ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

«ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΜΕΘΟΔΩΝ ΚΑΤΑΣΚΕΥΗΣ ΤΟΥ R* TREE ΜΕ ΟΡΟΥΣ ΑΠΟΔΟΤΙΚΟΤΗΤΑΣ ΓΙΑ ΕΡΩΤΗΜΑΤΑ ΚΟΝΤΙΝΟΤΕΡΟΥ ΓΕΙΤΟΝΑ ΣΕ ΠΟΛΥΔΙΑΣΤΑΤΟΥΣ ΧΩΡΟΥΣ ΔΕΔΟΜΕΝΩΝ»

του

ΑΠΟΣΤΟΛΙΔΗ ΕΥΛΑΜΠΙΟΥ ΤΟΥ ΕΥΑΓΓΕΛΟΥ

ΕΠΙΒΛΕΠΩΝ: ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ ΕΥΑΓΓΕΛΙΔΗΣ ΓΕΩΡΓΙΟΣ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος ειδίκευσης στα Πληροφοριακά Συστήματα

ΟΚΤΩΒΡΙΟΣ 2011

Ευχαριστίες:

Θα ήθελα να ευχαριστήσω θερμά τον κ. Γεώργιο Ευαγγελίδη για τη εμπιστοσύνη που έδειξε στο πρόσωπό μου, με την ανάθεση της παρούσας εργασίας. Μου έδωσε το κίνητρο και την ευκαιρία να ασχοληθώ με ένα πολύ ενδιαφέρον αντικείμενο, που αποτελεί αιχμή της τεχνολογίας και απασχολεί έντονα την ερευνητική κοινότητα σε όλο τον κόσμο. Καθ' όλη τη διάρκεια της προσπάθειας αποτέλεσε ένα πολύτιμο στήριγμα, παρέχοντας με προθυμία και θετική διάθεση σημαντική βοήθεια οποιαδήποτε στιγμή κι αν χρειάστηκε.

Θα ήθελα επίσης να εκφράσω τις ευχαριστίες μου στο πρόσωπο του κ. Μάριου Χατζηελευθερίου, ο οποίος από την μακρινή Αμερική, έκανε "αόρατη" τη διαφορά ώρας που μας χωρίζει και ανταποκρίθηκε άμεσα όποτε κι αν χρειάστηκε. Ο κ. Χατζηελευθερίου αποτέλεσε πηγή πολύτιμων συμβουλών και σκέψεων και ένας σημαντικός "συνεργάτης" σε αυτή την προσπάθεια, αφού πρόσφερε απλόχερα τον κώδικα που χρησιμοποιήθηκε στις πειραματικές μετρήσεις, μαζί με σημαντική βοήθεια για την επεξεργασία του, ώστε να εξυπηρετήσει τις ανάγκες της εργασίας.

Ευχαριστώ θερμά τους δύο αυτούς ανθρώπους για όλη τη συμπαράσταση, αφού η εργασία αυτή πραγματοποιήθηκε, έχοντας ως βασικό θεμέλιο την πολύτιμη συμβολή τους.

Περιεχόμενα

Εισαγωγή 1

ΚΕΦΑΛΑΙΟ 1: Ανασκόπηση Βιβλιογραφίας 5

1.1. Το R*-Tree 5

1.1.1. Τα Ελάχιστα Περιβάλλοντα Ορθογώνια 5

1.1.2. Το R-tree 7

1.1.2.1. Βασικές Αρχές 7

1.1.2.2. Βασικές Λειτουργίες 10

1.1.2.2.1. Αναζήτηση Δεδομένων 10

1.1.2.2.2. Εισαγωγή Δεδομένων 10

1.1.2.2.3. Διάσπαση Κόμβων 12

1.1.2.2.4. Διαγραφή Δεδομένων 14

1.1.2.2.5. Ενημέρωση Δεδομένων 15

1.1.3. Το R* tree 16

1.1.3.1. Παράμετροι και Κριτήρια Αποδοτικότητας 18

1.1.3.2. Παραλλαγές στο R*-tree 19

1.1.3.2.1. Αλγόριθμος ChooseSubtree 19

1.1.3.2.2. Διάσπαση του R*-tree 20

1.1.3.2.3. Εξαναγκασμένη Επανατοποθέτηση 22

1.2. Ερωτήματα Κοντινότερου Γείτονα 24

1.2.1. Μετρικά για Αναζήτηση Κοντινότερου Γείτονα 27

1.2.2. Αλγόριθμος Κοντινότερου Γείτονα 31

1.2.2.1. Ταξινόμηση και Κλάδεμα της Αναζήτησης 31

1.2.2.2. Αλγόριθμος Αναζήτησης Κοντινότερων Γειτόνων 35

1.2.2.3. Γενίκευση: Εύρεση των k Κοντινότερων Γειτόνων 35

1.3. Ομαδοποίηση Δεδομένων Εισαγωγής 37

1.3.1. Γενικός Αλγόριθμος 39

1.3.2. Ο Αλγόριθμος Sort-Tile-Recursive (STR) 40

1.4. Η "Κατάρα της Διαστατικότητας" 42

1.4.1. Το ερώτημα Κοντινότερου Γείτονα σε υψηλές διαστάσεις 44

ΚΕΦΑΛΑΙΟ 2: Μεθοδολογία προσέγγισης 49

ΚΕΦΑΛΑΙΟ 3: Ανάλυση Δεδομένων και Ερμηνεία Ευρημάτων 51

3.1. Παράμετροι Κατασκευής του R* tree 51

3.2. Δεδομένα Πειραματικής Διαδικασίας 55

3.3. Αποτελέσματα 56

ΜΕΡΟΣ 1: Κατώφλι Διάστασης Χώρου Δεδομένων 57

ΜΕΡΟΣ 2: Σύγκριση Μεθόδων Κατασκευής 63

ΚΕΦΑΛΑΙΟ 4: Συμπεράσματα και Προτάσεις 80

Αναφορές 83

ΠΑΡΑΡΤΗΜΑ Α: ΑΛΓΟΡΙΘΜΟΙ *R-TREE* 88

ΠΑΡΑΡΤΗΜΑ Β: ΑΛΓΟΡΙΘΜΟΙ *R*-TREE* 94

Κατάλογος Πινάκων

Πίνακας 1: Ποσοτική σχέση μεταξύ των φύλλων και των εσωτερικών κόμβων στο στη δομή του *R*-tree* για κάθε διάσταση. 58

Πίνακας 2: Τιμές *Index* και *Leaf Level Usage* για την πρώτη περίπτωση δεδομένων σε ερωτήματα 1NN, 10NN, 100NN και τις εξεταζόμενες διαστάσεις. 59

Πίνακας 3: Τιμές *Index* και *Leaf Level Usage* για τη δεύτερη περίπτωση δεδομένων σε ερωτήματα 1NN, 10NN, 100NN και τις εξεταζόμενες διαστάσεις. 60

Πίνακας 4: Τιμές *Index* και *Leaf Level Usage* για την τρίτη περίπτωση δεδομένων σε ερωτήματα 1NN, 10NN, 100NN και τις εξεταζόμενες διαστάσεις. 61

Πίνακας 5: Τιμές *Index* και *Leaf Level Usage* για την τέταρτη περίπτωση δεδομένων σε ερωτήματα 1NN, 10NN, 100NN και τις εξεταζόμενες διαστάσεις. 62

Πίνακας 6: Τιμές του *storage utilization* για τους δύο τρόπους εισαγωγής δεδομένων και όλες τις διαστάσεις. 64

Πίνακας 7: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 2 διαστάσεις. 67

Πίνακας 8: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 4 διαστάσεις. 68

Πίνακας 9: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 6 διαστάσεις. 69

Πίνακας 10: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 8 διαστάσεις. 71

Πίνακας 11: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 10 διαστάσεις. 72

Πίνακας 12: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 12 διαστάσεις. 73

Πίνακας 13: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για σημειακά αντικείμενα. 75

Πίνακας 14: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με σταθερό μήκος πλευράς (0.1% του άξονα συντεταγμένων)..... 76

Πίνακας 15: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με σταθερό μήκος πλευράς (1% του άξονα συντεταγμένων)..... 77

Πίνακας 16: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με τυχαίο μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1. 79

Κατάλογος Εικόνων

Εικόνα 1: Διατεταγμένο ευρετήριο. 3

Εικόνα 2: Το *MBR* στην περίπτωση των 2 διαστάσεων. 6

Εικόνα 3: Το *MBR* στην περίπτωση των 3 διαστάσεων. 6

Εικόνα 4: Χωρικές σχέσεις μεταξύ των *MBRs*. 7

Εικόνα 5: Οι παράμετροι *m* και *M* σε έναν κόμβο ή φύλλο του δένδρου. 7

Εικόνα 6: Το *R-tree* για τα δεδομένα του παραπάνω χώρου δύο διαστάσεων. 8

Εικόνα 7: Το *R-tree* για την περίπτωση χώρου δεδομένων τριών διαστάσεων. 9

Εικόνα 8: Αναζήτηση στο *R-tree*. 10

Εικόνα 9: Εισαγωγή στο *R-tree*. Αριστερά: το *R21* είναι το *MBR* του αντικειμένου που θα εισαχθεί στη δομή. Δεξιά: η μορφή της δομής μετά την εισαγωγή. 11

Εικόνα 10: Μια καλή και μια κακή επιλογή για διάσπαση κόμβων. 12

Εικόνα 11: Ο "νεκρός χώρος" σε ένα *MBR*. 13

Εικόνα 12: Διαγραφή στο *R-tree*. 14

Εικόνα 13: Τα *MBRs*, για το επίπεδο των φύλλων (πράσινο χρώμα), το επίπεδο εσωτερικών κόμβων (γαλάζιο χρώμα) και το επίπεδο της ρίζας (κόκκινο χρώμα). 17

Εικόνα 14: Η ιδιότητα της *MBR* όψης στο χώρο δύο διαστάσεων. 26

Εικόνα 15: Η ιδιότητα της *MBR* όψης στο χώρο τριών διαστάσεων. 26

Εικόνα 16: Τα *MINDIST* και *MINMAXDIST* στο χώρο δύο διαστάσεων. 30

Εικόνα 17: Τα *MINDIST* και *MINMAXDIST* στο χώρο τριών διαστάσεων. 30

Εικόνα 18: Η ταξινόμηση με χρήση του *MINDIST* μετρικού καταλήγει γρηγορότερα στον κοντινότερο γείτονα. 32

