέων δεν επιφέρει πρακτική βελτίωση στην προσέγγιση του κελιού. Ουσιαστικά από τους 16 τομείς και μετά το αποτέλεσμα παραμένει αναλλοίωτο. Για μεγαλύτερες τιμές υπάρχει μείωση του σφάλματος για 32 τομείς και πρακτικά ασήμαντη για $\lambda = 64$.

Ο αριθμός των κινούμενων αντικειμένων N επηρεάζει την τιμή του σφάλματος μη ντετερμινιστικά. Για μεγαλύτερα N έχουμε διαφορετική κατανομή σημείων στους κυκλικούς τομείς και μεγαλύτερη πιθανότητα να υπάρχουν σημεία σε όλους τους τομείς. Αυτό συνεπάγεται ότι πιο πολλές μεσοκάθετοι συντελούν στην δημιουργία του προσεγγιστικού κελιού. Το συμπέρασμα αυτό όμως δεν είναι απόλυτο, αφού τα σχήματα 4.16(α) και (β) δείχνουν ότι για $N = 5000$ έχουμε το μεγαλύτερο σφάλμα. Το γεγονός αυτό οφείλεται στην τυχαία κατανομή των σημείων και τον μεγάλο αριθμό των εγγύτερων γειτόνων.

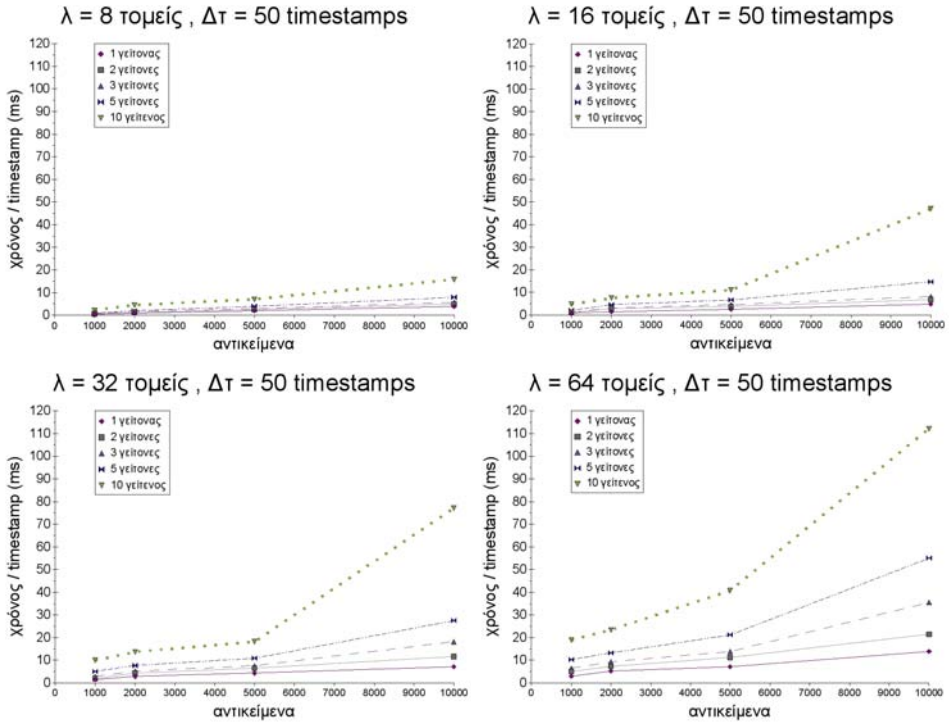


Σχήμα 4.17: Χρονική απόκριση ανά χρονόσημο συναρτήσει του N με $\Delta\tau = 10$,για (α) $\lambda = 8$, (β) $\lambda = 16$, (γ) $\lambda = 32$ και (δ) $\lambda = 64$

4.4.2 Μοντέλο χρονικά κυλιόμενου παραθύρου

Στο μοντέλο παραθύρου, τα τέσσερα σύνολα σημείων με 1000, 2000, 5000 και 10000 τροποποιήθηκαν για να περιγράφουν σημειακές θέσεις με διάρκεια ισχύος κυμαινόμενη 10, 30 και 50 χρονόσημα. Η έναρξη ισχύος κάθε αντικειμένου ορίστηκε με τυχαίο τρόπο σε ένα από τα πρώτα 10,30 ή 50 χρονόσημα. Με αυτόν τον τρόπο το προέκυψαν 12 σύνολα σημείων, που δεν ανανεώνουν τις σημειακές τους θέσεις ταυτόχρονα, αλλά όλα έχουν χρονική ισχύ ίση με το εκάστοτε παράθυρο. Η μορφή των σημειακών θέσεων είναι ίδια με αυτή των συνόλων που χρησιμοποιήθηκαν στο μοντέλο ταυτόχρονης ανανέωσης.Πραγματοποιήθηκαν πειραματικές μετρήσεις για τις εξής παραμέτρους:

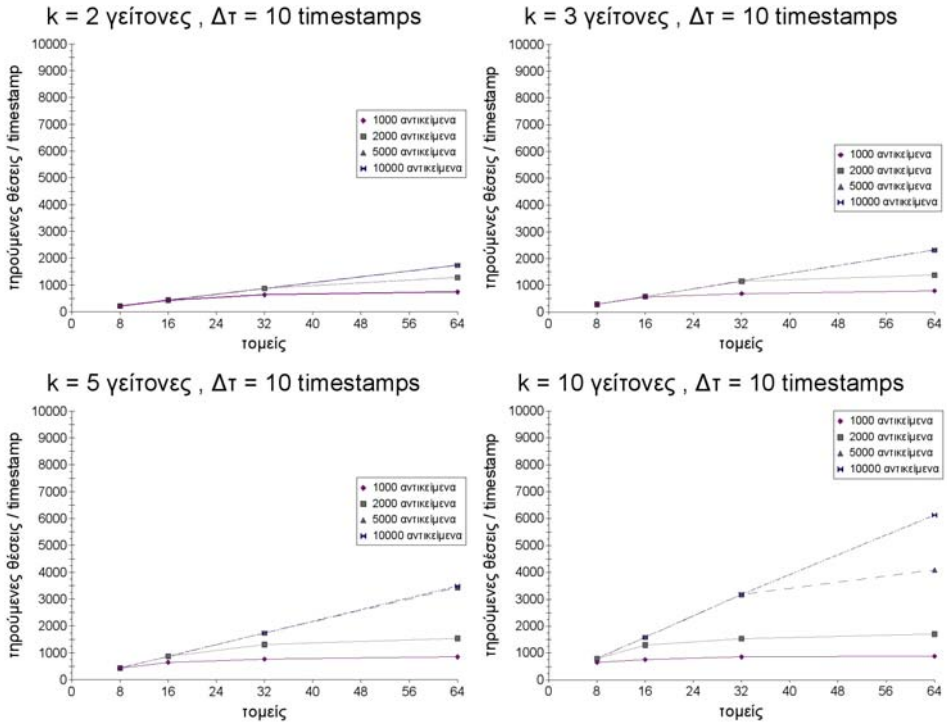
- Πλήθος κινούμενων αντικειμένων N (1000, 2000, 5000, 10000).
- Εύρος χρονικού παραθύρου $\Delta\tau$ (10, 30, 50).
- Αριθμός κυκλικών τομέων λ (8, 16, 32, 64).
- Αναζητούμενοι εγγύτεροι γείτονες k (1, 2, 3, 5, 10).



Σχήμα 4.18: Χρονική απόκριση ανά χρονόσημο συναρτήσει του N με $\Delta\tau = 50$,για (α) $\lambda = 8$, (β) $\lambda = 16$, (γ) $\lambda = 32$ και (δ) $\lambda = 64$

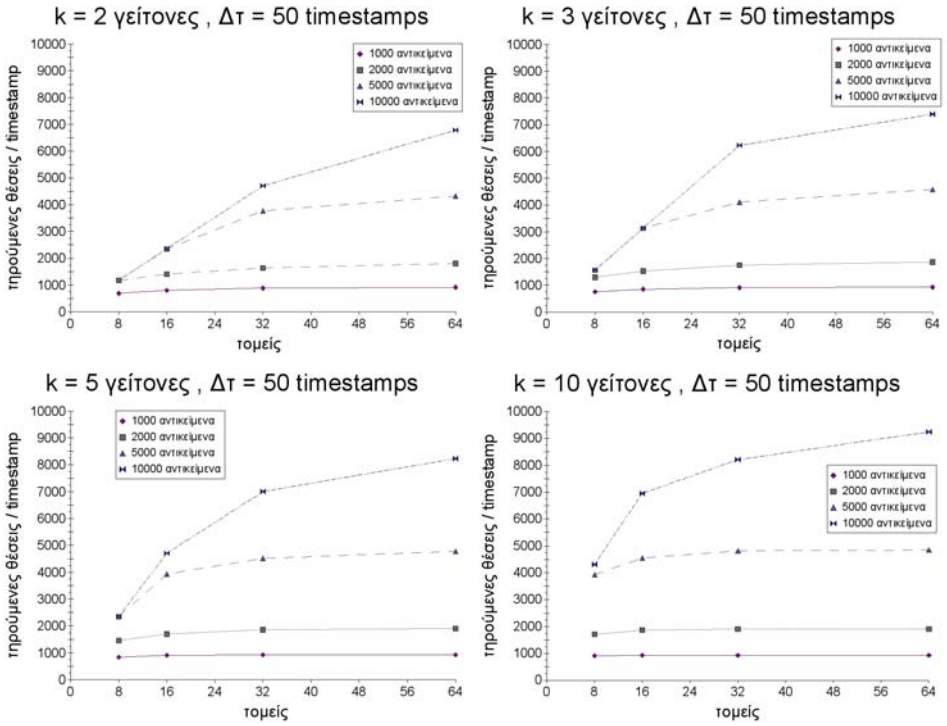
Η χρονική απόκριση του αλγόριθμου ανά χρονόσημο ήταν και πάλι το πρώτο μέγεθος που υπολογίσθηκε. Τα σχήματα 4.17 και 4.18 δίνουν την χρονική απόκριση για εύρος παραθύρου 10 και 50 αντίστοιχα. Η καμπύλη είναι γραμμική είτε ολόκληρη (4.17(β)) είτε για $N < 5000$. Η διαφορά αυτή παρατηρείται για μεγάλες τιμές των παραμέτρων του συστήματος. Για παράδειγμα για $\Delta\tau = 10, \lambda = 64, k = 10$ η χρονική απόκριση σχεδόν τριπλασιάζεται από $N = 5000$ σε $N = 10000$. Σε αυτές τις περιπτώσεις οι μεγάλες τιμές των παραμέτρων δίνουν την δυνατότητα στο σύστημα να αποθηκεύσει στους τομείς μεγάλο αριθμό σημειακών θέσεων. Έτσι η εισαγωγή των σημειακών θέσεων στους τομείς κοστίζει περισσότερο, αφού οι λίστες σημείων διατρέχονται σειριακά χωρίς κάποιου είδους ευρετήριο. Επίσης μεγαλύτερος αριθμός σημειακών θέσεων συνεπάγεται μεγαλύτερο κόστος στην αναζήτηση των αρχικών εγγύτερων γειτόνων στην φάση της αρχικοποίησης. Οι μέγιστες τιμές της χρονικής απόκρισης είναι 110 ms για $\Delta\tau = 10$ και 120 ms για $\Delta\tau = 50$ περίπου, που αντιστοιχούν σε ρυθμό ενημέρωσης 1000 αντικειμένων/χρονόσημο και 200 αντικειμένων/χρονόσημο αντίστοιχα.

Στη συνέχεια υπολογίσθηκε ο μέσος όρος των αποθηκευμένων σημειακών θέσεων στο σύστημα ανά χρονόσημο. Για $\Delta\tau = 10$ και $\Delta\tau = 50$ τα αποτελέσματα φαίνονται στα σχήματα 4.19 και 4.20. Παρατηρούμε και εδώ ότι όσο αυξάνονται



Σχήμα 4.19: Μέσος όρος αποθηκευμένων σημειακών θέσεων, συναρτήσει του λ με $\Delta\tau = 10$ για (α) $k = 2$, (β) $k = 3$, (γ) $k = 5$ και (γ) $k = 10$.

οι τιμές των παραμέτρων, οι καμπύλες τείνουν να γίνονται μη γραμμικές. Όπως και στο μοντέλο ταυτόχρονης ανανέωσης, ο αριθμός των αποθηκευμένων σημειακών θέσεων δεν είναι ποτέ ίσος με τον μέγιστο επιτρεπτό που σε κάθε περίπτωση υπολογίζεται ως $(k+1) * \Delta\tau * \lambda$. Ο λόγος είναι η απόρριψη και η αντικατάσταση σημειακών θέσεων κατά την διαδικασία της εισαγωγής. Το ποσοστό πλήρωσης των τομέων, εξαρτάται σημαντικά από την κατανομή των σημείων στο χώρο σε σχέση με την θέση του σημείου ερωτήσεως. Τα σχήματα 4.21 και 4.22 δείχνουν την διακύμανση του ποσοστού πλήρωσης των τομέων. Φαίνεται ότι για μικρές τιμές παραμέτρων η πληρότητα είναι μεγαλύτερη από 90%, ενώ όσο αυτές αυξάνονται το ποσοστό πέφτει. Η αύξηση των παραμέτρων συνεπάγεται μεγαλύτερο άνω όριο στον αριθμό των αποθηκευμένων σημειακών θέσεων. Συγκρίνοντας τώρα τα αντίστοιχα σχήματα μέσου όρου και ποσοστού πλήρωσης σημειακών θέσεων, βλέπουμε ότι οι γραμμικές καμπύλες αντιστοιχούν σε υψηλά ποσοστά πλήρωσης της τάξης του 90%. Καθώς με την αύξηση των τιμών των παραμέτρων το άνω όριο μεγαλώνει σε σχέση με το σύνολο αντικειμένων, το ποσοστό πλήρωσης πέφτει. Σε αυτές τις περιπτώσεις, το σύστημα μας παρέχει μεγαλύτερη αποθηκευτική ικανότητα από αυτή που χρησιμοποιούμε. Χαρακτηριστικά στο σχήμα 4.22(γ) το ποσοστό πλήρωσης είναι κάτω από 40% για $N = 10000$ και $\lambda = 64$.