Εικόνα 19: Η ταξινόμηση με χρήση του *MINMAXDIST* μετρικού καταλήγει γρηγορότερα στον κοντινότερο γείτονα. 32

Εικόνα 20: Το *MBR R* απορρίπτεται από τη στιγμή που η απόσταση *MINDIST(P,R)* από το σημείο *P* είναι μεγαλύτερη από την απόσταση *MINMAXDIST(P,R')* του σημείου από το *MBR R'*. 33

Εικόνα 21: Το αντικείμενο *O* απορρίπτεται από τη στιγμή που η πραγματική του απόσταση *ACTUAL_DIST(P,O)* από το σημείο *P* είναι μεγαλύτερη από την απόσταση *MINMAXDIST(P,R')* του σημείου από το *MBR R'*. 34

Εικόνα 22: Το *MBR R* απορρίπτεται από τη στιγμή που η απόσταση *MINDIST(P,R)* από το σημείο *P* είναι μεγαλύτερη από την πραγματική απόσταση *ACTUAL_DIST(P,O)* του σημείου από το αντικείμενο *O*. 34

Εικόνα 23: Για την απόρριψη ενός *MBR* γίνεται η σύγκριση της απόστασής του *MINDIST(P,R)* από το σημείο *P*, με την πραγματική απόσταση *ACTUAL_DIST(P,O_k)* του σημείου από το *k*-οστό αντικείμενο O_k που βρίσκεται στη μνήμη. 36

Εικόνα 24: Τα ορθογώνια οριοθέτησης στο επίπεδο των φύλλων για τα *Long Beach Data* με τη χρήση του *STR*. 41

Εικόνα 25: Ο χώρος πιθανοτήτων στην περίπτωση των δύο διαστάσεων. 42

Εικόνα 26: Ο χώρος πιθανοτήτων στην περίπτωση των τριών διαστάσεων. 43

Εικόνα 27: Πυκνότητα σημείων για χώρο (a) μίας, (b) δύο, (c) τριών διαστάσεων. .. 44

Εικόνα 28: Το σημείο ερωτήματος και ο κοντινότερος γείτονας 44

Εικόνα 29: Το σημείο ερωτήματος και ο κοντινότερος γείτονας. 45

Εικόνα 30: Περιοχή ερωτήματος και μεγεθυμένη περιοχή. Με *DMIN* αναπαρίσταται η απόσταση του κοντινότερου γείτονα και *DMAX* η απόσταση του πιο απομακρυσμένου σημείου. 45

Εικόνα 31: Η Ομοιόμορφη κατανομή για τα εξεταζόμενα πλήθη δεδομένων. 46

Εικόνα 32: Όλες οι κατανομές για την περίπτωση 1.000.000 εγγραφών. 46

Εικόνα 33: Ερώτημα *NN* σε *clustered* δεδομένα. 48

Εικόνα 34: Η μορφή καταγραφής του *MBR* κάθε αντικειμένου στο αρχείο δεδομένων, για την περίπτωση του χώρου δύο διαστάσεων. 52

Εικόνα 35: Ο χώρος δεδομένων στις (α) δύο διαστάσεις και (β) τρεις διαστάσεις. 55

Εικόνα 36: Τιμές *Index* και *Leaf Level Usage* για τον πρώτο τύπο δεδομένων. 59

Εικόνα 37: Τιμές *Index* και *Leaf Level Usage* για το δεύτερο τύπο δεδομένων..... 60

Εικόνα 38: Τιμές *Index* και *Leaf Level Usage* για τον τρίτο τύπο δεδομένων. 61

Εικόνα 39: Τιμές *Index* και *Leaf Level Usage* για τον τέταρτο τύπο δεδομένων 62

Εικόνα 40: Μέγεθος των αρχείων ευρετηρίου για τις δύο μεθόδους εισαγωγής και όλες τις εξεταζόμενες διαστάσεις 64

Εικόνα 41: Πλήθος των *Disk Accesses* κατά την κατασκευή της δομής όταν τα δεδομένα εισάγονται σειριακά (αριστερά) και μαζικά (δεξιά). 65

Εικόνα 42: Πλήθος των *Disk Accesses* κατά την εισαγωγή του 20, 40, 60 και 80% των δεδομένων όταν εισάγονται σειριακά (αριστερά) και μαζικά (δεξιά) 65

Εικόνα 43: Τιμές μετρικών για χώρο δεδομένων 2 διαστάσεων. 67

Εικόνα 44: Τιμές μετρικών για χώρο δεδομένων 4 διαστάσεων. 68

Εικόνα 45: Τιμές μετρικών για χώρο δεδομένων 6 διαστάσεων. 69

Εικόνα 46: Τιμές μετρικών για χώρο δεδομένων 8 διαστάσεων. 70

Εικόνα 47: Τιμές μετρικών για χώρο δεδομένων 10 διαστάσεων. 72

Εικόνα 48: Τιμές μετρικών για χώρο δεδομένων 12 διαστάσεων. 73

Εικόνα 49: Τιμές μετρικών για την περίπτωση σημειακών αντικειμένων. 74

Εικόνα 50: Τιμές μετρικών για την περίπτωση ορθογωνίων με σταθερό μήκος πλευράς (0.1% του άξονα συντεταγμένων). 76

Εικόνα 51: Τιμές μετρικών για την περίπτωση ορθογωνίων με σταθερό μήκος πλευράς (1% του άξονα συντεταγμένων). 77

Εικόνα 52: Τιμές μετρικών για την περίπτωση ορθογωνίων με τυχαίο μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1. 78

Περίληψη

Το *R*-tree* αποτελεί μια δενδρική δομή ευρετηριοποίησης ορθογωνίων. Ως παραλλαγή του *R-tree*, σχεδιάστηκε ώστε να οδηγεί σε αποτελεσματικότερη τμηματοποίηση του χώρου, εξασφαλίζοντας λιγότερες επικαλύψεις τμημάτων και αυξάνοντας με αυτό τον τρόπο την απόδοση κατά την εκτέλεση ερωτημάτων. Στην παρούσα εργασία ερευνάται η λειτουργία των μεθόδων σειριακής και μαζικής εισαγωγής πολυδιάστατων δεδομένων στη δομή του *R*-tree*, μέσω μετρήσεων αποδοτικότητας κατά την εκτέλεση ερωτημάτων Κοντινότερου Γείτονα. Στόχος είναι, η οριοθέτηση της "διαστατικότητας" των δεδομένων έτσι ώστε να εξασφαλίζεται αποδοτική λειτουργία του *R*-tree*, καθώς και η ενδελεχής σύγκριση των δύο μεθόδων κατασκευής προκειμένου να οδηγηθούμε σε συμπεράσματα για την καταλληλότητα τους. Από τις μετρήσεις καταλήξαμε ότι η δομή διαχειρίζεται αποδοτικά δεδομένα 8 διαστάσεων, με επέκταση στις 12 διαστάσεις για κάποιες περιπτώσεις. Όσον αφορά τη σύγκριση των μεθόδων κατασκευής, οι μετρήσεις έδειξαν ότι η μαζική εισαγωγή δεδομένων υπερτερεί σημαντικά ως προς την κατασκευή, εξασφαλίζοντας σχεδόν βέλτιστη χρήση της μνήμης και δημιουργώντας μια πιο συμπαγή δομή, μικρότερου μεγέθους. Τα αποτελέσματα ωστόσο για την καταλληλότητα των μεθόδων κατά την εκτέλεση ερωτημάτων, ποικίλουν και σχετίζονται με τον τύπο και τη διάσταση των δεδομένων. Συνοψίζοντας, σε μικρές διαστάσεις (έως 4) οι μέθοδοι πετυχαίνουν παρόμοιες αποδόσεις, ενώ σε μεγαλύτερες διαστάσεις η σειριακή μέθοδος εμφανίζεται καταλληλότερη για διαχείριση "κανονικοποιημένων" χωρικών αντικειμένων (σημείων, ορθογωνίων με σταθερά μήκη πλευρών) και η μαζική μέθοδος σε πολύπλοκες μορφές (ορθογώνια με τυχαία μήκη πλευρών). Συνολικά, τα ευρήματα της εργασίας μπορούν να αποτελέσουν χρήσιμη πληροφορία για τη λειτουργία του *R*-tree* σε πολυδιάστατα δεδομένα, που θα συμβάλει σε μια συστημική και αιτιολογημένη επιλογή της καταλληλότερης μεθόδου εισαγωγής.

Εισαγωγή

Στις 25 Οκτωβρίου του 2007, ο πρωθυπουργός της Μεγάλης Βρετανίας *Gordon Brown* στα πλαίσια μιας διάλεξης στο Πανεπιστήμιο του *Westminster*, σχετικά με τον ορισμό και την έννοια της "ελευθερίας", δήλωσε πως *«αυτός είναι ο αιώνας της πληροφορίας»* (http://www.nationalarchives.gov.uk/documents/18nov08.pdf). Σε μια απλή φράση συμπυκνώνεται ολόκληρη η σημασία και η σπουδαιότητα της έννοιας της *"πληροφορίας"* και διαφαίνεται ο καθοριστικός ρόλος που θα έχει στη ζωή των ανθρώπων στο κατώφλι του 21^{ου} αιώνα.

Στις δεκαετίες που πέρασαν, η ραγδαία τεχνολογική εξέλιξη των υπολογιστικών συστημάτων και των μέσων καταγραφής και αποθήκευσης πληροφορίας οδήγησαν στη δημιουργία των πρώτων Βάσεων Δεδομένων (ΒΔ). Οι τελευταίες με την ανάπτυξη Συστημάτων Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ), αποτέλεσαν ένα χρήσιμο και ισχυρό εργαλείο για την εύκολη αποθήκευση και αναζήτηση πληροφορίας, γεγονός που τις καθιέρωσε γρήγορα στον χώρο των υπολογιστικών συστημάτων. Η λειτουργία τους βασίζεται στην καταγραφή ενός πλήθους αριθμητικών και αλφαριθμητικών τιμών που αναπαριστούν τα χαρακτηριστικά μιας εγγραφής, βάσει των οποίων γίνεται στη συνέχεια η αναζήτηση και η εκτέλεση ερωτημάτων.