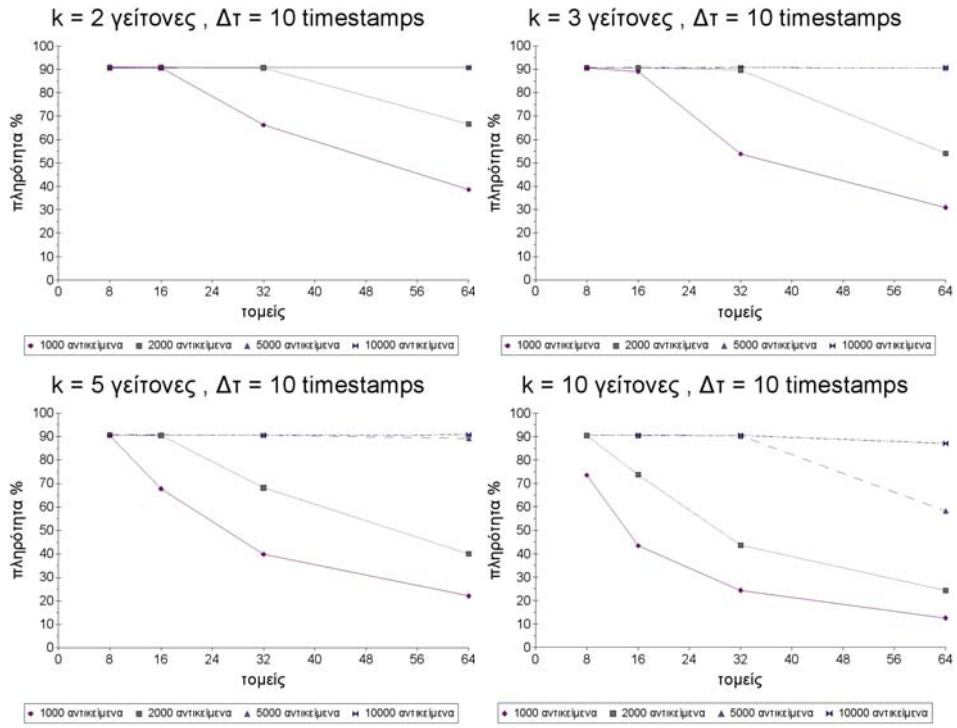


Σχήμα 4.20: Μέσος όρος αποθηκευμένων σημειακών θέσεων, συναρτήσει του λ με $\Delta\tau = 50$ για (α) $k = 1$, (β) $k = 3$, (γ) $k = 5$ και (δ) $k = 10$.

Από τα σχήματα 4.19(δ) και 4.20(γ) και (δ) δικαιολογείται η μεγάλη απόκλιση στην χρονική απόκριση που παρατηρήθηκε για $k = 10$ στις μεγάλες τιμές των παραμέτρων του συστήματος. Βλέπουμε ότι υπάρχει μια μεγάλη αύξηση στον αριθμό των σημειακών θέσεων που φτάνει ακόμα και το διπλάσιο. Η αύξηση αυτή επιφέρει και την αύξηση στην χρονική απόκριση για τους λόγους που αναφέρθηκαν πριν.

Και σε αυτή την περίπτωση, το πιο σημαντικό μέγεθος για την αξιολόγηση της μεθόδου που προτείνεται είναι το σφάλμα που υπεισέρχεται στον υπολογισμό του προσεγγιστικού κελιού σε σχέση με το ακριβές. Μετρήθηκε για αυτό το σκοπό το εκατοστιαίο μέσο σφάλμα στο εμβαδόν του προσεγγιστικού κελιού. Στο σχήμα 4.23 φαίνονται οι διακυμάνσεις του σφάλματος, για διάφορες τιμές του N και $\Delta\tau$. Κάθε γραφική παράσταση αντιστοιχεί σε ένα από τα 12 σύνολα σημείων που χρησιμοποιήθηκαν ως είσοδοι. Επειδή αυτά μεταξύ τους διαφέρουν είτε στο πλήθος των κινούμενων αντικειμένων, είτε στη χρονική ισχύ και στο εύρος του χρονικού παραθύρου, δεν είναι άμεσα συγκρίσιμα και άρα μόνο ποιοτικά συμπεράσματα μπορούν να εξαχθούν από την αντιπαραβολή τους.

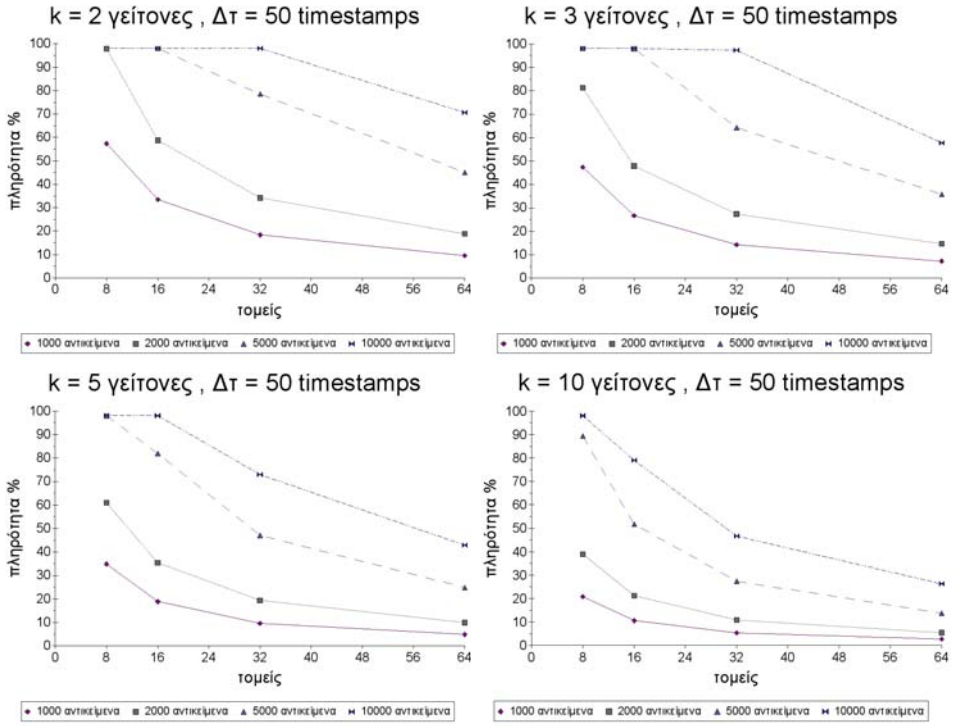
Στα διαγράμματα φαίνεται ότι το μέγιστο σφάλμα που παρατηρείται είναι περίπου 50% για μεγάλο αριθμό εγγύτερων γειτόνων ($k = 10$). Σε επίπεδο εφαρμογών



Σχήμα 4.21: Ποσοστό πλήρωσης συναρτήσει του λ με $\Delta\tau = 10$ για (α) $k = 2$, (β) $k = 3$, (γ) $k = 5$ και (δ) $k = 10$.

μπορούμε να υποθέσουμε ότι ένας μικρότερος αριθμός ($k < 5$) είναι πιο συνηθισμένος και σε αυτήν την περίπτωση ο αλγόριθμος δίνει ένα μέσο σφάλμα που κυμαίνεται μεταξύ $10 - 40\%$. Είναι απόλυτα φυσιολογικό ότι όσο αυξάνεται το k αυξάνεται και το σφάλμα στον υπολογισμό, αφού ο αλγόριθμος παραλείπει περισσότερες μεσοκάθετους κατά την δημιουργία του προσεγγιστικού κελιού. Όπως φαίνεται όμως στο σχήμα 4.23 για $N = 10000, \Delta\tau = 50$ αυτό δεν είναι απόλυτο καθώς και η κατανομή των σημειακών θέσεων στο επίπεδο σε σχέση με το σημείο ερωτήσεως παίζει ρόλο στο σφάλμα του κελιού.

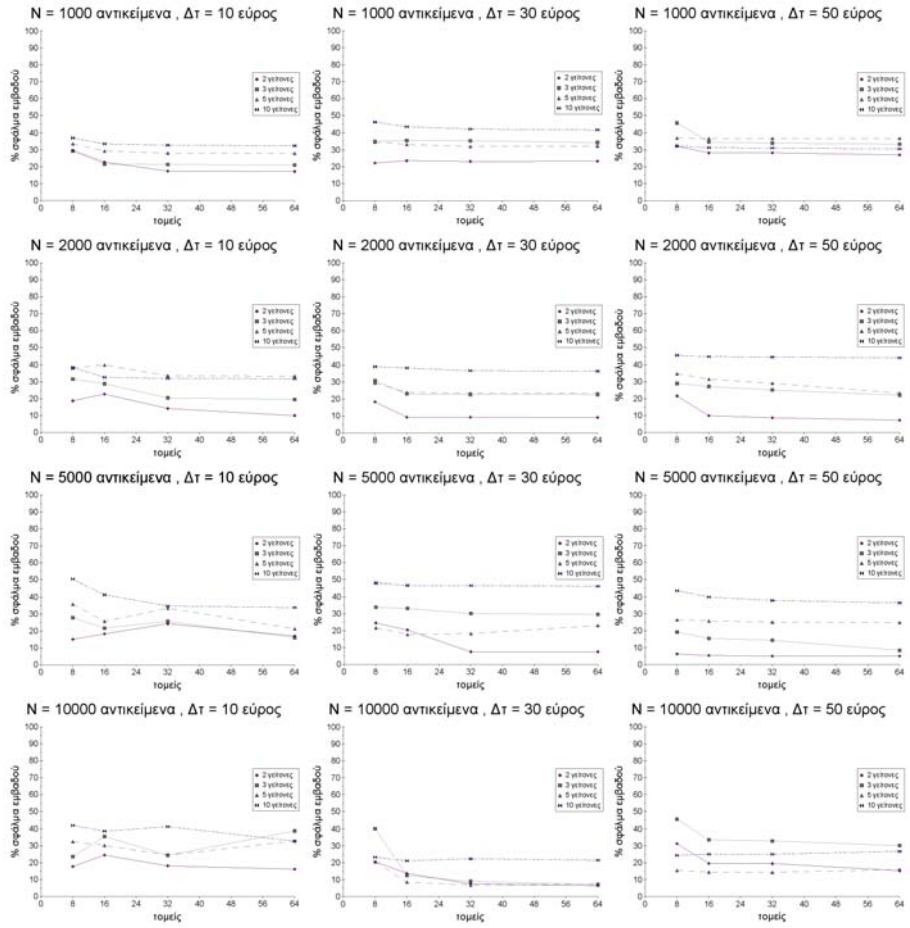
Θα πρέπει να σημειωθεί ότι ο αλγόριθμος όπως σχεδιάσθηκε και υλοποιήθηκε, σε κάποιες περιπτώσεις δημιουργεί προσεγγιστικά κελιά με εμβαδό έως και 99% μεγαλύτερο από τα αντίστοιχα ακριβή. Αυτό δεν αποτελεί παράλειψη ή σφάλμα, αλλά μια προβληματική κατάσταση εγγενή στον αλγόριθμο. Η περίπτωση αυτή συμβαίνει όταν σε έναν κυκλικό τομέα περιλαμβάνεται μόνο ένα σημείο (που αποτελεί και σημείο ελέγχου) του οποίου η χρονική ισχύς παρέρχεται και κανένα νέο σημείο δεν εισέρχεται στον τομέα αυτό. Ο αλγόριθμος τότε το αφαιρεί από τον τομέα, μαζί με την πλευρά του κελιού που πιθανώς δημιουργούσε η μεσοκάθετός του. Αν όντως υπήρχε μια τέτοια πλευρά στο κελί, και η επόμενη και η προηγούμενη πλευρά του δεν συγκλίνουν τότε έχουμε περίπτωση *ανοιχτού κελιού* (σχήμα 4.24(α)). Φυσικά



Σχήμα 4.22: Ποσοστό πλήρωσης συναρτήσει του λ με $\Delta\tau = 50$ για (α) $k = 2$, (β) $k = 3$, (γ) $k = 5$ και (δ) $k = 10$.

για λόγους απλότητας το κελί δεν παραμένει ανοιχτό πολύγωνο, αλλά κλείνει στην τομή του με το περιβάλλον ορθογώνιο (σχήμα 4.24(β)). Το περιβάλλον ορθογώνιο όμως περικλείει τον χώρο κίνησης όλων των αντικειμένων, με συνέπεια το κελί που προκύπτει να έχει εμβαδό κατά πολύ μεγαλύτερο από αυτό του προηγούμενου βήματος. Στο αντίστοιχο χρονόσημο, το ακριβές κελί που κατασκευάζεται με χρήση όλων των μεσοκαθέτων είναι πολύ πιθανό να μην είναι ανοιχτό. Άρα η μεταξύ τους διαφορά σε εμβαδό θα είναι πολύ μεγάλη. Η εμφάνιση του φαινομένου των ανοιχτών κελιών είναι τυχαία και δεν μπορεί να προβλεφτεί για κάποιο σύνολο αντικειμένων, καθιστώντας την κατανομή των σημειακών θέσεων στο χώρο σημαντικό παράγοντα στην ακρίβεια των προσεγγιστικών κελιών.

Παρατηρώντας τα διαγράμματα βλέπει κανείς την μικρή επίδραση που έχει η αύξηση του αριθμού των τομέων σε $\lambda = 64$ στις περισσότερες περιπτώσεις. Αυτό σε συνδυασμό με την σημαντική επιβάρυνση στην χρονική απόκριση που περιγράφηκε πριν, οδηγεί στο συμπέρασμα ότι για $k = 32$ το σύστημα δίνει μια ικανοποιητικής ακρίβειας λύση χωρίς υπερβολικό φόρτο στο σύστημα. Υπάρχουν και περιπτώσεις όπου και η μετάβαση από τους 16 στους 32 τομείς δεν απέδωσε ιδιαίτερη βελτίωση στην προσέγγιση, αλλά το επιπλέον κόστος είναι κατά πολύ μικρότερο και άρα μπορεί να δικαιολογηθεί.

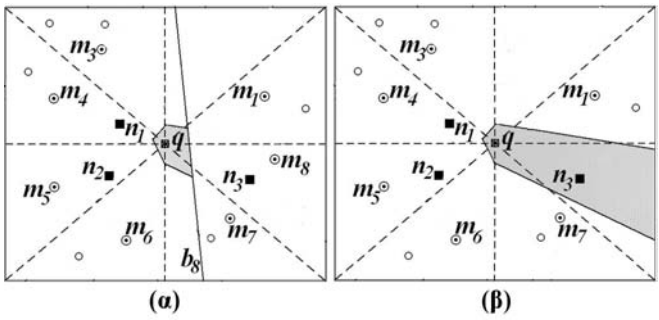


Σχήμα 4.23: Επί τοις εκατό ποσοστό σφάλματος στον υπολογισμό του προσεγγιστικού κελιού συναρτήσει του αριθμού των κυκλικών τομέων

Τέλος μετρήθηκε ο χρόνος που χρειάζεται ο αλγόριθμος για να χειριστεί την αλλαγή της σημειακής θέσης του σημείου ερωτήσεως. Η αλλαγή θεωρήθηκε ότι εκτελείται σε τέσσερις διακριτές φάσεις:

1. Στην συλλογή όλων των ενεργών σημειακών θέσεων σε μια λίστα.
2. Στην καταστροφή των δομών υποστήριξης (τρέχον κελί, λίστα εγγύτερων γειτόνων).
3. Στην αναδιανομή των σημείων.
4. Στην δημιουργία του προσεγγιστικού κελιού με ολίσθηση του χρόνου.

Μετά το τέλος των φάσεων αυτών το σύστημα δέχεται τις σημειακές θέσεις του νέου χρονόσημου. Οι μετρήσεις έδειξαν ότι η δεύτερη και η τέταρτη φάση



Σχήμα 4.24: Πρόβλημα ανοιχτού κελιού : (α) Ακύρωση σημείου ελέγχου σε τομέα (β) Τεχνητό κλείσιμο κελιού στα όρια της περιοχής μελέτης.