Τα τελευταία χρόνια, η λειτουργία των ΒΔ επεκτάθηκε σε νέα πεδία εφαρμογών με τη δημιουργία των Χωρικών Βάσεων Δεδομένων (ΧΒΔ). Για το σκοπό αυτό, οι ΒΔ ενσωμάτωσαν νέες δυνατότητες αποθήκευσης που αφορούν αντικείμενα στο χώρο όπως σημεία, γραμμές και πολύγωνα, καταγράφοντας πληροφορία που αφορά τη θέση και τη γεωμετρία τους. Αντίστοιχα τα νέα Συστήματα Διαχείρισης Χωρικών Βάσεων Δεδομένων (ΣΔΧΒΔ) υιοθέτησαν Μεθόδους Χωρικής Προσπέλασης *(Spatial Access Methods, SAMs)* που υποστηρίζουν την επεξεργασία των δεδομένων. Η λειτουργία των ΧΒΔ υιοθετήθηκε από ένα νέο πεδίο εφαρμογών το οποίο αποτελείται από:

- Γεωγραφικά Πληροφοριακά Συστήματα *(Geographical Information Systems)*
- Σχεδίαση με χρήση Υπολογιστή *(Computer-aided design)*
- Πολυμεσικές Βάσεις Δεδομένων *(Multimedia Data Bases)*
- Βάσεις Δεδομένων Εικόνων *(Image Data Bases)*
- Βάσεις Δεδομένων Χρονοσειρών *(Time Series Data Bases)*

Με τη βοήθεια των *SAMs*, τα ΣΔΧΒΔ μπορούν να υποστηρίξουν τους παρακάτω τύπους ερωτημάτων:

Εφαρμογές GIS και CAD

- *Ερωτήματα Εύρους:* δεδομένης μιας περιοχής στο χώρο, να βρεθούν όλα τα χωρικά αντικείμενα που βρίσκονται μέσα σε αυτή.
- *Ερωτήματα Σημείου:* δεδομένου ενός σημειακού αντικείμενου, να βρεθεί η περιοχή ή οι περιοχές που το περιλαμβάνουν.
- *Ερωτήματα Κοντινότερου Γείτονα:* δεδομένου ενός σημειακού αντικειμένου, να βρεθούν τα k κοντινότερα αντικείμενα στο χώρο δεδομένων, με το $k \in \mathrm{N}$.
- *Ερωτήματα Χωρικής Σύζευξης:* να βρεθούν όλα τα ζεύγη χωρικών αντικειμένων που ικανοποιούν μια συνθήκη.
- *Ερωτήματα Τομής:* δεδομένου ενός χωρικού αντικειμένου, να βρεθούν τα χωρικά αντικείμενα που τέμνονται με αυτό.

Εφαρμογές Image Data Base (Biomedical Data)

- *Ομοιότητα Εικόνων:* δεδομένης μιας εικόνας να βρεθούν εικόνες με παρόμοια κατανομή χρωμάτων.
- *Αναγνώριση Προτύπων:* αναγνώριση προσώπου, ίριδας ματιού, δακτυλικού αποτυπώματος, ανίχνευση βλάβης (π.χ. ιατρικές εικόνες) υπολογίζοντας μετρικά ομοιότητας με αντικείμενα της ΒΔ.

Εφαρμογές Time Series Data Base (Financial Data)

- *Ομοιότητα Χρονοσειρών:* δεδομένης μια χρονοσειράς να βρεθούν χρονοσειρές της ΒΔ που εμφανίζουν υψηλή ομοιότητα με αυτή.

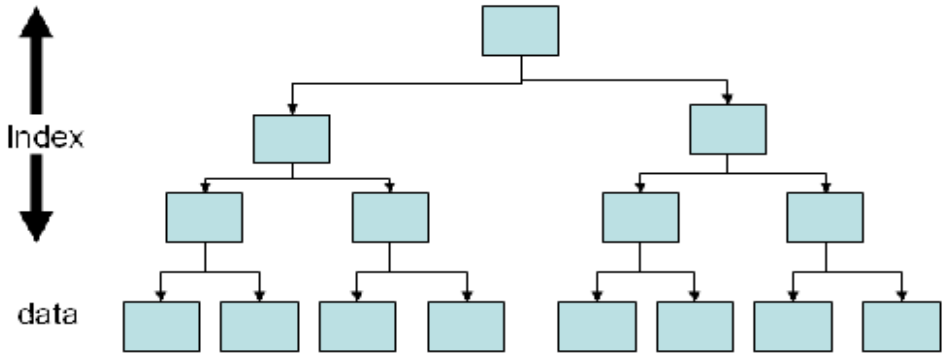
Εφαρμογές Multimedia Data Base

- *Αναγνώριση Φωνής:* αναζήτηση των χαρακτηριστικών συχνοτήτων μιας ηχητικής χρονοσειράς και αντιστοίχιση σε κάποιο από τα αντικείμενα της ΒΔ, βάσει της ομοιότητάς του.

Ένας βασικός παράγοντας της αποτελεσματικότητας των ΧΒΔ, είναι ο χρόνος που απαιτείται για την απάντηση σε ένα ερώτημα που θέτει ο χρήστης. Αντίστοιχα με τα ΣΔΒΔ και στοχεύοντας στη μείωση του χρόνου απόκρισης, τα ΣΔΧΒΔ ανέπτυξαν

μεθόδους ευρετηριοποίησης των δεδομένων, προκειμένου κατά την εκτέλεση ενός ερωτήματος να προσπελαύνουν ένα τμήμα και όχι όλα τα δεδομένα της ΒΔ. Μερικές από τις δομές ευρετηριοποίησης που αναπτύχθηκαν είναι οι*: Grid, Z-order, Quadtree, Octree, UB-tree, R-tree, R+-tree, R*-tree, Hilbert R-tree, X-tree, kd-tree* και *m-tree (*http://en.wikipedia.org/wiki/Spatial_access_method#Spatial_Index). Οι παραπάνω υλοποιήθηκαν ώστε να λειτουργούν στον Ευκλείδειο Χώρο των δύο διαστάσεων, αλλά κάποιες από αυτές επιτρέπουν τη γενίκευση σε περισσότερες διαστάσεις. Ωστόσο, ένα από τα μειονεκτήματα των *SAMs* είναι η σημαντική μείωση της αποδοτικότητάς τους όταν αυξάνονται οι διαστάσεις του χώρου των δεδομένων, ένα φαινόμενο που αναφέρεται στη σχετική βιβλιογραφία ως "Κατάρα της Διαστατικότητας".

Από τις παραπάνω δομές επιλέξαμε να μελετήσουμε το *R*-tree* (Beckmann et. al. (1990)). Η δομή αυτή αποτελεί την εξέλιξη του *R-tree* (Guttman (1984)) και ανήκει στην ευρύτερη κατηγορία των δενδρικών δομών. Η υλοποίησή του δημιουργεί ένα διατεταγμένο ευρετήριο με τη μορφή που φαίνεται στην Εικόνα 1. Τα επίπεδα της ρίζας και των εσωτερικών κόμβων συνθέτουν το ευρετήριο, ενώ το επίπεδο των φύλλων αντιστοιχίζεται στα δεδομένα της ΧΒΔ.



Εικόνα 1: Διατεταγμένο ευρετήριο.

(www.cs.sjsu.edu/faculty/lee/cs157/Data%20Indexing%20Presentation.ppt)

Η αποτελεσματικότητά και η χρησιμότητα του *R*-tree* αποδεικνύεται από το γεγονός ότι έχει ενσωματωθεί σε πολλά εμπορικά ΣΔΒΔ *(Informix, Oracle)* αλλά και από τη συνεχή έρευνα που γίνεται για τη βελτιστοποίησή του και την δημιουργία παραλλαγών όπως το *X-tree* (Berchold et. al. (1996)), το *SOM-based R*-tree* (Oh et. al. (2001)), το *SR-tree* (Katayama και Satoh (1997)) και το $R^o - tree$ (Xia και Zhang (2005)), αν και η αρχική του μορφή περιγράφηκε από τον Beckmann (1990).

Σκοπός της παρούσας εργασίας είναι να εξεταστεί η λειτουργία της δομής σε ερωτήματα Κοντινότερου Γείτονα, μεταβάλλοντας δύο βασικές παραμέτρους. Η πρώτη αφορά τη διάσταση του χώρου δεδομένων και λαμβάνει τιμές από 2 έως 16 και η δεύτερη σχετίζεται με τον τρόπο κατασκευής της δομής. Όπως θα περιγράφει στην Ενότητα 1.1. το *R*-tree* κατασκευάζεται σταδιακά κατά την εισαγωγή των δεδομένων και αποτελεί μια δυναμική δομή, η οποία επιτρέπει εισαγωγές και διαγραφές στοιχείων σε κάθε στάδιο της δημιουργίας του. Ωστόσο στην περίπτωση που τα δεδομένα είναι στατικά και δεν τροποποιούνται μετά τη δημιουργία τους, υπάρχει η δυνατότητα ομαδοποίησής και μαζικής εισαγωγής στο *R*-tree*, ώστε να βελτιωθεί η δομή του και να αυξηθεί η απόδοση. Με βάση τα παραπάνω, εξετάσαμε τη λειτουργία της δομής όταν τα δεδομένα εισάγονται σειριακά *(serial load)* και μαζικά *(bulk load)*. Στο σημείο αυτό αξίζει να αναφερθεί ότι έχουν υλοποιηθεί αντίστοιχα πειράματα για σειριακή εισαγωγή δεδομένων και τα αποτελέσματα παρουσιάζονται στις εργασίες για τα *X tree* (Berchold et. al. (1996)) και *SS tree* (White και Jain (1996)). Ωστόσο τα αποτελέσματά τους είναι χειρότερα από αυτά που προέκυψαν από τις πειραματικές μετρήσεις της παρούσας εργασίας. Επίσης, αντίστοιχα πειράματα έχουν γίνει για το *Hilbert R-tree* (Kamel και Faloutsos (1994)) που αποτελεί την επέκταση του *R-tree* σε υψηλότερες διαστάσεις.

Στόχος της πειραματικής διαδικασίας ήταν η εξαγωγή συμπερασμάτων για την αποδοτικότητα της δομής, με βάση τις δύο παραμέτρους που εξετάσαμε. Όσον αφορά την πρώτη, επιδιώκουμε να οδηγηθούμε σε ένα κατώφλι για τις διαστάσεις του χώρου δεδομένων, στις οποίες το *R*-tree* να επιτρέπει την αποτελεσματική εκτέλεση ερωτημάτων Κοντινότερου Γείτονα. Σχετικά με τη δεύτερη παράμετρο, προσπαθούμε να οδηγηθούμε σε ένα συμπέρασμα σχετικά με τον καταλληλότερο τρόπο κατασκευής του *R*-tree*, στην περίπτωση που τα δεδομένα της ΧΒΔ είναι στατικά και δεν μεταβάλλονται. Συνολικά ο στόχος όλης της ανάλυσης που ακολουθεί, είναι η μελέτη της χρήσης του *R*-tree* ως μιας δομής ευρετηριοποίησης πολυδιάστατων δεδομένων, προκειμένου να αποτελέσει ένα εργαλείο για την αποδοτική Εξόρυξη Γνώσης *(Knowledge Mining)* από μια Αποθήκη Δεδομένων *(Data Warehouse)*.

ΚΕΦΑΛΑΙΟ 1: Ανασκόπηση Βιβλιογραφίας

Στην παρούσα ενότητα αναλύεται το θεωρητικό υπόβαθρο για την υλοποίηση της ερευνητικής διαδικασίας. Πιο συγκεκριμένα, περιγράφεται η δομή και η λειτουργία του *R*-tree*, ο αλγόριθμος του ερωτήματος Κοντινότερου Γείτονα, ο αλγόριθμος της Sort-Tile-Recursive για την ομαδοποίηση και τη μαζική εισαγωγή δεδομένων στη δομή και η "κατάρα της διαστατικότητας".

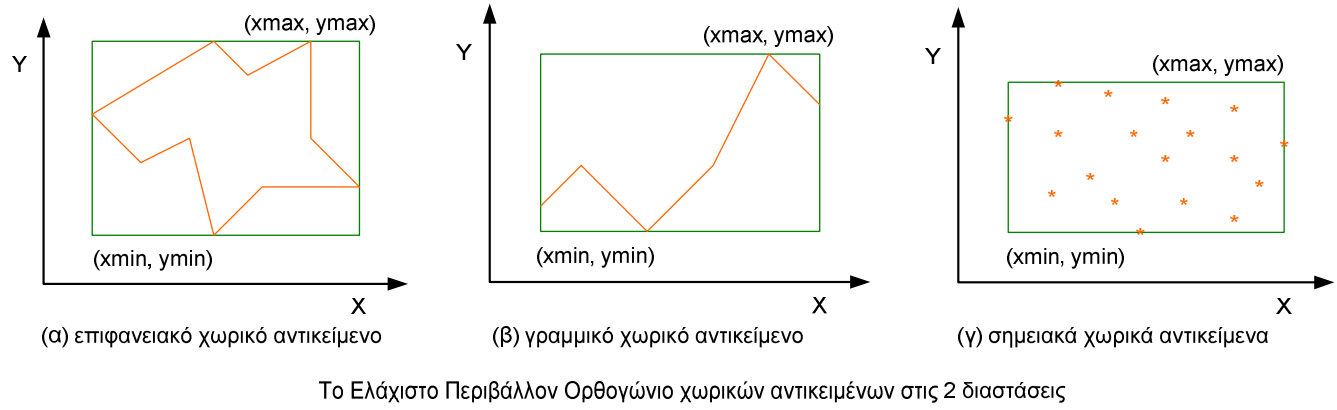
1.1. Το R*-Tree

Το *R*-tree* παρουσιάστηκε για πρώτη φορά από τον Beckmann et. al. (1990) και η υλοποίησή του βασίστηκε στη θεωρία που είχε αναπτυχθεί για το *R-tree,* από τον Guttman (1984). Οι συγγραφείς μελέτησαν τον τρόπο κατασκευής, τις ιδιότητες και τις λειτουργίες της δομής, στοχεύοντας στη βελτιστοποίηση κάποιων διαδικασιών και στο σχεδιασμό μιας νέας δομής, η οποία θα κάνει αποτελεσματικότερα την ευρετηριοποίηση και την αναζήτηση χωρικών δεδομένων. Η μελέτη λοιπόν του *R-tree* αποτελεί ένα χρήσιμο και απαραίτητο θεωρητικό υπόβαθρο για την ανάλυση της δομής και του τρόπου κατασκευής του *R*-tree*. Στην παρούσα ενότητα περιγράφεται αρχικά η θεωρία που αναπτύχθηκε από τον Gutmann (1984) και στη συνέχεια γίνεται η επέκταση στο μοντέλο που εισήγαγε ο Beckmann et. al. (1990). Ωστόσο, οι δομές αυτές χρησιμοποιούν ως θεμέλιο λίθο την έννοια του Ελάχιστου Περιβάλλοντος Ορθογωνίου *(Minimum Bounding Rectangle, MBR)* και για το λόγο αυτό η ανάλυσή θα ξεκινήσει με την περιγραφή της σχετικής θεωρίας.

1.1.1. Τα Ελάχιστα Περιβάλλοντα Ορθογώνια

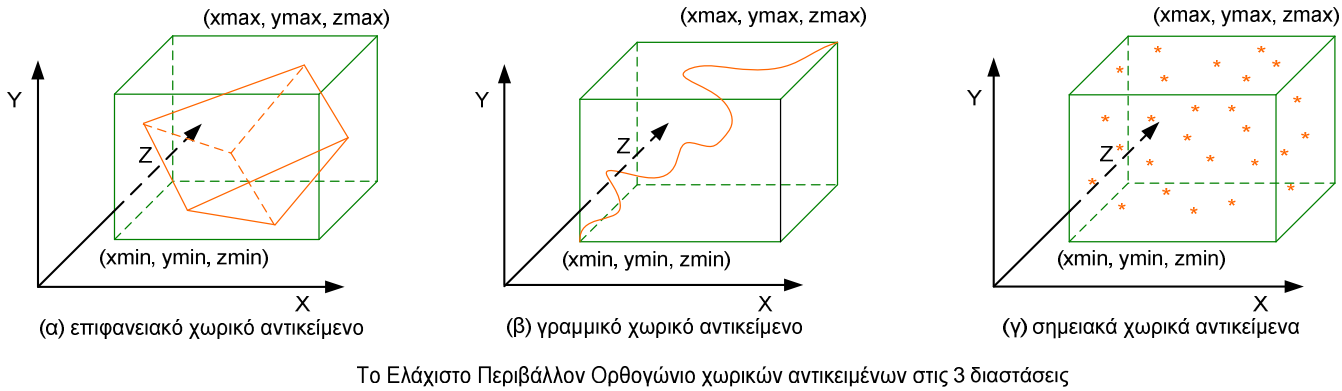
Το *R*-tree* όπως και το *R-tree* ανήκει στην οικογένεια των *SAMs*, που βασίζονται στην προσέγγιση ενός πολύπλοκου χωρικού αντικειμένου από ένα Ελάχιστό Περιβάλλον Ορθογώνιο *(Minimum Bounding Rectangle, MBR)* με πλευρές παράλληλες στους άξονες του χώρου δεδομένων. Ως αποτέλεσμα, η αναπαράσταση του αντικειμένου γίνεται πλέον από έναν περιορισμένο αριθμό *bytes*. Η πληροφορία που αποθηκεύεται

αποτελείται από την τιμή των συντεταγμένων της κάτω αριστερά και της πάνω δεξιά κορυφής του *MBR* που το περιβάλλει. Πιο συγκεκριμένα, στην περίπτωση του *2D* χώρου δεδομένων καταγράφεται η τιμή των $(x_{\min}, y_{\min})$ και $(x_{\max}, y_{\max})$ και περίπτωση του *3D* χώρου δεδομένων η τιμή των $(x_{\min}, y_{\min}, z_{\min})$ και $(x_{\max}, y_{\max}, z_{\max})$ αντίστοιχα. Την ίδια λογική ακολουθεί και η περίπτωση του χώρου υψηλότερων διαστάσεων. Τα χωρικά αντικείμενα μπορούν να ποικίλουν και να είναι (α) επιφάνειες, (β) γραμμές ή καμπύλες και (γ) σημεία. (Εικόνες 2 και 3).



(α) επιφανειακό χωρικό αντικείμενο (β) γραμμικό χωρικό αντικείμενο (γ) σημειακά χωρικά αντικείμενα

Εικόνα 2: Το *MBR* στην περίπτωση των 2 διαστάσεων.

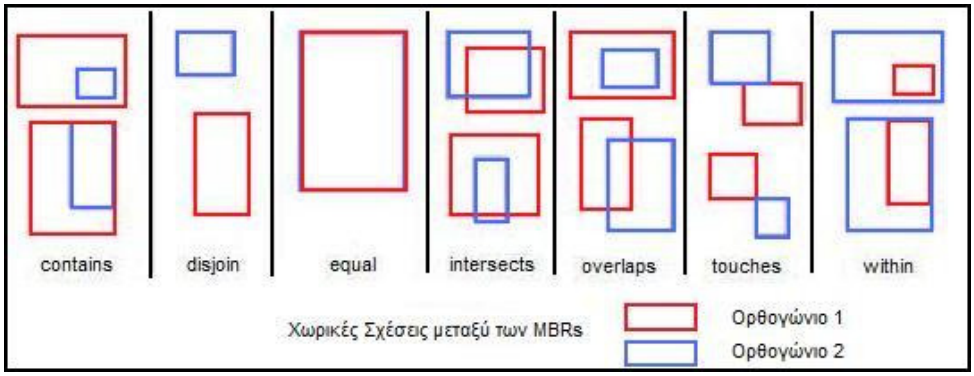


(α) επιφανειακό χωρικό αντικείμενο (β) γραμμικό χωρικό αντικείμενο (γ) σημειακά χωρικά αντικείμενα

Εικόνα 3: Το *MBR* στην περίπτωση των 3 διαστάσεων.