έχουν αμελητέα συνεισφορά στην χρονική απόκριση. Η φάση της αναδιανομής είναι συντριπτικά η πλέον χρονοβόρα (περίπου 90% της όλης διαδικασίας) και μόνο όταν έχουμε μεγάλο αριθμό αποθηκευμένων αντικειμένων στο σύστημα συνεισφέρει ουσιαστικά και η φάση της συλλογής αποτελώντας το 25% του κόστους.

Ο κύριος παράγοντας που επηρεάζει την χρονική απόκριση είναι ο αριθμός των αποθηκευμένων αντικειμένων. Οι διαφοροποιήσεις των $k, \lambda, \Delta\tau$ μεταβάλλουν αυτόν τον αριθμό και άρα έχουμε διαφορετικές αποκρίσεις. Για μεγάλο αριθμό σημείων που πλησιάζουν το μέγιστο των συνόλων που δοκιμάστηκαν, ο χρόνος αυτός πλησίασε το ένα δευτερόλεπτο σε ορισμένες περιπτώσεις. Για παράδειγμα για $k = 10, \lambda = 64, \Delta\tau = 50$ και $N = 10000$ η διαδικασία αλλαγής του Q με 8098 ενεργά αντικείμενα στο σύστημα κράτησε 945 ms. Το 20% αυτού του χρόνου το απαίτησε η συλλογή των σημείων, το 68% η αναδιανομή, ενώ μόλις το 2% αυτού αποτέλεσε η δημιουργία του κελιού και η ολίσθηση του χρόνου. Σε τέτοιες περιπτώσεις ο χειρισμός της αλλαγής της σημειακής θέσης του Q είναι αρκετά χρονοβόρος σε σχέση και με την απόκριση ολόκληρου του συστήματος. Για τις τιμές των παραμέτρων που αναφέρθηκαν, σε διάστημα 140 χρονοσήμων και έχοντας 14 ανανεώσεις σημειακών θέσεων του Q, ο συνολικός χρόνος του χειρισμού της μετακίνησης του Q αποτέλεσε το 64% του συνολικού χρόνου εκτέλεσης του προγράμματος. Φυσικά το ποσοστό αυτό για λιγότερες αποθηκευμένες θέσεις μειώνεται. Οι φάσεις της συλλογής και της αναδιανομής επηρεάζονται και από τον αριθμό των κυκλικών τομέων αν και κύριος παράγοντας επιρροής παραμένει ο αριθμός των ενεργών σημειακών θέσεων.

Η αρκετά σημαντική επιβάρυνση που επιφέρει ο χειρισμός της αλλαγής στην σημειακή θέση του Q δικαιολογείται από το γεγονός ότι ο αλγόριθμος βασίζεται εκ θεμελίων σε μια κατάτμηση του χώρου που εξαρτάται από την συγκεκριμένη σημειακή θέση ερωτήσεως. Ο χειρισμός της αλλαγής της δεν μπορεί παρά να αποτελεί μια επανεκκίνηση του αλγόριθμου με διατήρηση των ενεργών σημειακών θέσεων.

4.5 Αξιολόγηση της μεθόδου

Αλγόριθμος ταυτόχρονης ανανέωσης

Αξιολογώντας την μέθοδο υπολογισμού προσεγγιστικών κελιών *Voronoi* k-τάξεως με βάση τις πειραματικές μετρήσεις, μπορούμε να ισχυριστούμε ότι παρέχει ικανοποιητικά αποτελέσματα για διαχείριση αρκετά μεγάλου πλήθους κινούμενων αντικειμένων. Αν και η ταυτόχρονη ανανέωση των σημειακών θέσεων όλων των αντικειμένων αποτελεί ένα σενάριο που δεν απαντάται συνήθως σε πραγματικές εφαρμογές, μπορεί να δώσει χρήσιμα συμπεράσματα για την αξιολόγηση του αλγορίθμου. Η δε κατανομή των αντικειμένων που χρησιμοποιήθηκαν ως είσοδος (κίνηση στόλου οχημάτων στο οδικό δίκτυο της Αθήνας), προσδίδει ιδιαίτερη χρηστική αξία στα συμπεράσματα αυτά, αφού τα καθιστά εφαρμόσιμα σε ένα πραγματικό πλαίσιο.

Η κύρια παράμετρος ενδιαφέροντος είναι η ποιότητα της προσέγγισης που επιτυγχάνεται, η οποία από τις πειραματικές μετρήσεις προκύπτει ότι εξαρτάται σε σημαντικό βαθμό από την παραμετροποίηση του αλγορίθμου. Οι βασικές παράμετροι του συστήματος είναι ο αριθμός των κυκλικών τομέων λ και ο αριθμός των εγγύτερων γειτόνων k που αναζητούνται. Διαπιστώθηκε ότι όσο μεγαλώνει το k το σφάλμα προσέγγισης του κελιού αυξάνει όπως άλλωστε αναμενόταν. Αν θεωρηθεί ότι σε πραγματικές εφαρμογές η τιμή του k κυμαίνεται έως 5, το σφάλμα στην προσέγγιση είναι κάτω από 30%. Στο πλαίσιο των εφαρμογών πραγματικού χρόνου, όπου ο ρυθμός εισαγωγής είναι συνήθως καταιγιστικός το ποσοστό αυτό κρίνεται αρκετά ικανοποιητικό. Ακόμα και για μεγαλύτερες τιμές του k το σφάλμα δεν αυξάνεται υπερβολικά φτάνοντας για $k = 20$ το 50%, μια περίπτωση που δεν μπορεί εύκολα να χαρακτηρισθεί συνήθης.

Ο αριθμός των κυκλικών τομέων λ επηρεάζει την ακρίβεια του κελιού μέχρι την τιμή $\lambda = 16$. Παρατηρήθηκε ότι περαιτέρω αύξηση δεν επιφέρει ουσιαστική βελτίωση στην προσέγγιση, ενώ αντίθετα δημιουργεί αυξημένες απαιτήσεις στο πλήθος των αποθηκευμένων σημειακών θέσεων χωρίς πρακτικό όφελος. Μόνο στις περιπτώσεις όπου και το k είχε τιμή μεγαλύτερη του 10 παρατηρήθηκε μικρή βελτίωση στην προσέγγιση για $\lambda > 16$. Το όφελος αυτό όμως αντισταθμίζεται από τον επιπλέον επεξεργαστικό φόρτο που απαιτείται.

Εκτός από τις παραμέτρους που επιλέγονται από τον χρήστη, η αποδοτικότητα του αλγόριθμου παρουσιάζει εξάρτηση από την κατανομή των αντικειμένων στον χώρο σε σχέση με το σημείο ερωτήσεως. Όσο πιο ομοιόμορφη είναι η κατανομή τους στο σύνολο των κυκλικών τομέων, τόσο βελτιώνεται η ποιότητα της προσέγγισης του κελιού. Φυσικά αυτό δεν είναι δυνατό να προβλεφτεί, ειδικά όταν έχουμε και κινούμενο σημείο ερωτήσεως και άρα η σχετική θέση των αντικείμενων μεταβάλλεται συνεχώς. Το πρόβλημα των ανοιχτών κελιών είναι αποτέλεσμα της κατανομής των αντικειμένων στο χώρο και συμβάλλει αποφασιστικά στην ποιότητα της προσέγγισης.

Σε γενικές γραμμές ο αλγόριθμος όπως περιγράφηκε, ανταποκρίνεται πολύ ικανοποιητικά με $\lambda = 16$ για τιμές του k που φθάνουν μέχρι το 20. Βεβαίως για υψηλές τιμές αναμένεται αρκετά μεγάλο σφάλμα στην προσέγγιση, αλλά οι χρηστικές περιπτώσεις αναζήτησης τόσο μεγάλου πλήθους εγγύτερων γειτόνων είναι

πιο σπάνιες. Με αυτές τις παραμέτρους ο αλγόριθμος μπορεί να ανταπεξέλθει σε σύνολα κινούμενων αντικειμένων διαφόρων μεγεθών.

Αλγόριθμος χρονικά κυλιόμενου παραθύρου

Η επέκταση του αλγορίθμου με την εφαρμογή ενός χρονικά κυλιόμενου παραθύρου στα δεδομένα, αποτελεί μια προσαρμογή του σε συνθήκες που πλησιάζουν περισσότερο αυτές μιας πραγματικής εφαρμογής. Χρησιμοποιώντας και πάλι σύνολα σημειακών θέσεων που περιγράφουν την κίνηση ενός στόλου οχημάτων στο οδικό δίκτυο της Αθήνας, δίνεται μια πολύ πρακτική εικόνα της απόδοσης του αλγόριθμου σε πραγματικές συνθήκες. Υπό αυτό το πρίσμα τα αποτελέσματα των πειραματικών μετρήσεων μπορούν να χαρακτηρισθούν πολύ ενθαρρυντικά, δίνοντας σε χρόνους εφαρμόσιμους σε εφαρμογές πραγματικού χρόνου, καλή προσέγγιση για αρκετά μεγάλα σύνολα δεδομένων.

Οι παρατηρήσεις για τον αλγόριθμο ταυτόχρονης ανανέωσης ισχύουν στην πλειονότητά τους και για το μοντέλο παραθύρου. Στο μοντέλο παραθύρου υπάρχει μια επιπλέον παράμετρος, αυτή του εύρους του χρονικού παραθύρου $\Delta\tau$ που ταυτόχρονα αποτελεί και το χρόνο ισχύος κάθε σημειακής θέσης. Εφόσον το σύνολο των N σημείων κατανέμεται στο χρονικό εύρος $\Delta\tau$ σε κάθε χρονόσημο ο αλγόριθμος χρειάζεται να επεξεργασθεί λιγότερες σημειακές θέσεις. Αυτό καθιστά δυνατό τον χειρισμό συνόλων με μεγαλύτερο αριθμό κινούμενων αντικειμένων από αυτά που δοκιμάσθηκαν.

Το σφάλμα στην προσέγγιση του κελιού *Voronoi* k-τάξεως που προκύπτει για τα διάφορα σύνολα δεδομένων, δείχνει σαφώς ότι η μέθοδος μπορεί να χαρακτηριστεί αποδοτική στο πλαίσιο των εφαρμογών πραγματικού χρόνου. Το σφάλμα για $k < 5$ κυμαίνεται στο διάστημα $10 - 40\%$, τιμή σαφώς μέσα στα πλαίσια του αποδεκτού εφόσον πρωτεύουσα σημασία αποκτά η γρήγορη απόκριση. Ακόμα και για $k = 10$ το σφάλμα δεν αυξάνεται πέραν του 50%.

Για τιμές της παραμέτρου λ μεγαλύτερες του 32 δεν παρατηρείται ουσιαστική βελτίωση στο σφάλμα, οπότε η τιμή αυτή θεωρείται μια καλή επιλογή για την αντιμετώπιση της πλειοψηφίας των περιπτώσεων. Το εύρος του παραθύρου $\Delta\tau$ μαζί με το πλήθος των αντικειμένων N αποτελούν εξωγενείς παράγοντες που καθορίζονται από το σύνολο αντικειμένων που δρα σαν είσοδος στο σύστημα. Η κύρια επίδραση του $\Delta\tau$ είναι στον αριθμό των σημειακών θέσεων που αποθηκεύει το σύστημα σε κάθε χρονόσημο. Η τιμή του μάλιστα καθορίζει και το άνω όριο στον αριθμό αυτό με συνέπεια για μεγάλες τιμές του $\Delta\tau$ να παρατηρείται πολύ μικρή πληρότητα στις πειραματικές μετρήσεις.

Συνολικά εφόσον οι τιμές των $\Delta\tau$ και N καθορίζονται από το ρεύμα δεδομένων εισόδου, θέτοντας τον αριθμό των κυκλικών τομέων στο $\lambda = 32$, ο αλγόριθμος μπορεί να επιστρέψει πολύ ικανοποιητικής ακρίβειας αποτελέσματα με πολύ καλή χρονική απόκριση.

Κεφάλαιο 5

Συμπεράσματα

5.1 Αξιολόγηση τεχνικών

Η αποδοτική μοντελοποίηση χωρικών προβλημάτων αποτελεί ένα ανοιχτό και ιδιαίτερα ενεργό ερευνητικό πεδίο, που αποσκοπεί στην ικανοποίηση των συνεχώς αυξανόμενων αναγκών για έξυπνες και κομψές λύσεις σε γεωμετρικά υπολογιστικά προβλήματα. Παράλληλα, η έκρηξη στον τομέα των τηλεπικοινωνιών έκανε προσιτή σε μια μεγάλη μάζα χρηστών τεχνολογία που καθιστά εύκολο και άμεσο τον γεωγραφικό εντοπισμό σημείων ενδιαφέροντος κατά βούληση. Το λογισμικό υπόβαθρο αυτών των υπηρεσιών οφείλει να εκμεταλλεύεται κατά το δοκούν αλγοριθμικές λύσεις από όλα τα συναφή επιστημονικά πεδία, όπως η υπολογιστική γεωμετρία, και να αποφεύγει τους περιορισμούς που φυσιολογικά επιβάλει το υλικό υπόβαθρο, αξιοποιώντας έξυπνα τους διαθέσιμους πόρους.

Ειδικά σε συστήματα αποτελούμενα από μεγάλο αριθμό ανεξάρτητων κόμβων που αποστέλλουν δεδομένα, η είσοδος που καλείται να διαχειριστεί το κεντρικό υποσύστημα είναι συνήθως ένα μεγάλου όγκου και καταιγιστικού ρυθμού ρεύμα δεδομένων. Οι απαιτήσεις στην επεξεργασία ενός τέτοιου συνόλου δεδομένων διαφέρουν ριζικά από αυτές ενός συμβατικού συστήματος διαχείρισης δεδομένων, όπου η ακρίβεια στην απάντηση είναι το κύριο ζητούμενο. Τα συστήματα διαχείρισης ρευμάτων δεδομένων αναπτύχθηκαν για να εκπληρώσουν αυτό το κενό, αποσκοπώντας πρωτίστως στην ικανοποίηση της απαίτησης για απόκριση σε πραγματικό χρόνο. Η έρευνα στον τομέα έχει δώσει ικανοποιητικά αποτελέσματα, προτείνοντας λύσεις που ήδη έχουν βρει μεγάλη απήχηση. Ωστόσο το πεδίο παραμένει ανοιχτό με πολλά προβλήματα προς επίλυση.