Παρ' ότι με αυτή την προσέγγιση χάνεται αρκετή πληροφορία σχετικά με τη λεπτομερή μορφή των χωρικών αντικειμένων, τα *MBRs* διατηρούν αποτελεσματικά τις πιο ουσιώδεις γεωμετρικές ιδιότητες τους, δηλαδή τη θέση και την έκτασή τους σε κάθε άξονα. Έτσι λοιπόν, η αναπαράσταση με *MBRs* μετέτρεψε το πρόβλημα της διαχείρισης πολύπλοκων γεωμετρικών σχημάτων στην απλούστερη διαχείριση ορθογωνίων. Με αυτό τον τρόπο καθίσταται ευκολότερος ο υπολογισμός διαφόρων

σχέσεων μεταξύ των χωρικών αντικειμένων (Εικόνα 4), καθώς επίσης και η εκτέλεση άλλων διαδικασιών όπως η ταξινόμηση και η ευρετηριοποίησή τους.



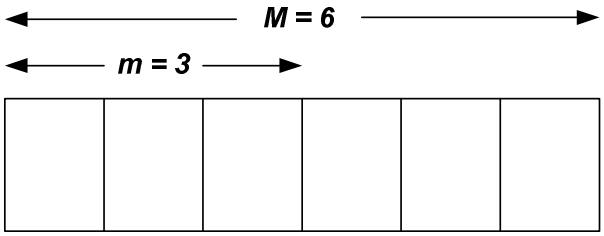
Εικόνα 4: Χωρικές σχέσεις μεταξύ των *MBRs*.

1.1.2. Το *R-tree*

Το *R-tree* είναι παρόμοιο με τη δομή του *B+-tree* (Comer (1979)) και ουσιαστικά αποτελεί την επέκτασή του σε δύο ή υψηλότερες διαστάσεις. Βασικό χαρακτηριστικό της δομής αποτελεί το γεγονός ότι αποθηκεύει πολυδιάστατα ορθογώνια σαν πλήρη αντικείμενα, χωρίς να τα "κουρεύει" ή να τα μετασχηματίζει σε σημεία υψηλότερης διάστασης.

1.1.2.1. Βασικές Αρχές

Αρχικά γίνεται η θεώρηση ότι οι παράμετροι M και m αναπαριστούν το μέγιστο και τον ελάχιστο αριθμό καταχωρήσεων σε έναν κόμβο ή φύλλο του δένδρου αντίστοιχα, όπως φαίνεται στην Εικόνα 5. Η σχέση που συνδέει τις δύο παραμέτρους είναι: $2 \le m \le M/2$

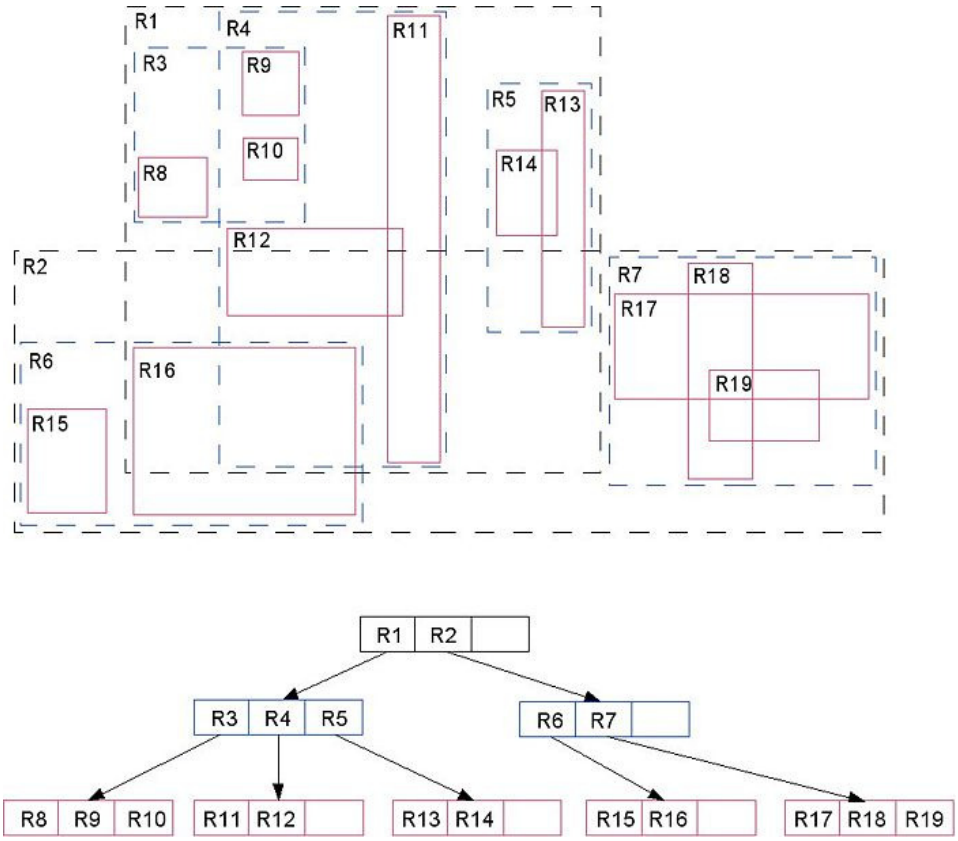


Εικόνα 5: Οι παράμετροι m και M σε έναν κόμβο ή φύλλο του δένδρου.

Με βάση τα παραπάνω, οι βασικές ιδιότητες του *R-tree* είναι οι ακόλουθες:

- Η ρίζα έχει τουλάχιστον δύο παιδιά, εκτός εάν είναι φύλλο.
- Κάθε εσωτερικός κόμβος περιλαμβάνει πλήθος παιδιών μεταξύ του *m* και του *M* εκτός εάν είναι η ρίζα.
- Κάθε φύλλο περιλαμβάνει πλήθος καταχωρήσεων μεταξύ του *m* και του *M* εκτός εάν είναι η ρίζα.
- Όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο (ισοζυγισμένο δένδρο).
- Ένας εσωτερικός κόμβος του δένδρου, περιλαμβάνει καταχωρήσεις της μορφής (*cp*, *Rectangle*) όπου *cp* είναι ένας δείκτης στον κόμβο-παιδί και *Rectangle* είναι το *MBR* που περικλείει τα ορθογώνια του κόμβου αυτού.
- Ένα φύλλο του δένδρου περιλαμβάνει καταχωρήσεις της μορφής (*Oid, Rectangle)* όπου το *Oid* είναι ένας δείκτης σε ένα χωρικό αντικείμενο της Βάσης Δεδομένων και το *Rectangle* είναι το *MBR* που περικλείει το αντικείμενο αυτό.

Οι ιδιότητες αυτές, γίνονται ευκολότερα κατανοητές από την Εικόνα 6.

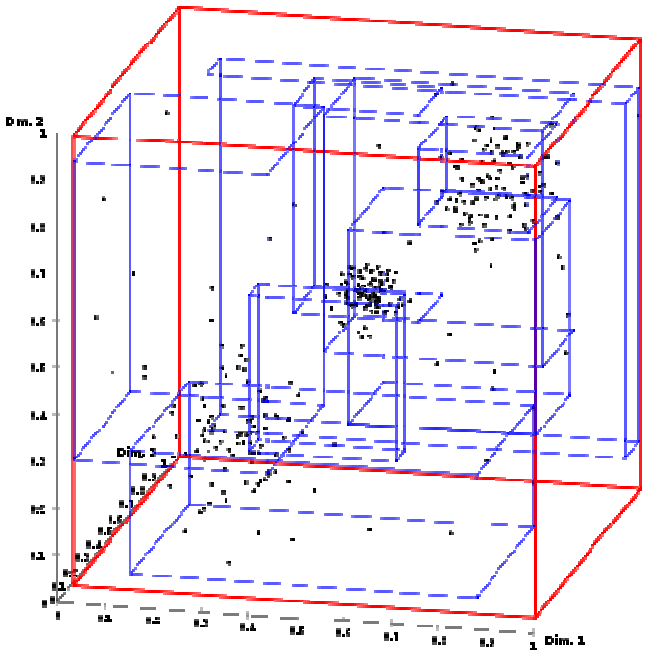


Εικόνα 6: Το *R-tree* για τα δεδομένα του παραπάνω χώρου δύο διαστάσεων.

(http://en.wikipedia.org/wiki/R-tree)

Τα ορθογώνια με κόκκινο χρώμα αποτελούν τα *MBRs* των δυσδιάστατων χωρικών αντικειμένων της ΒΔ. Για το λόγο αποθηκεύονται στα φύλλα του δένδρου. Κάθε ένα από αυτά περικλείει ένα πλήθος αντικειμένων που κυμαίνεται μεταξύ *m* και *M*. Τα ορθογώνια με γαλάζιο χρώμα αποτελούν τα *MBRs* του αμέσως ανώτερου επιπέδου του δένδρου. Καθένα από αυτά αποτελεί το *MBR* μιας ομάδας ορθογωνίων μεταξύ *m* και *M* του επιπέδου των φύλλων. Τέλος, τα ορθογώνια με μαύρο χρώμα αποτελούν αντίστοιχα τα *MBRs* της ρίζας του δένδρου τα οποία περικλείουν αντίστοιχα τα *MBRs* των εσωτερικών κόμβων του αμέσως κατώτερου επιπέδου του δένδρου.

Στην εικόνα που ακολουθεί, φαίνονται τα *MBRs* της δομής του *R-tree*, για την περίπτωση χώρου δεδομένων τριών διαστάσεων. Το σχήμα αυτό έχει δημιουργηθεί με τη βοήθεια του ELKI (http://elki.dbs.ifi.lmu.de/).



Εικόνα 7: Το *R-tree* για την περίπτωση χώρου δεδομένων τριών διαστάσεων.