Ειδικότερα σε εφαρμογές εντοπισμού γεωγραφικής θέσης, ένα από τα συνηθέστερα ερωτήματα που υποβάλλεται είναι αυτό της εύρεσης των k εγγύτερων γειτόνων. Αντικείμενο αυτής της διπλωματικής αποτέλεσε η μελέτη των ερωτημάτων εγγύτερων γειτόνων τόσο για στατικά όσο και για δυναμικά κινούμενα αντικείμενα που καταφθάνουν με την μορφή ρεύματος δεδομένων. Η μελέτη των διαγραμμάτων *Voronoi* στατικών αντικειμένων, αποτέλεσε χρήσιμο υπόβαθρο στην κατανόηση της εφαρμογής γεωμετρικών αλγορίθμων στο πεδίο των βάσεων δεδομένων. Ο αλ-

γόριθμος του *Fortune* αξιοποιήθηκε για την κατασκευή του διαγράμματος *Voronoi* ενός συνόλου στατικών αντικειμένων και την απάντηση ερωτημάτων εγγύτερου γείτονα για ένα ρεύμα κινούμενων αντικειμένων επί αυτού. Από την πειραματική αξιολόγηση έγινε φανερή η χρηστική αξία των γεωμετρικών αλγορίθμων και η ευκολία χειρισμού χωρικών δεδομένων που επιτυγχάνουν.

Στη συνέχεια, με βάση τις ιδιότητες των διαγραμμάτων *Voronoi* k τάξεως, δημιουργήθηκε πρωτότυπος αλγόριθμος για την δημιουργία μεμονωμένων προσεγγιστικών κελιών *Voronoi* k τάξεως για ένα σημείο ερωτήσεως, με βάση ένα σύνολο δυναμικά κινούμενων αντικειμένων επεκτείνοντας παλιότερες εργασίες.

Ο αλγόριθμος που προτείνεται παράγει το προσεγγιστικό κελί *Voronoi* k τάξεως για ένα σημείο ερωτήσεως δημιουργώντας μια κατάτμηση του χώρου σε κυκλικούς τομείς γύρω από το σημείο ερωτήσεως. Η κατάτμηση αξιοποιείται κατάλληλα για την επεξεργασία ενός ρεύματος δυναμικά κινούμενων αντικειμένων που δρα ως είσοδος. Ανάλογα με την μορφή του ρεύματος εισόδου σχεδιάστηκαν δύο παραλλαγές του αλγορίθμου, μία για ταυτόχρονη ανανέωση όλων των σημειακών θέσεων των κινούμενων αντικειμένων και μια για ασύγχρονη ανανέωση με εφαρμογή κυλιόμενου χρονικού παραθύρου.

Στην περίπτωση της ταυτόχρονης ανανέωσης ο αλγόριθμος κατασκευάζει το προσεγγιστικό κελί Voronoi k-τάξεως σε κάθε χρονόσημο, για τις σημειακές θέσεις των αντικειμένων που είναι ενεργές. Αν και το σενάριο λειτουργίας του δεν είναι ιδιαίτερα αποδοτικό σε πρακτικές εφαρμογές, ο αλγόριθμος αποτέλεσε την βάση για την επέκταση στο μοντέλο κυλιόμενου παραθύρου και κατέδειξε την αποτελεσματικότητα της μεθόδου που προτείνεται.

Θεωρώντας ότι τα κινούμενα αντικείμενα ανανεώνουν ασύγχρονα τις σημειακές τους θέσεις, ο αλγόριθμος δοκιμάστηκε σε ένα πραγματικό πλαίσιο εφαρμογής. Η χρήση συνόλων δεδομένων που περιγράφουν την κίνηση ενός στόλου οχημάτων σε οδικό δίκτυο, δίνει ιδιαίτερα σημαντικά αποτελέσματα και επιβεβαιώνει ότι ο αλγόριθμος που προτείνεται μπορεί να εφαρμοσθεί σε πρακτικές εφαρμογές.

Οι πειραματικές μετρήσεις έδωσαν ικανοποιητικά αποτελέσματα με τους χρόνους απόκρισης να κυμαίνονται σε επίπεδα αξιοποιήσιμα για εφαρμογές πραγματικού χρόνου. Αξιολογώντας τα αποτελέσματα επιλέχθηκαν τιμές για την παράμετρο λ του συστήματος (ο αριθμός των κυκλικών τομέων) που σε κάθε περίπτωση δίνουν ικανοποιητική αναλογία απόδοσης/ακρίβειας. Συγκεκριμένα στην περίπτωση της ταυτόχρονης ανανέωσης προτείνεται η τιμή $\lambda = 16$ ενώ για το μοντέλο κυλιόμενου παραθύρου $\lambda = 32$. Οι υπόλοιπες παράμετροι θεωρούνται εξωγενείς που καθορίζονται από τον χρήστη. Ο αριθμός των εγγύτερων γειτόνων k εξαρτάται από το ερώτημα που θέτει ο χρήστης, ενώ το εύρος του παραθύρου $\Delta\tau$ στο μοντέλο παραθύρου επιλέγεται από αυτόν, καθορίζοντας ταυτόχρονα και το σύνολο δεδομένων που δρα ως ρεύμα εισόδου. Η τιμή του k έχει μεγάλη επίδραση στο σφάλμα προσέγγισης του κελιού, γεγονός που προβλεπόταν από την σχεδίαση του αλγόριθμου και επιβεβαιώθηκε από τις πειραματικές μετρήσεις. Για μικρές τιμές όμως λ.χ. $k <= 5$, το σφάλμα κυμαίνεται σε ικανοποιητικά επίπεδα. Η παράμετρος $\Delta\tau$ και ο αριθμός N των κινούμενων αντικειμένων του ρεύματος επηρεάζουν μη ντετερμινιστικά την προσέγγιση, με παράγοντα επιρροής την κατανομή των σημείων στο χώρο και όχι τις τιμές των $\Delta\tau$ και N.

Συνολικά μπορούμε να πούμε ότι ο αλγόριθμος που προτείνεται δίνει μια κομψή

και εύκολα υλοποιήσιμη λύση στο πρόβλημα αναζήτησης εγγύτερου γείτονα. Ο υπολογισμός των προσεγγιστικών κελιών την καθιστά κατάλληλη για προβλήματα πραγματικού χρόνου, όπου η ακρίβεια στην απάντηση μπορεί να θυσιαστεί για χάρη της γρήγορης απόκρισης.

5.2 Πιθανές επεκτάσεις

Κατά την σχεδίαση του αλγόριθμου ταυτόχρονης ανανέωσης ακολουθήθηκε μια διαδικασία απλοποίησης κατά την εισαγωγή των σημειακών θέσεων στους κυκλικούς τομείς και κατά την αναζήτηση των k εγγύτερων γειτόνων . Και στις δύο περιπτώσεις γίνεται επεξεργασία του συνόλου των σημειακών θέσεων με συνέπεια ο χρόνος απόκρισης του αλγόριθμου ανά χρονόσημο να αποτελεί μια γραμμική συνάρτηση του πλήθους των αντικειμένων N. Μια πιθανή κλιμάκωση του N σε μεγαλύτερης τάξεως τιμές διαφαίνεται ότι θα οδηγήσει τον αλγόριθμο σε αποκρίσεις μη αποδεκτές για εφαρμογές πραγματικού χρόνου.

Μια πιθανή βελτίωση στο πρόβλημα αυτό είναι να εφαρμοσθεί η τεχνική filter and refinement με την μορφή ενός ευρετηρίου όπως η διαμέριση σε πλέγμα (grid partitioning) στις σημειακές θέσεις του ρεύματος. Σε ένα προπαρασκευαστικό στάδιο για ένα δεδομένο χρονόσημο, οι σημειακές θέσεις δεικτοδοτούνται στο πλέγμα. Μετά τον καθορισμό του σημείου ερωτήσεως Q, ο αλγόριθμος επεξεργάζεται διαδοχικά τα σημεία που περιέχονται στην γειτονιά του Q μέχρι να βρει τους k εγγύτερους γείτονες και το σημείο ελέγχου για κάθε κυκλικό τομέα. Έτσι αποφεύγεται στην πλειοψηφία των περιπτώσεων η εξαντλητική αναζήτηση στο σύνολο των δεδομένων, οδηγώντας σε μικρότερους χρόνους απόκρισης.

Πέρα από την εξασφάλιση της αποδοτικής επεξεργασίας μεγάλων συνόλων κινούμενων αντικειμένων, αυτή η λύση προσφέρει την δυνατότητα για ταυτόχρονη επεξεργασία πολλών σημείων ερωτήσεως. Η μορφή του αλγόριθμου που περιγράφηκε αποκλείει αυτό το ενδεχόμενο, αφού οι κατατμήσεις των διάφορων σημείων ερωτήσεως είναι ασύμβατες μεταξύ τους. Η παράλληλη εκτέλεση νοείται μόνο για πολλαπλές εκτελέσεις του αλγόριθμου είτε σε διαφορετικούς υπολογιστές είτε σε διαφορετικά νήματα. Η χρήση όμως του πλέγματος στο προπαρασκευαστικό στάδιο, μπορεί να δώσει ένα κοινό σύστημα αναφοράς για όλα τα κινούμενα αντικείμενα και άρα καθιστά δυνατή την παράλληλη εκτέλεση πολλαπλών ερωτημάτων. Για κάθε σημείο ερωτήσεως Q_i επεξεργάζονται οι σημειακές θέσεις ξεκινώντας από μια κοντινή γειτονιά του τμήματος του πλέγματος στο οποίο αυτό αντιστοιχεί, μέχρις ότου καλυφθούν τα στοιχεία που είναι απαραίτητα σε κάθε τομέα.

Παράρτημα Α΄

Περιγραφή υλοποιήσεων

Α΄.1 Ο αλγόριθμος του Fortune

Α΄.1.1 Περιγραφή υλοποιημένων δομών

Για την κατασκευή των διαγραμμάτων *Voronoi* προτιμήθηκε ο αλγόριθμος του Fortune που αν και εισάγει πολυπλοκότητα στην υλοποίηση εξασφαλίζει μειωμένους χρόνους κατασκευής. Έτσι σύμφωνα και με τις δομές που περιγράφηκαν προηγουμένως στην θεωρητική παρουσίαση του αλγόριθμου δημιουργήθηκαν οι εξής κλάσεις

Α΄.1.2 Κύριες δομές

Ακολουθούν οι δομές που υλοποιήθηκαν. Μια πιο αναλυτική περιγραφή και των μεθόδων χειρισμού της καθεμιάς θα γίνει σε επόμενη ενότητα.

- Μια διπλοσυνδεδεμένη λίστα ακμών (*halfEdgeList*), όπου αποθηκεύονται οι ακμές του διαγράμματος. Σε αυτή τη δομή αποθηκεύονται επίσης οι κορυφές του διαγράμματος αλλά και τα κελιά που ορίζουν οι ακμές. Να σημειωθεί ότι όλες οι half edges του προγράμματος παρακολουθούνται από την αρχή προς το τέλος τους. Για την ακρίβεια όταν δημιουργούνται τίθεται η αρχή τους, είτε σε κορυφή (αν είναι μετά από γεγονός κύκλου), είτε σε ένα προσωρινό σημείο (αν είναι μετά από γεγονός εστίας).

- Μια ουρά γεγονότων (*eventQueue*) υλοποιημένη ως ουρά προτεραιότητας. Σε αυτή αποθηκεύονται όλα τα επικείμενα γεγονότα εστίας και κύκλου, ταξινομημένα ως προς την y-συντεταγμένη τους. Η δομή αυτή για ευκολία χρησιμοποιήθηκε επίσης για να κατασκευαστεί μια λίστα με τις αρχικές εστίες και τις κορυφές του διαγράμματος. Η βοηθητική αυτή λίστα αξιοποιείται στην κατασκευή του περιβάλλοντος ορθογωνίου του διαγράμματος.

- Η κεντρική δομή είναι το ισορροπημένο δένδρο όπου φυλάσσεται η κατάσταση της παραλιακής γραμμής. Η κλάση *beachLineTree* υλοποιεί αυτό το δένδρο όπου τα φύλλα αντιστοιχούν στα τόξα που συγκροτούν την παραλιακή γραμμή από αριστερά προς τα δεξιά, αποθηκεύοντας για κάθε τόξο το εστιακό σημείο p_i που το ορίζει, ενώ κάθε εσωτερικός κόμβος αντιπροσωπεύει το σημείο καμπής των τόξων που αντιστοιχούν στα δύο παιδιά του. Κάθε φύλλο του δένδρου έχει έναν δείκτη (pointer) προς το αντίστοιχο γεγονός κύκλου της ουράς eventQueue, ενώ κάθε εσωτερικός κόμβος δείχνει σε μια ακμή της λίστας halfEdgeList. Το δένδρο θεωρείται εκτός ισορροπίας αν δύο διαδοχικά φύλλα έχουν διαφορά βάθους μεγαλύτερη από ένα.

- Για την διευκόλυνση της υλοποίησης στις θεωρητικά περιγραφόμενες δομές, προστέθηκε μια δομή λίστας (*leafList*), που περιέχει κατ' αύξουσα y-συντεταγμένη όλα τα φύλλα του δένδρου. Η δομή αυτή ανανεώνεται κάθε φορά που αλλάζει το δένδρο που περιγράφει η beachLineTree. Η λίστα αυτή έχει εισαχθεί στην κλάση beachLineTree.

Σε όλες τις δομές που περιγράφηκαν παραπάνω αντιστοιχούν δύο κλάσεις. Μία που περιγράφει τον τυχαίο κόμβο της δομής και την σύνδεσή του με τους υπόλοιπους, και μία άλλη που έχει έναν δείκτη στον πρώτο κόμβο της δομής. Στην δεύτερη αυτή κλάση, η οποία σε κάθε περίπτωση έχει το όνομα της δομής όπως περιγράφηκε παραπάνω, υλοποιούνται και οι μέθοδοι οι οποίες διαχειρίζονται τα μηνύματα που ανταλλάσσουν οι δομές.

Α΄.1.3 Μέθοδοι διαχείρισης

Για κάθε κλάση θα δώσουμε μια μικρή περιγραφή των κύριων μεθόδων που διαθέτει. Δεν θα αναφερθούν μέθοδοι που προσθέτουν ή αφαιρούν κόμβους στις λίστες αφού η λειτουργία τους είναι τετριμμένη. Η περιγραφή αυτή θα διευκολύνει την επεξήγηση της λειτουργίας του προγράμματος αργότερα.

- **HalfEdgeList**
  - **isAVertex :** Ελέγχει αν ένα σημείο ανήκει στις κορυφές του διαγράμματος που έχουν βρεθεί.
  - **setCell :** Δέχεται μια εστία και μια ακμή και ενημερώνει τον δείκτη του half edge στο σωστό κελί.
  - **addNewVertex :** Προσθέτει μια νέα κορυφή.
  - **createBoundingBox :** Δημιουργεί το περιβάλλον ορθογώνιο γύρω από το διάγραμμα.
  - **pointInCell :** ελέγχει αν ένα σημείο είναι μέσα σε ένα κελί.

- **beachLineTree**
  - **findLeaf :** Βρίσκει το φύλλο κάτω από το οποίο βρίσκεται το καινούργιο γεγονός.

- **handleSiteEvent :** Χειρίζεται τα γεγονότα εστίας.
- **checkForCircleEvents :** Ελέγχει τα τόξα για εμφάνιση πιθανού γεγονότος κύκλου.
- **handleCircleEvent :** Χειρίζεται τα γεγονότα κύκλου.
- **reHashLeafList :** Αναδημιουργεί την λίστα με τους δείκτες στα φύλλα του δέντρου. (leafList)
- **calculateDepths :** Ενημερώνει σε κάθε κόμβο την τιμή του βάθους στο οποίο βρίσκεται (βάθος ρίζας = 0).
- **checkIfUnbalanced :** ελέγχει αν το δένδρο είναι εκτός ισορροπίας συγκρίνοντας τα βάθη διαδοχικών φύλλων.
- **balanceBeachLineTree :** Αναδιατάσσει κατάλληλα τους κόμβους του δένδρου έτσι ώστε να είναι ισορροπημένο.

Εκτός από τις κλάσεις, έχουν υλοποιηθεί και μερικές βοηθητικές συναρτήσεις οι κυριότερες από τις οποίες είναι:

- **findIntersect :** Βρίσκει το κέντρο του κύκλου που ορίζουν τρεις εστίες, υπολογίζοντας το σημείο τομής των μεσοκαθέτων των ευθύγραμμων τμημάτωνπου ενώνουν τις εστίες ανά δύο.
- **findBreakPoint :** Βρίσκει το σημείο τομής των παραβολών που ορίζουν δύο εστίες και η ευθεία σάρωσης.
- **calculateBoundingBoxIntersectPoint :** Βρίσκει για μια μη πεπερασμένη ακμή το σημείο όπου τέμνεται με το περιβάλλον ορθογώνιο.
- **segmentIntersect :** Χρησιμοποιείται για να επιστρέψει την τομή ή όχι μιας ημιευθείας με ένα ευθύγραμμο τμήμα. Την χρησιμοποιεί ο αλγόριθμος ποιντ ιν πολψγον.

Α΄.1.4 Περιγραφή λειτουργίας

Η πορεία εκτέλεσης του προγράμματος έχει ως εξής. Τα αρχικά σημεία (εστίες) για τα οποία ζητείται το διάγραμμα *Voronoi* διαβάζονται από ένα αρχείο ASCII με κατάλληλο format. Αποθηκεύονται στην eventQueue ως γεγονότα εστίας, παίρνοντας αυτόματα θέση μέσα στη λίστα ανάλογα με την y-συντεταγμένη τους. Τα γεγονότα διαβάζονται διαδοχικά ξεκινώντας με το πρώτο στη λίστα, δηλαδή αυτό με την μεγαλύτερη y-συντεταγμένη. Αν το γεγονός είναι γεγονός εστίας, καλείται η μέθοδος handleSiteEvent και γίνεται χειρισμός του γεγονότος. Αν είναι γεγονός κύκλου το χειρίζεται η handleCircleEvent. Το πρόγραμμα σταματά την επεξεργασία όταν εξαντληθούν τα περιεχόμενα της λίστας.

Όταν κληθεί η handleSiteEvent ο νέος κόμβος προστίθεται στο δένδρο και δημιουργούνται οι νέες half edges. Σε αυτές προσδιορίζονται τα κελιά όπου ανήκουν αλλά και τα σημεία αρχής τους. Εφόσον σημειώθηκε αλλαγή στο δένδρο

ανανεώνεται η leafList και γίνεται έλεγχος για γεγονότα κύκλου καλώντας την checkForCircleEvents. Η checkForCircleEvents χρησιμοποιεί την βοηθητική λίστα leafList για να εξετάσει όλες τις διαδοχικές τριάδες φύλλων (τόξων) για γεγονότα κύκλου. Αυτό το κάνει υπολογίζοντας ανά δύο τα σημεία καμπής με κλήση της findBreakPoint και βρίσκοντας την ευκλείδεια απόσταση ανάμεσα στην τωρινή τους θέση και σε μια επόμενη που απέχει λίγο. Αν υπάρχει σύγκλιση των σημείων καμπής ορίζεται ένα γεγονός κύκλου, το κέντρο του οποίου υπολογίζεται με την findIntersect. Δημιουργείται ένα καινούργιο γεγονός κύκλου και εισάγεται στην λίστα γεγονότων eventQueue. Στο τέλος κάθε γεγονότος κύκλου ή εστίας καλούνται οι calculateDepths και checkIfUnbalanced για να διαπιστωθεί αν το δένδρο με την προσθαφαίρεση κόμβων έχει βρεθεί εκτός ισορροπίας. Στην περίπτωση που είναι εκτός ισορροπίας καλείται η balanceBeachLineTree για να το επαναφέρει.

Η handleCircleEvent βρίσκει τους κόμβους του δένδρου που πρέπει να αφαιρεθούν, περατώνει τις half edges που παρακολουθούσαν οι κόμβοι αυτοί, αποθηκεύει την κορυφή του διαγράμματος που βρέθηκε και δημιουργεί ένα νέο ζεύγος half edges που ξεκινούν από την κορυφή αυτή. Αφού ενημερώσει την leafList, κάνει νέο έλεγχο για γεγονότα κύκλου καλώντας την checkForCircleEvents. Μόλις επεξεργασθεί ολόκληρη η eventQueue, καλείται η createBoundingBox η οποία και κλείνει το διάγραμμα *Voronoi* μέσα σε ένα ορθογώνιο, οι διαστάσεις του οποίου μπορούν να μεταβληθούν μέσω μιας σταθεράς. Αυτό επιφέρει και την προσθήκη νέων κορυφών και ακμών στο διάγραμμα.

Μετά την ολοκλήρωση της κατασκευής του στατικού διαγράμματος *Voronoi* για τις εστίες που έχουν οριστεί, ακολουθεί η αξιοποίησή του για την ανεύρεση της εγγύτερης εστίας για ένα δυναμικά μεταβαλλόμενο σύνολο σημείων. Το πρόγραμμα διαβάζει τα κινούμενα σημεία από ένα αρχείο ASCII παρόμοιας μορφής με το αρχείο των εστιών. Για κάθε σημείο που δέχεται ελέγχει μέσα σε ποιο κελί του διαγράμματος *Voronoi* βρίσκεται (point in polygon) καλώντας την μέθοδο pointInCell διαδοχικά για όλα τα κελιά. Έτσι προσδιορίζεται για κάθε κινούμενο σημείο ο εγγύτερος του γείτονας.

Α΄.2 Αλγόριθμος υπολογισμού προσεγγιστικού κελιού Voronoi

Α΄.2.1 Κύριες δομές

Στην συνέχεια παρατίθενται οι δομές που υλοποιήθηκαν. Μια πιο αναλυτική περιγραφή και των μεθόδων χειρισμού της καθεμιάς θα γίνει σε επόμενη ενότητα.

- Μια διπλοσυνδεδεμένη λίστα ακμών (*halfEdgeList*), όπου αποθηκεύονται οι ακμές του κελιού.
- Μια λίστα σημείων για κάθε τομέα (*pointList*), όπου αποθηκεύονται τα σημεία που αντιστοιχούν στον τομέα αυτό, αλλά κανένας τους δεν ανήκει στους k εγγύτερους γείτονες.

- Μία λίστα σημείων (*neighborsList*), όπου αποθηκεύονται τα σημεία που αποτελούν τους k τρέχοντες εγγύτερους γείτονες του Q.
- Μία κυκλική λίστα τομέων (*domainList*) που αποτελεί την κεντρική δομή του προγράμματος. Έχει αριθμό κόμβων ίσο με αυτό των τομέων που ο χρήστης επιλέγει να κατατμήσει τον χώρο. Από την δομή αυτή παρέχεται πρόσβαση με δείκτες στην *halfEdgeList* που περιγράφει το κελί, αλλά και στην *neighborsList* που περιέχει τους k εγγύτερους γείτονες. Επίσης εδώ αποθηκεύονται οι συντεταγμένες του σημείου ερωτήσεως και στην παραλλαγή του χρονικού παραθύρου, τα δύο ευρετήρια παρακολούθησης, αυτό των αντικειμένων και αυτό του παραθύρου. Κάθε κόμβος της διαθέτει από μια pointList από ένα ευρετήριο σημείων ανά χρονόσημο.

Α΄.3 Αλγόριθμος ταυτόχρονης ανανέωσης

Α΄.3.1 Μέθοδοι διαχείρισης

Ακολουθεί μια μικρή περιγραφή των κύριων μεθόδων. Δεν θα αναφερθούν μέθοδοι που προσθέτουν ή αφαιρούν κόμβους στις λίστες αφού η λειτουργία τους είναι τετριμμένη.

- **domainList**
  - **createPartitions :** χωρίζει το επίπεδο σε λ τομείς με $2\pi/\lambda$ γωνία ο καθένας.
  - **findDomainOfPoint :** βρίσκει σε ποιόν από τους τομείς του επιπέδου αντιστοιχεί το σημείο p και το εισάγει στην αντίστοιχη λίστα pointList.
  - **createInitialBoundingBox :** κατασκευάζει ένα βοηθητικό αρχικό κελί γύρω από τα σημεία στα οποία εκτελείται η φάση αρχικοποίησης. Το κελί αυτό είναι απλά ένα ορθογώνιο που περιέχει όλα τα σημεία και το εύρος του εξαρτάται από την καθολική μεταβλητή FLOAT.
  - **constructLines :** για κάθε τομέα καλεί την **constructLineForASector**, για να υπολογίσει το νέο κελί.
  - *constructLineForASector :* βρίσκει την πιο περιοριστική μεσοκάθετο που αντιστοιχεί στο σημείο ελέγχου του τομέα και καλεί την μέθοδο clipCell.
  - **clipCell :** ανανεώνει το κελί *Voronoi* περικόπτοντας το παλιό χρησιμοποιώντας την μεσοκάθετο που ορίζει το σημείο ελέγχου ενός τομέα.
  - **findInitialNeighbors :** για τα σημεία που έχουν εισαχθεί στον χώρο βρίσκει τους k εγγύτερους γείτονες και τους βάζει στην λίστα nearestNeighbors, σημαδεύοντάς τα ταυτόχρονα στον τομέα.

Σημαδεύει επίσης τα σημεία ελέγχου και τα υπόλοιπα σημεία σε κάθε τομέα.

* **addPointToList :** για δεδομένο τομέα και δεδομένο σημείο, εξετάζει αν και πως θα εισαχθεί στην λίστα σημείων του.

Α΄.3.2 Περιγραφή λειτουργίας

Η πορεία εκτέλεσης του αλγορίθμου έχει ως εξής. Η σημειακή θέση του Q διαβάζεται από ένα αρχείο ASCII με κατάλληλο format και ένα αντικείμενο της κλάσης *domainList* δημιουργείται με βάση αυτό. Οι υπόλοιπες παράμετροι που χρειάζονται για την δημιουργία του, δηλώνονται στον κώδικα ως σταθερές και είναι ο αριθμός των κινούμενων αντικειμένων, ο αριθμός των τομέων και ο αριθμός των εγγύτερων γειτόνων που αναζητούνται. Μετά την δημιουργία της κατάτμησης, με την κλήση της *createPartitions*, το πρόγραμμα διαβάζει τις σημειακές θέσεις των κινούμενων αντικείμενων από αρχείο ASCII, ίδιου format με αυτό του Q. Κατά την διάρκεια του διαβάσματος των σημειακών θέσεων, ελέγχεται το χρονόσημο (timestamp) που φέρουν και συγκρίνεται με το τρέχον του συστήματος. Εφόσον αυτά βρίσκονται ίσα, γίνεται εισαγωγή της σημειακής θέσης με την κλήση της *findDomainOfPoint*. Η μέθοδος αυτή εξασφαλίζει ότι δεν θα εισαχθεί μεγαλύτερος από τον επιτρεπτό αριθμό σημειακών θέσεων σε έναν τομέα, απορρίπτοντας σημεία σύμφωνα με τα κριτήρια που περιγράφτηκαν πριν. Ο έλεγχος αυτός υλοποιείται στην *addPointToList*.

Μόλις βρεθεί η πρώτη σημειακή θέση που φέρει χρονόσημο μεγαλύτερο από αυτό του συστήματος, αυτό δεν εισάγεται και το σύστημα καλεί τις *findInitialNeighbors* και *createInitialBoundingBox*. Έτσι δημιουργείται η λίστα των k εγγύτερων γειτόνων και ένα ορθογώνιο κελί γύρω από τις σημειακές θέσεις. Έχοντας το περιβάλλον ορθογώνιο, καλείται η *constructLines* και δημιουργείται το προσεγγιστικό κελί για το τρέχον χρονόσημο.

Πριν την έναρξη του νέου χρονοσήμου το αντικείμενο της κλάσης domainList που περιγράφει την κατάτμηση του χώρου διαγράφεται και ένα νέο δημιουργείται. Από το αρχείο εισόδου διαβάζεται το νέο σημείο ερωτήσεως (αν υπάρχει) και η διαδικασία επαναλαμβάνεται από την αρχή διαβάζοντας νέα στοιχεία από το ρεύμα εισόδου μέχρι την εξάντλησή του.

Α΄.4 Αλγόριθμος χρονικού παραθύρου

Α΄.4.1 Κύριες δομές

Εκτός από τις δομές του αλγόριθμου ταυτόχρονης ανανέωσης, η παραλλαγή του χρονικού παραθύρου χρησιμοποιεί τρία ευρετήρια. Τα ευρετήρια σχεδιάστηκαν με την μορφή πίνακα και είναι τα εξής:

- **Ευρετήριο σημείων ανά χρονόσημο :** Βρίσκεται σε κάθε κόμβο της *domainList* και δείχνει πόσες σημειακές θέσεις έχει αποθηκευμένες ανά

χρονόσημο ο τομέας αυτός. Είναι ένας μονοδιάστατος πίνακας ακεραίων, με διάσταση $\Delta\tau$.

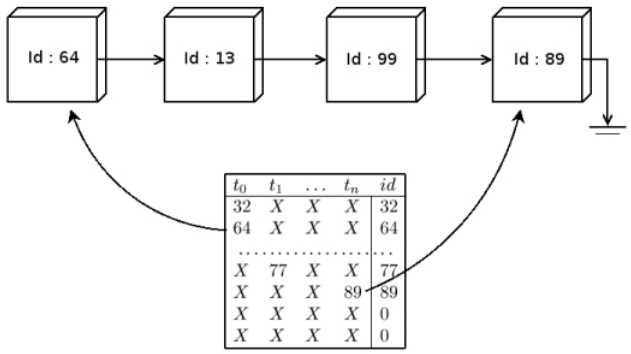
- **Ευρετήριο παρακολούθησης αντικειμένων :** Είναι μέλος της *domainList* και παρακολουθεί ποια αντικείμενα είναι ενεργά στο σύστημα. Υλοποιήθηκε ως μονοδιάστατος πίνακας δεικτών σε κόμβους της λίστας *pointList* και έχει διάσταση ίση με των αριθμό των συνολικών κινούμενων αντικειμένων που καλείται να εξυπηρετήσει το σύστημα. Η αντιστοίχηση των id tags των αντικειμένων από το αρχείο εισόδου με την θέση τους στον πίνακα είναι ένα προς ένα. Με αυτό τον τρόπο ελέγχοντας απλά την θέση του πίνακα που αντιστοιχεί στο id του αντικειμένου μπορούμε να διαπιστώσουμε αν είναι ενεργό ή όχι.