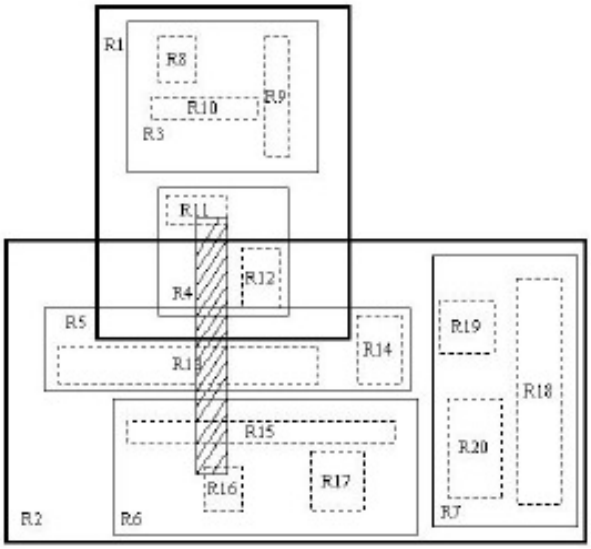
Ένα ακόμη βασικό χαρακτηριστικό του *R-tree* είναι η "δυναμική" του δομή, που επιτρέπει οι εισαγωγές και διαγραφές δεδομένων στο δένδρο να παρεμβάλλονται ανά πάσα στιγμή από ερωτήματα χωρίς να απαιτείται περιοδική καθολική αναδιοργάνωση της δομής. Γι' αυτό το λόγο το *R-tree* επιτρέπει επικαλυπτόμενα *MBRs* στους εσωτερικούς κόμβους του δένδρου και κατά συνέπεια, δεν εγγυάται ότι μόνο μια διαδρομή αναζήτησης είναι απαραίτητη για την εύρεση της απάντησης ενός ερωτήματος.

1.1.2.2. Βασικές Λειτουργίες

Οι βασικές λειτουργίες που υποστηρίζει το *R-tree* είναι (α) η αναζήτηση δεδομένων, (β) η εισαγωγή δεδομένων, (γ) η διαγραφή δεδομένων και (δ) η ενημέρωση δεδομένων. Στη συνέχεια περιγράφεται η μεθοδολογία για κάθε μια από αυτές, με τους αντίστοιχους αλγορίθμους να παρατίθενται στο Παράρτημα Α.

1.1.2.2.1. Αναζήτηση Δεδομένων

Κατά την αναζήτηση ο αλγόριθμος επιστρέφει όλα τα δεδομένα της ΧΒΔ που επικαλύπτονται από το ορθογώνιο ερωτήματος (Εικόνα 8). Ο αλγόριθμος αναζήτησης διασχίζει το δένδρο ξεκινώντας από τη ρίζα, με ένα τρόπο παρόμοιο με την αναζήτηση στο *B-tree* (Comer (1979)) (Παράρτημα Α). Ωστόσο, τα υπο-δένδρα που πρέπει να εξεταστούν κάθε φορά μπορεί να είναι περισσότερα του ενός και αιτία γι' αυτό αποτελεί το γεγονός ότι το *R-tree* επιτρέπει την επικάλυψη μεταξύ των *MBRs* στους εσωτερικούς κόμβους του δένδρου όπως προαναφέρθηκε.



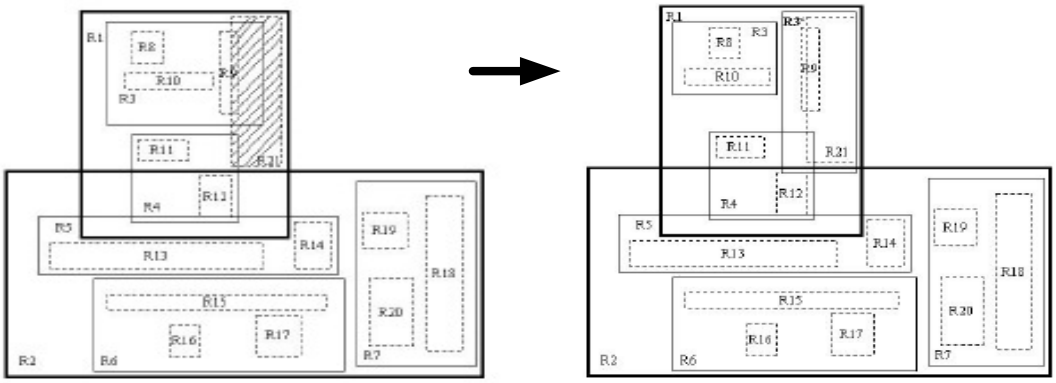
Εικόνα 8: Αναζήτηση στο *R-tree*.
(http://students.fim.uni-passau.de/~kaebisch/Proseminar/Proseminar.pdf)

1.1.2.2.2. Εισαγωγή Δεδομένων

Όπως η διαδικασία αναζήτησης, έτσι και η διαδικασία εισαγωγής δεδομένων στη δομή του *R-tree* είναι παρόμοια με αυτή του *B-tree* (Comer (1979)). Τα νέα στοιχεία

προστίθενται στα φύλλα του δένδρου, αφού πρώτα έχει ακολουθηθεί μια πορεία από πάνω προς τα κάτω ώστε να εντοπισθεί ο κατάλληλος κόμβος. Οι κόμβοι που υπερχειλίζουν διασπώνται και οι διασπάσεις μεταδίδονται προς τα ανώτερα επίπεδα του δένδρου.

Η διαδικασία εισαγωγής φαίνεται στην εικόνα που ακολουθεί (Παράρτημα Α). Το ορθογώνιο προς εισαγωγή είναι το σκιαγραφημένο ορθογώνιο *R21*. Αρχικά καλείται η *ChooseLeaf* που επιστρέφει το ορθογώνιο *R1* το οποίο περιλαμβάνει εξ' ολοκλήρου το νέο ορθογώνιο. Συνεχίζοντας την αναζήτηση, ο αλγόριθμος πρέπει να επιλέξει μεταξύ των ορθογωνίων *R3* και *R4* τα οποία τέμνονται με το ορθογώνιο που εισάγεται. Ο αλγόριθμος εντοπίζει το κατάλληλο φύλλο και επιστρέφει το *R3* επειδή απαιτεί τη μικρότερη αύξηση της επιφάνειάς του προκειμένου να συμπεριλάβει το νέο ορθογώνιο. Ωστόσο, η εισαγωγή του *R21* στο *R3* προκαλεί την υπερχείλιση του τελευταίου (αφού *M=3)*. Ως αποτέλεσμα, καλείται η διαδικασία διάσπασης κόμβων, που περιγράφεται στην επόμενη ενότητα και δημιουργεί τον νέο κόμβο *R3'*. Οι *R3* και *R3'* θα μοιραστούν κατάλληλα τα ορθογώνια *R8, R9, R10* και *R21*. Τέλος καλείται η *AdjustTree,* προκειμένου να προσαρμοστούν τα περιθώρια των δύο κόμβων και να διαδοθούν οι αλλαγές προς τα πάνω. Επειδή το ορθογώνιο *R1* έχει χώρο για να συμπεριλάβει ακόμη ένα ορθογώνιο (το νέο *R3'*) δεν εφαρμόζεται καμία αλλαγή στα ανώτερα επίπεδα του δένδρου.



Εικόνα 9: Εισαγωγή στο *R-tree*. Αριστερά: το *R21* είναι το *MBR* του αντικειμένου που θα εισαχθεί στη δομή. Δεξιά: η μορφή της δομής μετά την εισαγωγή.

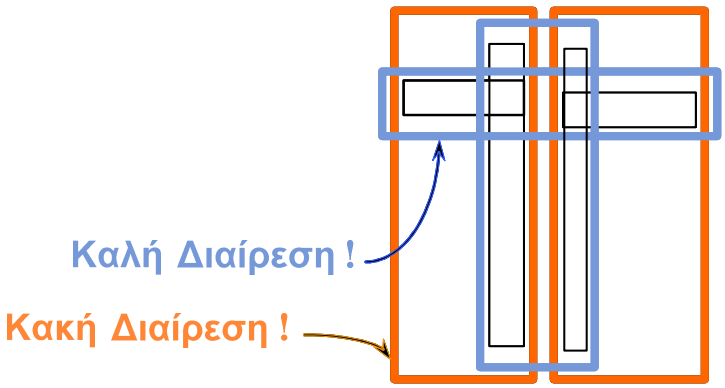
(http://students.fim.uni-passau.de/~kaebisch/Proseminar/Proseminar.pdf)

Στη συνέχεια παρουσιάζεται η μέθοδος διάσπασης κόμβων στην περίπτωση υπερχείλισης, όπως περιγράφηκε από τον Guttman (1984). Όπως φάνηκε, η διαδικασία

αποτελεί βασικό τμήμα της μεθόδου εισαγωγής και κρίνεται κατάλληλη η ανάλυσή της σε αυτό το σημείο.

1.1.2.2.3. Διάσπαση Κόμβων

Όπως περιγράφηκε, κατά την εισαγωγή μιας νέας εγγραφής σε ένα πλήρη κόμβο με *M* εγγραφές, είναι απαραίτητη η δημιουργία ενός δεύτερου κόμβου και η διαίρεση των *M+1* εγγραφών μεταξύ δύο. Ο διαμοιρασμός πρέπει να γίνει έτσι ώστε να ελαχιστοποιείται η πιθανότητα εξέτασης και των δύο νέων κόμβων, σε μια μεταγενέστερη αναζήτηση. Δεδομένου ότι η απόφαση για το αν θα εξεταστεί ένας κόμβος εξαρτάται από την επικάλυψη του *MBR* με την περιοχή αναζήτησης, η συνολική επιφάνεια των *MBRs* των δύο νέων κόμβων θα πρέπει να ελαχιστοποιηθεί. Η Εικόνα 10 παρουσιάζει δύο διαφορετικές επιλογές για τη διάσπαση κόμβων. Ο διαχωρισμός που σχεδιάζεται με μπλε χρωματισμό αποτελεί την καλύτερη περίπτωση, μιας και η επιφάνεια των δύο *MBRs* είναι η μικρότερη δυνατή.