- **Ευρετήριο παρακολούθησης παραθύρου :** Πρόκειται για την δομή που ουσιαστικά εφαρμόζει το μοντέλο παραθύρου στον αλγόριθμο. Είναι ένας δισδιάστατος πίνακας δεικτών σε κόμβους της λίστας *pointList*, διαστάσεων $[\Delta\tau+1]\times[(k+1)*\lambda]$. Η λειτουργία του έγκειται στην παρακολούθηση του τρέχοντος χρονοσήμου, συν τα επόμενα $\Delta\tau-1$. Χωρίζεται σε δύο κομμάτια, τις πρώτες $\Delta\tau$ στήλες και την τελευταία στήλη. Στο πρώτο μέρος, κάθε στήλη αντιπροσωπεύει ένα χρονόσημο. Στην στήλη που αντιστοιχεί στο χρονόσημο που η σημειακή θέση λήγει, υπάρχει ένας δείκτης στον αντίστοιχο κόμβο της *pointList* του τομέα. Οι υπόλοιπες στήλες έχουν τιμή 0. Το δεύτερο μέρος αποτελεί μια στήλη όπου δείχνει αν η αντίστοιχη γραμμή του ευρετηρίου είναι κατειλημμένη, και αν ναι από ποιο σημείο. Αυτό μαρκάρεται πάλι ή με δείκτη στον κόμβο της *pointList* ή με 0. Κατά την ολίσθηση του ευρετηρίου προς τα αριστερά, αν στην πρώτη στήλη του πρώτου μέρους υπάρχει δείκτης προς σημείο, τότε η ισχύς της σημειακής του θέσης έχει παρέλθει και αφαιρείται από το σύστημα. Όταν ένα αντικείμενο παίρνει μια σημειακή θέση, δείκτες προς τον αντίστοιχο κόμβο της *pointList* μπαίνουν στην τελευταία στήλη του πρώτου μέρους και στην μοναδική του δεύτερου.

Α΄.4.2 Μέθοδοι διαχείρισης

Ακολουθεί μια μικρή περιγραφή των κύριων μεθόδων που ορίζονται επιπρόσθετα σε αυτές του αλγόριθμου ταυτόχρονης ανανέωσης.

- **domainList**
  - **updateCellAfterControlPointRemoval :** όταν αλλάζει ένα σημείο ελέγχου πρέπει να αφαιρεθεί η halfedge του κελιού που κατασκευάστηκε από την μεσοκάθετο που ορίζει το σημείο. Υπάρχουν δύο περιπτώσεις. Η επόμενη και η προηγούμενη halfedge του κελιού να συγκλίνουν και άρα απλά πρέπει να ενωθούν. Στην περίπτωση που δεν συγκλίνουν, κλείνουμε το κελί με ένα *περιβάλλον ορθογώνιο* (bounding box), τα όρια του οποίου είναι αρκετά μεγάλα.



Σχήμα Α΄.1: Ευρετήριο παρακολούθησης παραθύρου

- **shiftTime :** καλείται πριν αλλάξει το χρονόσημο. Αφαιρεί τα σημεία που βρίσκονται στην αριστερή στήλη του ευρετηρίου χρόνου timeStamps από τις λίστες σημείων, ολισθαίνοντας το ευρετήριο μια θέση προς τα αριστερά. Για κάθε τομέα ανανεώνει τα σημεία ανά timeStamp και υπολογίζει τον συνολικό τρέχοντα αριθμό σημείων του
- **updateCellAfterPointRemoval :** όταν αφαιρείται από έναν τομέα το σημείο ελέγχου του πρέπει να ακυρωθεί και η μεσοκάθετος που συνεισέφερε στο κελί. Έτσι η μέθοδος αυτή, βρίσκει την μεσοκάθετο που αντιστοιχεί στο σημείο ελέγχου, την αφαιρεί και κλείνει το κελί στα όρια του περιβάλλοντος κουτιού (bounding box) που περικλείει τον χώρο κίνησης των αντικειμένων.
- **updateCell :** Η μέθοδος αναλαμβάνει τον έλεγχο και την υλοποίηση των τυχόν διορθώσεων που χρειάζεται το κελί στο τέλος της φάσης ανανέωσης. Πρώτα ελέγχει αν έχει αλλάξει κάποιος από τους εγγύτερους γείτονες. Αν έχει αλλάξει, τότε επανεκκινεί τον αλγόριθμο από την φάση αρχικοποίησης, καλώντας τις *createInitialBoundingBox* και *constuctLines*. Ο επόμενος έλεγχος είναι αν έχει αλλάξει κάποιο σημείο ελέγχου. Στην περίπτωση που αυτό ισχύει καλείται η *updateCellAfterPointRemoval* και μετά η *constructLineForASector* για να ανανεωθεί η συνεισφορά του τομέα στο κελί σύμφωνα με το νέο σημείο ελέγχου.

Α΄.4.3 Περιγραφή λειτουργίας

Η πορεία εκτέλεσης για την περίπτωση ακίνητου σημείου ερωτήσεως Q έχει ως εξής. Η σημειακή θέση του Q διαβάζεται από ένα αρχείο ASCII με κατάλληλο format και ένα αντικείμενο της κλάσης *domainList* δημιουργείται με βάση

αυτό. Οι υπόλοιπες παράμετροι που χρειάζονται για την δημιουργία του, δηλώνονται στον κώδικα ως σταθερές και είναι ο αριθμός των κινούμενων αντικειμένων, ο αριθμός των τομέων, ο αριθμός των εγγύτερων γειτόνων και το εύρος του παραθύρου. Μετά την δημιουργία της κατάτμησης, με την κλήση της *createPartitions*, το πρόγραμμα διαβάζει τις σημειακές θέσεις των κινούμενων αντικείμενων από αρχείο ASCII, ίδιου format με αυτό του Q. Κατά την διάρκεια του διαβάσματος των σημειακών θέσεων, ελέγχεται το χρονόσημο (timestamp) που φέρουν και συγκρίνεται με το τρέχον του συστήματος. Εφόσον αυτά βρίσκονται ίδια, γίνεται εισαγωγή της σημειακής θέσης με την κλήση της *findDomainOfPoint*. Η μέθοδος αυτή εξασφαλίζει ότι δεν θα εισαχθεί μεγαλύτερος από τον επιτρεπτό αριθμό σημειακών θέσεων σε έναν τομέα, απορρίπτοντας σημεία σύμφωνα με τα κριτήρια που περιγράφτηκαν πριν. Ο έλεγχος αυτός υλοποιείται στην *addPointToList*.

Μόλις βρεθεί η πρώτη σημειακή θέση που φέρει χρονόσημο μεγαλύτερο από αυτό του συστήματος, αυτό δεν εισάγεται και το σύστημα καλεί τις *findInitialNeighbors* και *createInitialBoundingBox*. Έτσι δημιουργείται μια πρώτη λίστα k εγγύτερων γειτόνων και ένα ορθογώνιο κελί γύρω από τις σημειακές θέσεις. Να σημειωθεί ότι η διαδικασία αυτή μπορεί να γίνει και πιο νωρίς, θεωρητικά μόλις εισαχθούν $k+1$ σημειακές θέσεις στο σύστημα. Για λόγους απλότητας, στην υλοποίηση επιλέχτηκε να γίνει με το τέλος του πρώτου χρονοσήμου Έχοντας το περιβάλλον ορθογώνιο, καλείται η *constructLines* και δημιουργείται το προσεγγιστικό κελί, σηματοδοτώντας το τέλος της *φάσης αρχικοποίησης*.

Η διαδικασία που ακολουθείται για τα επόμενα χρονόσημα είναι επαναλαμβανόμενη και αποτελεί την *φάση ανανέωσης*. Στην αρχή κάθε χρονοσήμου γίνεται ολίσθηση του χρόνου και των ευρετηρίων με την κλήση της *shiftTime*. Αφού αφαιρεθούν οι σημειακές θέσεις που έχει λήξει η ισχύς τους, συνεχίζεται η ανάγνωση των νέων σημειακών θέσεων από το αρχείο ASCII, μέχρι να συναντηθεί εκ νέου διαφορετικό χρονόσημο. Τότε καλείται η *updateCell* και το κελί διαμορφώνεται έτσι ώστε να ανταποκρίνεται στο τωρινό χρονόσημο. Από αυτό το σημείο αρχίζει η επανάληψη της φάσης ανανέωσης που περιγράφτηκε, μέχρι να εξαντληθούν οι σημειακές θέσεις από το αρχείο.

Στην περίπτωση που έχουμε κινούμενο σημείο ερωτήσεως στο τέλος κάθε χρονοσήμου ελέγχεται αν αλλάζει την σημειακή του θέση. Όταν εντοπιστεί αλλαγή, αποθηκεύονται οι ενεργές σημειακές θέσεις καλώντας την *backupPoints*, ενώ οι καταστρέφονται οι λίστες που περιγράφουν το κελί και τους γείτονες. Η κατάτμηση ενημερώνεται στο νέο σημείο ερωτήσεως και καλείται η *redistributePoints* για να αντιστοιχησθούν οι σημειακές θέσεις σε αυτήν. Στην συνέχεια υπολογίζονται οι εγγύτεροι γείτονες και το κελί με την διαδικασία που ακολουθείται στην φάση αρχικοποίησης. Τέλος γίνεται ολίσθηση του χρόνου καλώντας την *shiftTime* και διαβάζεται εκ νέου το ρεύμα εισόδου.

Βιβλιογραφία

[ABB+02] A. Arasu, B. Babcock, S.Babu, J. McAlister, and J. Widom. Characterizing memory requirements for queries over continuous data streams. In *Proceedings of the* 21^{st} *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*,pp. 221-232, Madison, Winsconsin, May 2002.

[Aur86] Franz Aurenhammer. The one-dimensional weighted Voronoi diagram. *Information Processing Letters*, 22(3):119-123, 3 March 1986.

[Aur91] Franz Aurenhammer. Voronoi diagrams-A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345-405, September 1991.

[BBD+02] Babcock, Babu, Datar, Motwani, and Widom. Models and issues in data stream systems. In *PODS: 21th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2002.

[Chi00] Charles Chien. Distributed services for microsensor networks (sensorware). In *Rockwell Science Center Mani Srivastavas UCLA*, April 2000.

[dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, January 2000.

[GG98] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170-231, 1998.

[For87] Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153-174, 1987.

[GG98] Gaede and Gunther. Multidimensional access methods. *CSURV: Computing Surveys*, 30, 1998.

[GKMS01] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin J.Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the 27th International Conference on Very Large Data Bases(VLDB '01)*, pages 79-88, Orlando, September 2001. Morgan Kaufmann.

[GO03] Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *SIGMOD Record*, 32(2):5-14, 2003.

[Gut94] Ralf Hartmut Guting. Special issue on spatial database systems: An introduction to spatial database systems. *Very Large Data Bases Journal: Very Large Data Bases*, 3(4):357-399, October 1994. Electronic edition.

[Har95] William W. Hargrove. Dynamics segmentation and thiessen polygons: A solution to the river mile problem. In *Fifteenth Annual ESRI User Conference Proceedings*, Palm Springs, California, 1995.

[KS04] Mohammad R. Kolahdouzan and Cyrus Shahabi. Continuous K-nearest neighbor queries in spatial network databases. In Jorg Sander and Mario A. Nascimento, editors, STDBM, pages 33-40, 2004.

[Kyp06] Εμμανουήλ Κυπραίος. *Επεξεργασία ερωτημάτων διαρκείας σε ρεύματα κινούμενων αντικειμένων*. Αθήνα 2006.

[LS05] Paul Labute and Martin Santavy. Locating binding sites in protein structures. *Chemical Computing Group Inc.*, 2005.

[MDA02] Vincent Masselus, Philip Dutre, and Frederik Anrys. The free form light stage. In Simon Gibson and Paul Debevec, editors, *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 247-256, Aire-la-Ville, Switzerland, June 26-28 2002. Eurographics Association.

[OBS94] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. Nearest neighbourhood operations with generalized voronoi diagrams: a review. In *International Journal of Geographical Information Systems*, volume 8, pages 43-71. 1994.

[Pat03] Κωστας Πατρούμπας. *Συστήματα ρευμάτων δεδομένων για κινούμενα αντικείμενα*. ΑΘήνα 2003.

[RRH99] V. Raman, B. Raman, and J, Hellerstein. Online Dynamic Reordering for Interactive Data Processing. In *Proceedings of the 1999 International Conference On Very Large Data Bases*, Edinburgh, Scotland, September 1999.

[RSV02] Philippe Rigaux, Michel Scholl, and AgnAs Voisard. Spatial Databases with application to GIS. Morgan Kaufmann, San Francisco, CA, 2002.

[SCZ05] Michael Stonebraker, Ugur Cetintemel, and Stanley B. Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Record*, 34(4):42-47, 2005.

[SH75] Shamos and Hoey. Closest-point problems. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1975.

[SOB92] K. Sugihara, A. Okabe, and B. Boots. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 1992.

[SS06] Mehdi Sharifzadeh and Cyrus Shahabi. Utilizing voronoi cells of location data streams for accurate computation of aggregate functions in sensor networks. *GeoInformatica*, 10(1):9-36, 2006.

[TCZ+03] Nesime Tatbul, Ugur Cetintemel, Stanley B. Zdonik, Mitch Cherniack, and Michael Stonebraker. Load shedding in a data stream manager. In *Very Large Data Bases*, pages 309-320, 2003.

[The03] Yannis Theodoridis. Ten benchmark database queries for location-based services. *Comput. J*, 46(6):713-725, 2003.

[The99] Γιάννης Θεοδωρίδης. Βάσεις Χωρικών δεδομένων. *Προχωρημένα θέματα βάσεων δεδομένων. Συμπληρωματικές σημειώσεις*: 167-193. Εκδόσεις ΕΜΠ, Αθήνα 1999.

[TMS003] P. Tucker, D. Maier, T. Sheard. Applying Punctuation Schemes to Queries Over Continuous Data Streams. *Bulletin of the IEEE on Data Engineering*, 26(1):33-40, March 2003

[TMSF02] P. Tucker, D. Maier, T. Sheard, and L. Fegaras. Enhancing Relational Operators for Querying over Punctuated Data Streams. In *Proceedings of the* 28^{th} *International Conference On Very Large Data Bases*, Hong Kong, China, 2002.

[WSS06] Yuan Wei, Sang Hyuk Son, and John A. Stankovic. RTSTREAM: Real-time query processing for data streams. In *IEEE International Symposium on Object-oriented Real-time distributed Computing*, pages 141-150. IEEE Computer Society, 2006.

[YMV05] A. Papadopoulos Y. Manolopoulos and M. Vassilakopoulos, editors. Spatial Databases: Technologies, Techniques and Trend. *IDEA Group Publishing, Information Science Publishing and IRM Press*, 2005.

[ZLL04] Baihua Zheng, Wang-Chien Lee, and Dik Lun Lee. On semantic caching and query scheduling for mobile nearest-neighbor search. *Wireless Networks*, 10(6):653-664, 2004.

[ZZP+03] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D.L. Lee. Location-based Spatial Queries. In *Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data*,pp. 443-454, San Diego, California, June 2003.

Γλωσσάρι

αποβολή φόρτου	load shedding
γραμμή σάρωσης	sweep line
ελάχιστο περιβάλλον παραλληλόγραμμο	minimum bounding box
ερώτημα διαρκείας	continuous query
ερώτημα εγγύτερου γείτονα	nearest-neighbor query
ερώτημα θέσης	location-based query
ερώτημα περιοχής	range query
ερώτημα στιγμιοτύπου	one-time, snapshot query
ερώτημα τροχιάς	trajectory-based query
κινούμενο αντικείμενο	moving object
κλιμάκωση	scalability
κυματίδιο	wavelet
κυρτό περίβλημα	convex hull
παράθυρο κυλιόμενο	sliding window
παραλιακή γραμμή	beachline
περίληψη	summary
ρεύμα δεδομένων	data stream
ρεύμα τροχιάς	trajectory stream
σκίτσο	sketch
συνάθροιση	aggregation
σύνδεση	join
σύνοψη	synopsis
σχέση	relation
τελεστής	operator
υπηρεσίες εντοπισμού	location-based services
χρονόσημο	timestamp
χωρικές μέθοδοι προσπέλασης	spatial access methods
χωρική σύνδεση	spatial join

Επεξεργασία χωρικών ρευμάτων δεδομένων με διαγράμματα Voronoi

Μηνόγιαννης Θεοφάνης
el01248@mail.ntua.gr

Διπλωματική εργασία στο Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων
Επιβλέπων: Καθηγητής ***Τ. Σελλής***

Περίληψη

1 Γενικό πλαίσιο

Αντικείμενο της διπλωματικής εργασίας είναι η διερεύνηση της εφαρμογής των διαγραμμάτων Voronoi για την διαχείριση *χωρικών ρευμάτων δεδομένων* (data streams) τα οποία προκύπτουν λ.χ. από τις καταγραφές αισθητήρων ή δεκτών GPS. Τα χωρικά ρεύματα δεδομένων δημιουργούνται από την καταγραφή των σημειακών θέσεων πολλών κινούμενων αντικειμένων τα οποία ανανεώνουν συχνά το στίγμα τους και έχουν τα εξής χαρακτηριστικά:

- Τα δεδομένα καταφθάνουν με μεγάλο και δυναμικά μεταβλητό ρυθμό σε πραγματικό χρόνο και έχουν πιθανώς απεριόριστο μέγεθος.

- Για την επεξεργασία τους απαιτείται η χρήση μόνο της κύριας μνήμης, αφού η αποθήκευσή τους επιφέρει ανεπιθύμητες καθυστερήσεις, γι' αυτό και δεν ενδείκνυται η χρήση συμβατικών χωρικών βάσεων δεδομένων.

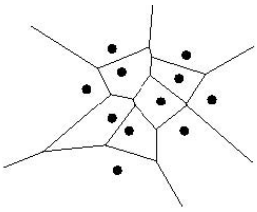
Ειδικότερα, μελετάται το ζήτημα της ταχείας επεξεργασίας *ερωτημάτων εγγύτερου γείτονα* (nearest neighbor) για στατικές εστίες, καθώς και ο online υπολογισμός μεμονωμένων κελιών *Voronoi* για δυναμικά κινούμενες εστίες. Για την περίπτωση των μεμονωμένων κελιών προτείνεται πρωτότυπος αλγόριθμος, που υπολογίζει και ανανεώνει ένα *προσεγγιστικό κελί Voronoi* k*-τάξεως* χρησιμοποιώντας ένα ρεύμα κινούμενων σημείων ως είσοδο. Ο αλγόριθμος που προτείνεται επιλέχθηκε να είναι προσεγγιστικός και χειρίζεται αποτελεσματικά τις περιπτώσεις ταυτόχρονης και ασύγχρονης ανανέωσης των θέσεων των αντικειμένων.

2 Διαχείριση ρευμάτων δεδομένων

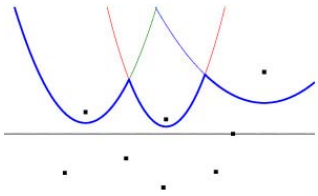
Σε συστήματα αποτελούμενα από μεγάλο αριθμό ανεξάρτητων κόμβων που αποστέλλουν δεδομένα, η είσοδος που καλείται να διαχειριστεί το κεντρικό υποσύστημα είναι συνήθως ένα μεγάλου όγκου και καταιγιστικού ρυθμού ρεύμα δεδομένων. Οι απαιτήσεις στην επεξεργασία ενός τέτοιου συνόλου δεδομένων διαφέρουν ριζικά από αυτές ενός συμβατικού συστήματος διαχείρισης δεδομένων, όπου η ακρίβεια στην απάντηση είναι το κύριο ζητούμενο. Τα *συστήματα διαχείρισης ρευμάτων δεδομένων* αναπτύχθηκαν για να απαντήσουν στο πρόβλημα αυτό, αποσκοπώντας πρωτίστως στην ικανοποίηση της απαίτησης για απόκριση σε πραγματικό χρόνο.

Το μοντέλο ρεύματος δεδομένων απαιτεί αλλαγές στον τρόπο επεξεργασίας των δεδομένων, χρησιμοποιώντας *ερωτήματα διαρκείας* των οποίων η απάντηση ανανεώνεται συνεχώς. Το πιθανώς άπειρο μέγεθος τόσο της εισόδου όσο και της εξόδου καθιστά ασύμφορη ή ακόμη και αδύνατη την αποθήκευση στοιχείων. Έτσι κρίνεται αναγκαία η επεξεργασία τους στην κύρια μνήμη και ύστερα η απόρριψή τους. Στην περίπτωση που χρειάζεται να διατηρείται ένα ιστορικό των δεδομένων χρησιμοποιούνται *τεχνικές προσέγγισης*, όπως είναι οι *περιλήψεις*, τα *σκίτσα δεδομένων* και τα *κυματίδια*.

Πολύ σημαντικές τεχνικές αποτελούν αυτές της αποβολής φόρτου και της εφαρμογής χρονικών παραθύρων στα δεδομένα. Η *αποβολή φόρτου* επιτρέπει την βελτίωση της χρονικής απόκρισης του συστήματος ακόμη και σε μεγάλου πλήθους δεδομένα, θυσιάζοντας ένα προκαθορισμένο ποσοστό της ποιό-



Σχήμα 1: Διάγραμμα Voronoi σημείων στο επίπεδο



Σχήμα 2: Η παραλιακή γραμμή

τητας της απάντησης με την απόρριψη μέρους των δεδομένων πριν την επεξεργασία τους. Το χρονικό παράθυρο επικεντρώνει συνήθως το ενδιαφέρον των ερωτημάτων κάθε φορά στα πιο πρόσφατα στοιχεία του ρεύματος.

3 Διαγράμματα Voronoi

Ιδιαίτερα σημαντική για την επεξεργασία σημειακών δεδομένων είναι η χρήση γεωμετρικών αλγορίθμων όπως το *κυρτό περίβλημα*, ο *τριγωνισμός Delaunay* και το *διάγραμμα Voronoi*. Το διάγραμμα *Voronoi* αποτελεί μια γεωμετρική λύση στο πρόβλημα της ανεύρεσης του εγγύτερου γείτονα και για ένα σύνολο στατικών εστιών στο επίπεδο, συνίσταται στην κατασκευή ενός πολυγώνου (κελιού) γύρω από κάθε εστία *ώστε κάθε σημείο εντός πολυγώνου να βρίσκεται πλησιέστερα προς την συγκεκριμένη εστία σε σύγκριση με οποιαδήποτε άλλη* (σχήμα 1) .

Στα πλαίσια της διπλωματικής μελετήθηκαν οι ιδιότητες των διαγραμμάτων και οι αλγόριθμοι κατασκευής τους. Ειδικότερα υλοποιήθηκε ο αλγόριθμος του *Fortune*, που αποτελεί και την βέλτιστη λύση για της κατασκευή του διαγράμματος στο επίπεδο.

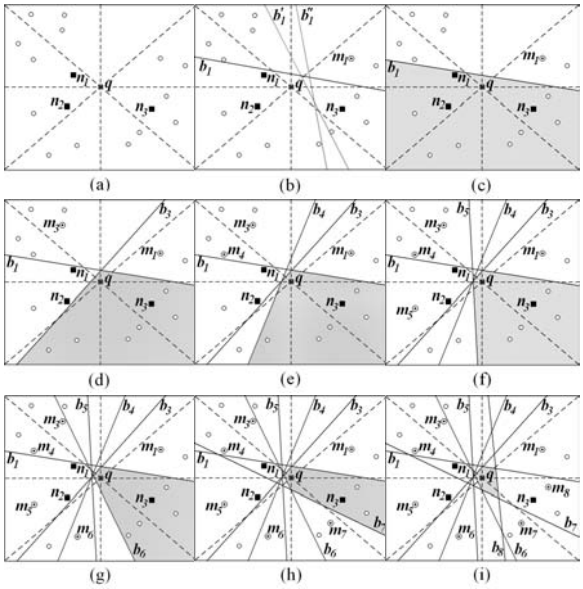
3.1 Ο αλγόριθμος του Fortune

Ο αλγόριθμος στηρίζεται στην τεχνική της *γραμμής σάρωσης* (sweep line) για την παραγωγή του διαγράμματος, σαρώνοντας το επίπεδο από πάνω προς τα κάτω και ενημερώνοντας σταδιακά τη δομή δεδομένων όπου τηρείται το διάγραμμα *Voronoi*. Επειδή το τμήμα του διαγράμματος που βρίσκεται πάνω από την γραμμή σάρωσης δεν είναι εξ ολοκλήρου γνωστό, για την παρακολούθηση της δημιουργίας του χρησιμοποιείται η *παραλιακή γραμμή* (beach line) που αποτελεί την ακολουθία τόξων που σχηματίζεται από την τομή των παραβολών που ορίζονται από τις γνωστές εστίες με διευθετούσα την γραμμή σάρωσης (σχήμα 2).

Η παραλιακή γραμμή αποτελεί το σύνορο του αμετάβλητου τμήματος του διαγράμματος *Voronoi* σε κάθε βήμα του αλγόριθμου. Όταν η γραμμή σάρωσης συναντήσει μια νέα εστία, συμβαίνει ένα *γεγονός εστίας* και ένα νέο παραβολικό τόξο πρέπει να εντοχθεί στην παραλιακή γραμμή. Αν η γραμμή σάρωσης συναντήσει το χαμηλότερο σημείο ενός κύκλου, ο οποίος ορίζεται από τρεις εστίες με διαδοχικά τμήματα στην παραλιακή γραμμή, ορίζεται ένα *γεγονός κύκλου* και το κέντρο του κύκλου προστίθεται ως κορυφή του διαγράμματος στην δομή τήρησης. Ο αλγόριθμος προϋποθέτει ότι όλα τα γεγονότα εστίας είναι γνωστά εξαρχής, ενώ τα γεγονότα κύκλου ανιχνεύονται κατά την εξέλιξή του.

3.2 Αξιολόγηση της μεθόδου

Για την πειραματική αξιολόγηση επιλέχθηκε, για σύνολα δυναμικά κινούμενων αντικειμένων, να απαντηθούν ερωτήματα εγγύτερου γείτονα με χρήση διαγραμμάτων Voronoi για διάφορα σύνολα στατικών εστιών. Για κάθε κινούμενο αντικείμενο εφαρμόσθηκε ο αλγόριθμος point in polygon, ο οποίος ελέγχει αν ένα σημείο βρίσκεται εντός κάποιου κελιού του διαγράμματος, οπότε η αντίστοιχη εστία αναγνωρίζεται ως ο εγγύτερος γείτονάς του. Από τις μετρήσεις προέκυψε ότι μικροί χρόνοι απόκρισης στα ερωτήματα και δυνατότητα κλιμάκωσης του ρυθμού εισόδου των κινούμενων αντικειμένων είναι δυνατοί ακόμη και για μεγάλα σύνολα σημείων. Τα αποτελέσματα αυτά επιβεβαιώνουν την αξία των γεωμετρικών αλγορίθμων στην απάντηση ερωτημάτων εγγύτερου γείτονα.



Σχήμα 3: Προσεγγιστικό κελί Voronoi $3^{\eta\varsigma}$ τάξεως.

4 Προσεγγιστικά κελιά Voronoi k-τάξεως

Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα *διαγράμματα Voronoi k-τάξεως* (order-k Voronoi diagrams) όπου τα κελιά του διαγράμματος προσδιορίζουν τα σημεία που βρίσκονται εγγύτερα στις ίδιες k εστίες. Υπάρχουν περιπτώσεις όπου η τήρηση ολόκληρου του διαγράμματος Voronoi αποτελεί δύσκολο εγχείρημα, όπως για παράδειγμα όταν οι εστίες είναι κινούμενες, όπου και απαιτείται ο εξαρχής υπολογισμός του διαγράμματος σε κάθε αλλαγή σημειακής θέσης μιας εστίας. Σε αυτές τις περιπτώσεις είναι πιο συμφέρουσα η μελέτη μεμονωμένων κελιών Voronoi.

Στα πλαίσια της εργασίας σχεδιάστηκε πρωτότυπος αλγόριθμος, ο οποίος υπολογίζει και ανανεώνει ένα προσεγγιστικό κελί Voronoi k-τάξεως σε διακριτές χρονικές στιγμές, όταν οι θέσεις των εστιών λαμβάνονται με την μορφή ρεύματος κινούμενων σημείων. Ο αλγόριθμος βασίζεται στην κατανομή του χώρου σε λ διακριτούς κυκλικούς τομείς γύρω από το σημείο ενδιαφέροντος Q. Τότε, για κάθε κυκλικό τομέα λαμβάνεται υπόψη μόνο ένας περιορισμένος αριθμός εστιών που ελέγχουν το σχήμα του κελιού για το Q. Η υλοποίηση του αλγόριθμου έγινε (i) για την περίπτωση της ταυτόχρονης ανανέωσης όλων των εστιών, αλλά και (ii) για το μοντέλο κυλιόμενου παραθύρου (sliding window) ως προς την διάρκεια ισχύος των σημειακών θέσεων.

4.1 Αλγόριθμος ταυτόχρονης ανανέωσης

Θεωρούμε ένα σύνολο $P = \{p_1, p_2, \ldots, p_m\}$ από N κινούμενα αντικείμενα στο επίπεδο, όπου κινείται και το σημείο ερωτήσεως Q. Ο αλγόριθμος στηρίζεται στην ιδέα της ισομερούς διαμέρισης του επιπέδου σε κυκλικούς τομείς με κέντρο το σημείο ερωτήσεως Q. Έτσι ο δισδιάστατος χώρος διαμερίζεται σε λ τομείς, χρησιμοποιώντας λ διανύσματα που ξεκινούν από το Q και έχουν ίση γωνιακή απόσταση μεταξύ τους $\theta = 2\pi/\lambda$.