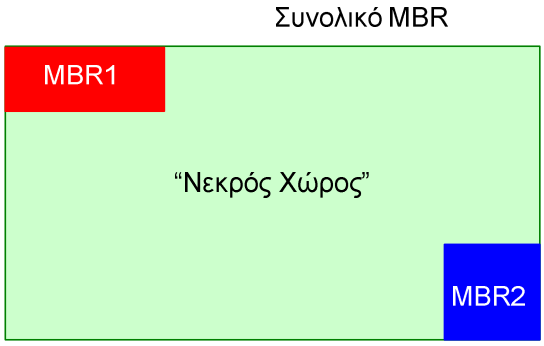


Εικόνα 10: Μια καλή και μια κακή επιλογή για διάσπαση κόμβων.

Ο Guttman (1984) πρότεινε στην εργασία του τρεις μεθόδους διάσπασης κόμβων, την εξαντλητική *(Exhaustive)*, την τετραγωνική *(Quadratic)* και τη γραμμική *(Linear)*. Όλες έχουν σχεδιαστεί ώστε να ελαχιστοποιούν την επιφάνεια που καλύπτεται από τα δύο ορθογώνια που προκύπτουν μετά τη διάσπαση. Η πρώτη μέθοδος εντοπίζει την επιφάνεια με το συνολικό ελάχιστο. Ωστόσο, παρουσιάζει εκθετική πολυπλοκότητα ως προς το πλήθος των εγγραφών *M* σε κάθε κόμβο ή φύλλο του δένδρου κι έτσι ο χρόνος εκτέλεσης και η χρήση της *CPU* την καθιστά αναποτελεσματική. Οι δύο επόμενες μέθοδοι προσπαθούν να προσεγγίσουν τη βέλτιστη λύση. Πιο συγκεκριμένα, η τετραγωνική μέθοδος παρουσιάζει τετραγωνική πολυπλοκότητα σε σχέση με την τιμή του *M* οδηγώντας σε αυξημένο χρόνο εκτέλεσης, εντούτοις επιτυγχάνει μια καλή

προσέγγιση. Η γραμμική μέθοδος είναι η ταχύτερη, αφού η πολυπλοκότητα σε σχέση με την τιμή του *M* είναι γραμμική, αλλά οδηγείται σε χειρότερη προσέγγιση. Συνολικά, η τετραγωνική μέθοδος διάσπασης είναι η πιο κατάλληλη από τις τρεις, αφού συμβιβάζει την ταχύτητα με την αποδοτικότητα κατά την αναζήτηση και στη συνέχεια γίνεται περιγραφή αυτής της μεθόδου.

Η μέθοδος για τη δημιουργία των ομάδων περιλαμβάνει δύο στάδια (Παράρτημα Α). Το πρώτο εκτελείται μια φορά και κάνει την επιλογή των εγγραφών που θα αποτελέσουν τα πρώτα στοιχεία των δύο ομάδων. Το κριτήριο που χρησιμοποιείται για τον εντοπισμό των καταλληλότερων εγγραφών είναι η απόσταση μεταξύ των ορθογωνίων τους. Συγκεκριμένα αναζητείται το ζεύγος εγγραφών με ορθογώνια που απέχουν όσο το δυνατόν περισσότερο και συνεπώς θα σπαταλούσαν τον μεγαλύτερο χώρο αν τοποθετούνταν στην ίδια ομάδα. Η σπατάλη μετριέται από τον "νεκρό χώρο" ο οποίος συνίσταται από τη διαφορά της επιφάνειας του ορθογωνίου που περικλείει τα ορθογώνια των δύο εγγραφών, από τις επιφάνειές των ορθογωνίων καθεμιάς από αυτές (Εικόνα 11).



Εικόνα 11: Ο "νεκρός χώρος" σε ένα *MBR*.

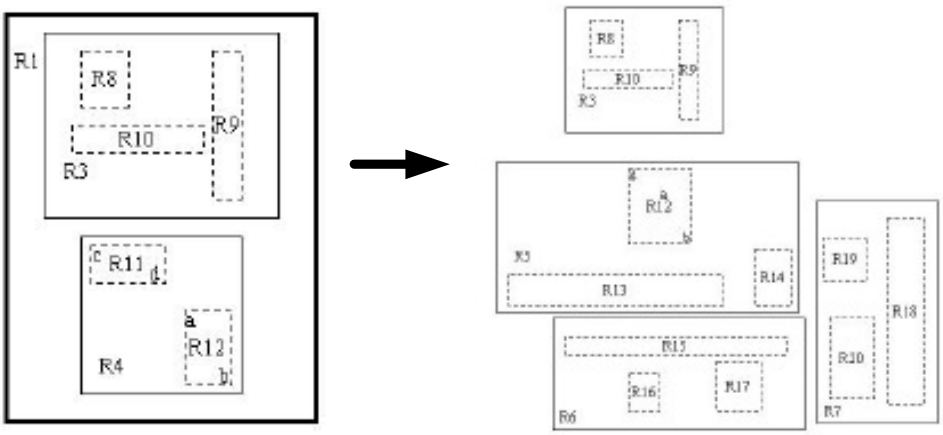
Για κάθε ζεύγος των *M+1* εγγραφών, υπολογίζεται η τιμή του "νεκρού χώρου" που προκύπτει από την ομαδοποίησή τους και τελικά επιλέγεται το ζεύγος με τη μεγαλύτερη από αυτές. Οι εγγραφές του τελευταίου θα διαχωριστούν και θα αποτελέσουν τα πρώτα στοιχεία των ομάδων και άρα των δύο νέων κόμβων. Με αυτό τον τρόπο ελαχιστοποιείται ο "νεκρός χώρος" και συνακόλουθα η πιθανότητα να ακολουθήσουμε περισσότερα του ενός μονοπάτια προς τα φύλλα, κατά τη διάρκεια μιας αναζήτησης.

Στο δεύτερο στάδιο εκτελείται μια επαναληπτική διαδικασία για την αντιστοίχιση των υπολοίπων *M-1* εγγραφών στις δύο ομάδες. Για κάθε εγγραφή υπολογίζεται η αύξηση που απαιτείται στην επιφάνεια του *MBR* κάθε ομάδας, προκειμένου να συμπεριλάβει

το *MBR* της εγγραφής. Μετά την ολοκλήρωση των υπολογισμών για όλες τις εγγραφές, επιλέγεται εκείνη που εμφανίζει τη μεγαλύτερη διαφορά μεταξύ των δύο ομάδων και αντιστοιχίζεται στην ομάδα με τη μικρότερη αύξηση. Κατόπιν η διαδικασία εκτελείται επαναληπτικά για το νέο υπόλοιπο των εγγραφών.

1.1.2.2.4. Διαγραφή Δεδομένων

Η περίπτωση της διαγραφής είναι και αυτή παρόμοια με τον τρόπο διαγραφής στο *B-tree* (Comer (1979)) και απεικονίζεται στην εικόνα που ακολουθεί. Για το δένδρο θεωρείται *m=2*, *M=4* και ότι το αντικείμενο προς διαγραφή είναι το *c*. Αρχικά καλείται η *FindLeaf* (Παράρτημα Α) ώστε να βρεθεί το φύλλο που περιλαμβάνει την εγγραφή και ως αποτέλεσμα επιστρέφεται το *R11*. Μετά τη διαγραφή καλείται η *CondenseTree* (Παράρτημα Α) ώστε να προσαρμοστούν τα *MBRs* σε καθένα από τα επίπεδα του δένδρου. Η διαγραφή του αντικειμένου *c* οδηγεί σε διαγραφή του *R11* μιας και πλέον περιλαμβάνει μικρό αριθμό εγγραφών. Ως αποτέλεσμα διαγράφεται το *R4* που πλέον περιλαμβάνει μικρό αριθμό *MBRs*, γεγονός που οδηγεί και στη διαγραφή του *R1* στο αμέσως ανώτερο επίπεδο, για τον ίδιο λόγο. Τα *R1* και *R2* (Εικόνα 8) αποτελούν το επίπεδο της ρίζας του κέντρου. Έτσι λοιπόν μετά την διαγραφή το δένδρο που προκύπτει έχει στο επίπεδο της ρίζας τα *R3, R5, R6* και *R7*, όπως φαίνεται και στην παρακάτω εικόνα.



Εικόνα 12: Διαγραφή στο *R-tree*.

(http://students.fim.uni-passau.de/~kaebisch/Proseminar/Proseminar.pdf)

Από τη διαδικασία που περιγράφηκε παραπάνω, φαίνεται ότι ο τρόπος διαχείρισης των κόμβων με πλήθος εγγραφών μικρότερο από το ελάχιστο επιτρεπτό, διαφέρει από την περίπτωση του *B-tree* (Comer (1979)). Στο τελευταίο, δύο ή περισσότεροι γειτονικοί

κόμβοι με τέτοια χαρακτηριστικά απλώς συγχωνεύονται. Αντιθέτως, στην περίπτωση του *R-tree*, οι κόμβοι αυτοί διαγράφονται και οι εγγραφές τους επαν-εισάγονται στη δομή. Τα πλεονεκτήματα της μεθόδου είναι τα ακόλουθα:

- επιτυγχάνει το ίδιο αποτέλεσμα με τη συγχώνευση, αλλά είναι ευκολότερη στην υλοποίησή της αφού χρησιμοποιεί την ήδη έτοιμη μέθοδο εισαγωγής *Insert.*
- λόγω του γεγονότος ότι ο αριθμός των προσπελάσεων των σελίδων στη μνήμη αποτελεί ένα κρίσιμο μέγεθος της αποδοτικότητας, η τελευταία είναι συγκρίσιμη μεταξύ των δύο μεθόδων μιας και οι σελίδες που θα χρησιμοποιηθούν για την επαν-εισαγωγή θα βρίσκονται ήδη στη μνήμη από τη διαδικασία αναζήτησης (στις περισσότερες των περιπτώσεων).
- βελτιώνει σταδιακά τη χωρική δομή του δένδρου και αποτρέπει την επιδείνωσή της, που μπορεί να προκύψει εάν κάθε εγγραφή τοποθετείται συνεχώς κάτω από τον ίδιο κόμβο-γονέα.