Αφού δημιουργηθούν οι τομείς, το σύστημα δέχεται τις σημειακές θέσεις των κινούμενων αντικειμένων, υλοποιώντας μια μέθοδο *αποβολής φόρτου* (load shedding), η οποία επιτρέπει σε κάθε κυκλικό τομέα να αποθηκεύει το πολύ τις $k+1$ εγγύτερες σημειακές θέσεις στο Q. Ο αριθμός αυτός είναι το ελάχιστο όριο για το οποίο μπορούμε να είμαστε σίγουροι ότι στο τέλος του τρέχοντος *χρονοσήμου* (timestamp), θα υπάρχουν στο σύστημα οι k εγγύτεροι γείτονες που ζητήθηκαν και μια σημειακή θέση που θα δημιουργήσει το κελί.

Για κάθε μία σημειακή θέση που διαβάζεται από το ρεύμα εισόδου, εξετάζεται η θέση της στο επίπεδο και αντιστοιχίζεται σε έναν από τους κυκλικούς τομείς. Εάν ο τομέας έχει λιγότερες από $k+1$ αποθηκευμένες σημειακές θέσεις, τότε εισάγεται στην λίστα σημείων του. Σε αντίθετη περίπτωση μόνο αν βρίσκεται εγγύτερα στο Q από την πιο απομακρυσμένη σημειακή θέση της λίστας αποθηκεύεται παίρνοντας την θέση της, αλλιώς απορρίπτεται. Η διαδικασία αυτή γίνεται για όλες τις επόμενες σημειακές θέσεις και συνεχίζεται μέχρι να ανιχνευθεί χρονόσημο διαφορετικό από το τρέχον.

Στο τέλος κάθε χρονοσήμου ξεκινάει η διαδικασία κατασκευής του κελιού με την εύρεση των k εγγύτερων γειτόνων του Q από το σύνολο των ενεργών αντικειμένων στο σύστημα (σχήμα 3a). Στη συνέχεια για κάθε τομέα εντοπίζεται το *σημείο ελέγχου* που αποτελεί την εγγύτερη στο σημείο ερωτήσεως σημειακή θέση που δεν έχει σημαδευτεί στο προηγούμενο βήμα ως εγγύτερος γείτονας του Q (σχήμα 3, τα σημεία $m_1, \ldots, m_8$).

Δημιουργούμε πρώτα την πρώιμη μορφή του κελιού ως ένα παραλληλόγραμμο που περιλαμβάνει όλη

την περιοχή μελέτης. Στη συνέχεια δημιουργούμε το προσεγγιστικό κελί Voronoi, εξετάζοντας την περιοχή μελέτης και βρίσκοντας τις μεσοκαθέτους που ορίζουν τα ευθύγραμμα τμήματα που ενώνουν το σημείο ελέγχου κάθε τομέα, με κάθε έναν από τους εγγύτερους γείτονες (π.χ. σχήμα 3b) για το σημείο ελέγχου m_1).

Από αυτές τις μεσοκαθέτους για κάθε τομέα επιλέγεται αυτή που βρίσκεται σε μικρότερη απόσταση από το Q. Η μεσοκάθετος αυτή ονομάζεται *πιο περιοριστική μεσοκάθετος* για το σημείο ελέγχου και χρησιμοποιείται για να σχηματιστεί το κελί (σχήμα 3c), βρίσκοντας την τομή της με το υπάρχον σχήμα του κελιού. Η διαδικασία αυτή όταν γίνει για όλους τους τομείς, με ανθωρολογιακή φορά, παρέχει το προσεγγιστικό κελί για το τρέχον χρονόσημο (σχήμα 3d - 3i).

Ο αλγόριθμος όπως περιγράφηκε, βρίσκει εφαρμογή στην περίπτωση όπου τα κινούμενα αντικείμενα ανανεώνουν τις σημειακές τους θέσεις ταυτόχρονα σε κάθε χρονόσημο. Το σημείο ερωτήσεως Q θεωρείται ακίνητο. Στην περίπτωση που είναι κινητό, τότε ο αλγόριθμος σε κάθε μετακίνηση του Q πρέπει να προσαρμόζει την κατάτμηση του χώρου στην νέα σημειακή του θέση.

4.2 Αλγόριθμος χρονικά κυλιόμενου παραθύρου

Σε πολλές εφαρμογές πραγματικού χρόνου τα αντικείμενα που το σύστημα καλείται να παρακολουθήσει, δεν ανανεώνουν τις σημειακές τους θέσεις σύγχρονα, όπως για παράδειγμα αν το σύστημα τροφοδοτείται με στοιχεία κίνησης αυτοκινήτων στο οδικό δίκτυο μιας πόλης. Σε τέτοιες εφαρμογές όμως ιδιαίτερη αξία έχουν τα πιο πρόσφατα δεδομένα και η διατήρηση του συνόλου των δεδομένων είναι περιττή. Η εφαρμογή ενός χρονικά κυλιόμενου παραθύρου (λ.χ. θέσεις των αυτοκινήτων τα τελευταία 5 λεπτά) επιτρέπει στο σύστημα αφενός να διαχειριστεί τα δεδομένα αποτελεσματικότερα, ενώ ταυτόχρονα μειώνεται και ο φόρτος του δικτύου που αναλαμβάνει την μετάδοση των στοιχείων.

Δεχόμαστε ότι τα κινούμενα αντικείμενα έχουν σημειακές θέσεις με χρονική διάρκεια ισχύος μεγαλύτερη του ενός χρονοσήμου και ότι αυτές δεν ανανεώνονται για όλα τα αντικείμενα ταυτόχρονα. Έτσι στο εύρος των χρονοσήμων που καλύπτει το παράθυρο, υπάρχουν σημειακές θέσεις με διαφορετική υπολειπόμενη χρονική ισχύ, σενάριο που πλησιάζει περισσότερο τις συνθήκες μιας πραγματικής εφαρμογής. Για λόγους απλότητας γίνεται επίσης η παραδοχή ότι όλες οι σημειακές θέσεις έχουν χρονική ισχύ $\Delta\tau$ χρονόσημα, ίση με το εύρος του παραθύρου.

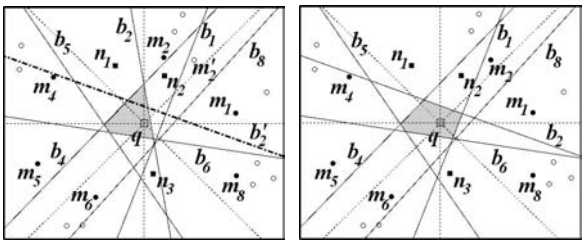
Ένα δεύτερο χαρακτηριστικό είναι η αλλαγή του μέγιστου αριθμού σημειακών θέσεων που αποθηκεύονται σε κάθε τομέα. Η λίστα σημειακών θέσεων κάθε τομέα δεν μπορεί να διατηρεί για κάθε χρονόσημο εντός παραθύρου αριθμό εγγραφών μεγαλύτερο από $k+1$. Ο περιορισμός αυτός μας εξασφαλίζει την ύπαρξη του κελιού για κάθε χρονόσημο, αλλά ταυτόχρονα μειώνει σημαντικά τον αποθηκευτικό και υπολογιστικό φόρτο.

Ο αλγόριθμος χωρίζεται σε δύο φάσεις, στην φάση αρχικοποίησης και στην φάση ανανέωσης. Την φάση αρχικοποίησης προηγείται η εισαγωγή ενός ελάχιστου αριθμού σημειακών θέσεων στο σύστημα, ίσου με $k+1$ σημειακές θέσεις, δηλαδή όσοι είναι οι εγγύτεροι γείτονες που αναζητούμε συν ένα. Η επιπρόσθετη σημειακή θέση απαιτείται για να χρησιμοποιηθεί στην δημιουργία ενός αρχικού κελιού των k υπολοίπων σημειακών θέσεων και επιλέγεται να είναι η πιο απομακρυσμένη από τις $k+1$. Για να επιτευχθεί καλύτερη απόκριση είναι δυνατή η εισαγωγή ενός μεγαλύτερου αριθμού σημειακών θέσεων.

Στην *φάση της αρχικοποίησης*, δημιουργείται ένα αρχικό κελί όπως ακριβώς έγινε με τον αλγόριθμο ταυτόχρονης ανανέωσης. Δηλαδή για σημειακές θέσεις που έχουν εισέλθει ως τώρα στο σύστημα γίνεται αναζήτηση των k εγγύτερων γειτόνων του Q και των *σημείων ελέγχου* κάθε τομέα.

Στη συνέχεια αρχίζει η *φάση της ανανέωσης* χωριστά για κάθε διακριτό χρονόσημο, όπου το σύστημα δέχεται και πάλι νέες σημειακές θέσεις. Αφού βρεθεί ο τομέας όπου αντιστοιχεί μια καινούργια σημειακή θέση p, εξετάζεται αν αυτή πληροί τις προϋποθέσεις εισαγωγής. Αν για το τρέχον χρονόσημο η λίστα σημειακών θέσεων δεν είναι πλήρης, η p εισάγεται εκεί. Αν όμως η λίστα είναι πλήρης, τότε γίνεται έλεγχος αν η απόρριψη της σημειακής θέσης p θα επηρεάσει την μορφή του κελιού στα επόμενα χρονόσημα. Αυτό συμβαίνει μόνο αν η p βρίσκεται εγγύτερα στο Q από κάποια άλλη ήδη αποθηκευμένη σημειακή θέση r και ταυτόχρονα η r έχει χρονόσημο μικρότερο από την p. Στην περίπτωση που βρεθούν πολλαπλές σημειακές θέσεις που πληρούν τα κριτήρια, επιλέγεται η πιο απομακρυσμένη και διαγράφεται, αφού αποκλείεται να επηρεάσει την μορφή του κελιού.

Η διαδικασία συνεχίζεται μέχρι να ανιχνευθεί χρονόσημο διαφορετικό από το τρέχον τ, οπότε και γί-



Σχήμα 4: Λήξη του σημείου m_2 και αντικατάσταση της μεσοκαθέτου του από αυτήν του m_2'

νεται η ανανέωση του υπάρχοντος κελιού. Για κάθε τομέα πρέπει να ελεγχθεί αν έχει αλλάξει το σημείο ελέγχου και αν κάποια σημειακή θέση του τομέα βρίσκεται πλησιέστερα στο Q από τους τωρινούς εγγύτερους γείτονες. Εάν βρεθεί ένας καινούργιος εγγύτερος γείτονας τότε ο αλγόριθμος επανεκκινεί από την φάση αρχικοποίησης, με τις υπάρχουσες σημειακές θέσεις.

Αν έχει αλλάξει μόνο το σημείο ελέγχου τότε είναι απαραίτητο να γίνει ανανέωση της πλευράς του κελιού που αντιστοιχεί στον τομέα αυτό. Για να επιτευχθεί αυτό χρειάζεται να αναζητηθεί και να αφαιρεθεί (αν υπάρχει) η πλευρά αυτή από το κελί. Το κελί επεκτείνεται και κλείνει είτε με την τομή των γειτονικών πλευρών (αν είναι συγκλίνουσες) είτε στο περιβάλλον παραλληλόγραμμο (αν είναι αποκλίνουσες). Στην συνέχεια το κελί εκλεπτύνεται βρίσκοντας την τομή του με την πιο περιοριστική μεσοκάθετο που ορίζεται από το νέο σημείο ελέγχου (σχήμα 4). Πριν την αρχή του επόμενου χρονοσήμου $\tau + 1$, πραγματοποιείται ολίσθηση του χρονικού παραθύρου με την αφαίρεση όλων των σημειακών θέσεων που έχουν λήξει.

Στην περίπτωση που το σημείο ερωτήσεως Q είναι ακίνητο, ο αλγόριθμος επαναλαμβάνεται μέχρι το πέρας του ρεύματος εισόδου. Αν και το Q αλλάζει σημειακή θέση (λογικά με ρυθμό κατά πολύ μικρότερο από αυτόν των κινούμενων αντικειμένων) τότε σε κάθε αλλαγή της πρέπει να εκκινηθεί ο αλγόριθμος από την αρχή. Αυτό γίνεται με την αντιγραφή των ενεργών σημειακών θέσεων σε μια προσωρινή λίστα, την δημιουργία νέας κατάτμησης για την νέα θέση του Q και την αναδιανομή των ενεργών σημειακών θέσεων σε αυτή. Ο αλγόριθμος συνεχίζει με την ανάγνωση των σημειακών θέσεων για το τρέχον χρονόσημο και την δημιουργία του αρχικού κελιού.

4.3 Αξιολόγηση της μεθόδου

Οι δύο παραλλαγές του αλγόριθμου ελέγχθηκαν πειραματικά για σύνολα δυναμικά κινούμενων αντικειμένων, που περιγράφουν την κίνηση ενός στόλου οχημάτων στο οδικό δίκτυο της Αθήνας.

Ο αλγόριθμος ταυτόχρονης ανανέωσης επιτυγχάνει πολύ καλή προσέγγιση του πραγματικού κελιού για μικρό αριθμό εγγύτερων γειτόνων με μικρή χρονική επιβάρυνση. Η μη τήρηση όλων των σημειακών θέσεων στο σύστημα εξασφαλίζει ότι οι απαιτήσεις σε αποθηκευτικό χώρο παραμένουν μικρές, στοιχείο σημαντικό για εφαρμογές διαχείρισης ρευμάτων δεδομένων.

Το μοντέλο χρονικού παραθύρου δίνει μια πολύ πρακτική εικόνα της απόδοσης του αλγόριθμου σε πραγματικές συνθήκες. Η απόκλιση από το πραγματικό κελί είναι αποδεκτή και σε αυτή την περίπτωση, ειδικά για μικρό αριθμό εγγύτερων γειτόνων που είναι και η συνηθισμένη περίπτωση σε πραγματικές εφαρμογές. Η χρονική απόκριση του αλγορίθμου και οι απαιτήσεις σε αποθηκευτικό χώρο κρίνονται επαρκείς.

5 Συμπεράσματα

Η διαχείριση ρευμάτων χωρικών δεδομένων αποτελεί μια ιδιαίτερη πρόκληση και ένα πεδίο με ιδιαίτερο ενδιαφέρον. Σημαντική στην αντιμετώπιση της πρόκλησης αυτής είναι η συμβολή των γεωμετρικών αλγορίθμων, που προσφέρουν κομψές και ιδιαίτερης δυναμικής λύσεις σε δύσκολα προβλήματα. Τόσο τα διαγράμματα *Voronoi* όσο και ο πρωτότυπος αλγόριθμος που προτείνεται επιβεβαιώνουν την άποψη αυτή, ανταποκρινόμενοι με επιτυχία σε πειραματικές συνθήκες που πλησιάζουν αυτές μιας πραγματικής εφαρμογής.

Μια πιθανή βελτίωση στον αλγόριθμο ταυτόχρονης ανανέωσης είναι να εφαρμοσθεί ένα ευρετηρίο όπως η *διαμέριση σε πλέγμα* (grid partitioning) στις σημειακές θέσεις του ρεύματος. Με αυτό τον τρόπο αποφεύγεται στην πλειοψηφία των περιπτώσεων η εξαντλητική αναζήτηση στο σύνολο των δεδομένων και γίνεται δυνατή η ταυτόχρονη επεξεργασία πολλαπλών σημείων ερωτήσεως.