1.1.2.2.5. Ενημέρωση Δεδομένων

Στην περίπτωση που μια εγγραφή της ΒΔ μεταβληθεί έτσι ώστε να τροποποιηθεί το *MBR* της, πρέπει να διαγραφεί από τη δομή, να ενημερωθεί με τα νέα δεδομένα και κατόπιν να επαν-εισαχθεί, αναζητώντας τη νέα κατάλληλη θέση στο δένδρο.

1.1.3. Το R* tree

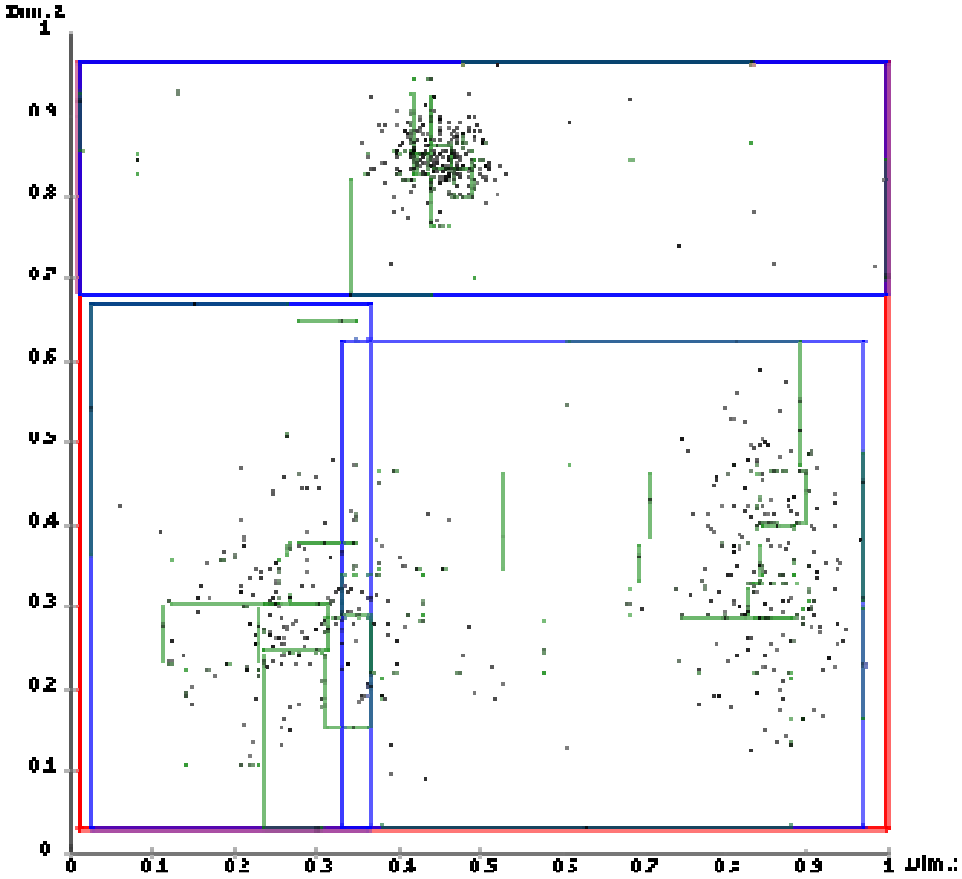
Το *R-tree* (Guttman (1984)) υποστηρίζει με αποδοτικό τρόπο, τη δυναμική κατασκευή *MBRs* από υποσύνολα πλήθους μεταξύ *m* και *M* για ένα τυχαίο σύνολο ορθογωνίων, έτσι ώστε να εκτελούνται τυχαίες διαδικασίες ανάκτησης με ορθογώνια ερωτήματος τυχαίου μεγέθους. Η λειτουργία της δομής στηρίζεται στην ευρηστική βελτιστοποίηση της περιοχής του *MBR* σε κάθε εσωτερικό κόμβο του δένδρου. Το κριτήριο αυτό είναι το μοναδικό που λαμβάνεται υπόψη, δίχως ωστόσο να είναι το καλύτερο δυνατό. Μετά την ανάλυση του τρόπου κατασκευής του R tree εγείρονται ερωτήματα όπως:

- Γιατί να μην ελαχιστοποιείται το περιθώριο των *MBRs*;
- Γιατί να μην ελαχιστοποιείται η επικάλυψη μεταξύ των *MBRs*;
- Γιατί να μην βελτιστοποιείται η διαδικασία αποθήκευσης;
- Γιατί να μη βελτιστοποιούνται όλα αυτά τα κριτήρια ταυτόχρονα;
- Μπορεί αυτά τα κριτήρια να αλληλεπιδρούν αρνητικά;

Οι παραπάνω παράμετροι, που αφορούν την απόδοση της διαδικασίας ανάκτησης δεδομένων, αλληλεπιδρούν μεταξύ τους με έναν πολύπλοκο τρόπο με αποτέλεσμα να είναι αδύνατη η βελτιστοποίηση μιας εξ' αυτών χωρίς να επηρεάζονται οι υπόλοιπες. Επιπρόσθετα, αφού τα *MBRs* των εγγραφών της ΒΔ μπορούν να έχουν πολύ διαφορετικά μεγέθη και σχήματα και τα *MBRs* των εσωτερικών κόμβων της δομής μεγεθύνονται και σμικρύνονται δυναμικά, η επιτυχία μεθόδων που βελτιστοποιούν μία παράμετρο είναι πολύ αβέβαιη.

Εκτελώντας πολυάριθμα και ποικίλα πειράματα, ο Beckmann et. al. (1990) σχεδίασε το *R*-tree* που ενσωματώνει έναν συνδυασμό βελτιστοποίησης της περιοχής, του περιθωρίου και τις επικάλυψης κάθε *MBR* στους κόμβους και τα φύλλα του δένδρου. Μετά από μια εξαντλητική σύγκριση επιδόσεων, απέδειξε ότι το *R*-tree* υπερτερεί σε σχέση με τις υπάρχουσες παραλλαγές του *R-tree* (*Linear R-tree, Quadratic R-tree, Greene's R-tree*). Η υπεροχή αυτή ισχύει για διαφόρους τύπους ερωτημάτων και πράξεων. Επιπρόσθετα των συνηθισμένων *point query, rectangle intersection* και *rectangle enclosure query*, οι συγγραφείς ανέλυσαν τη λειτουργία της νέας δομής στο *spatial join* ορθογωνίων ή πολυδιάστατων σημείων που αποτελεί την πλέον σημαντική πράξη σε συστήματα Γεωγραφικών και Περιβαλλοντικών ΒΔ.

Η Εικόνα 13 αναπαριστά τα *MBRs* ενός *R*-tree*. Η σύγκριση με την Εικόνα 7 κάνει εμφανή τη διαφορά της από το *R-tree*, αφού η επικάλυψη μεταξύ των ορθογωνίων στο *R*-tree* είναι πολύ μικρότερη, γεγονός που οδηγεί σε καλύτερη απόδοση κατά την εκτέλεση ενός ερωτήματος. Με πράσινο χρώμα αναπαριστώνται τα *MBRs* των αντικειμένων της ΒΔ (επίπεδο φύλλων) και με γαλάζιο και κόκκινο χρώμα σχεδιάζονται τα *MBRs* του ευρετηρίου, δηλ. των ανωτέρων επιπέδων του δένδρου (επίπεδο εσωτερικών κόμβων και επίπεδο ρίζας αντίστοιχα). Το σχήμα αυτό δημιουργήθηκε με τη βοήθεια του ELKI (http://elki.dbs.ifi.lmu.de/).



Εικόνα 13: Τα *MBRs*, για το επίπεδο των φύλλων (πράσινο χρώμα), το επίπεδο εσωτερικών κόμβων (γαλάζιο χρώμα) και το επίπεδο της ρίζας (κόκκινο χρώμα).

Συμπερασματικά, από πρακτικής άποψης το *R*-tree* (Beckmann et. al. (1990)) αποτελεί μια ελκυστική μεθοδολογία, αφού υποστηρίζει με αποδοτικό τρόπο σημειακά και χωρικά δεδομένα ταυτόχρονα και η υλοποίησή του κοστίζει ελαφρώς περισσότερο σε σχέση με το *R-tree* (Guttman (1984)) και τις υπόλοιπες παραλλαγές του (Greene (1989)).

1.1.3.1. Παράμετροι και Κριτήρια Αποδοτικότητας

Στη συνέχεια γίνεται θεώρηση κάποιων παραμέτρων που είναι απαραίτητες για την απόδοση της ανάκτησης δεδομένων και αναλύονται οι αλληλεξαρτήσεις τους.

(Π1) Η επιφάνεια που καλύπτεται από το MBR σε κάποιον εσωτερικό κόμβο του δένδρου πρέπει να ελαχιστοποιείται. Πιο συγκεκριμένα ο "νεκρός χώρος" στο MBR του κόμβου πρέπει να ελαχιστοποιείται, γεγονός που βελτιώνει σημαντικά την απόδοση της αναζήτησης αφού οι αποφάσεις για τη διαδρομή που πρέπει να ακολουθηθεί, θα λαμβάνονται από τα ανώτερα επίπεδα.

(Π2) Η επικάλυψη μεταξύ των MBRs στους εσωτερικούς κόμβους του δένδρου πρέπει να ελαχιστοποιείται. Ο παράγοντας αυτός οδηγεί σε περαιτέρω μείωση του αριθμού των διαδρομών που πρέπει να διασχιστούν κατά την αναζήτηση.

(Π3) Το περιθώριο του MBR σε κάποιον εσωτερικό κόμβο του δένδρου πρέπει να ελαχιστοποιείται. Εδώ το περιθώριο αποτελείται από το άθροισμα των μηκών των πλευρών ενός ορθογωνίου. Υποθέτοντας σταθερή επιφάνεια, το αντικείμενο με το μικρότερο περιθώριο είναι το τετράγωνο. Έτσι, ελαχιστοποιώντας το περιθώριο αντί της επιφάνειας, τα ορθογώνια των εσωτερικών κόμβων του δένδρου θα σχηματίζονται πιο τετραγωνισμένα βελτιώνοντας την απόδοση σε περιπτώσεις με μεγάλα τετραγωνισμένα ορθογώνια ερωτήματος. Επιπρόσθετα, η ελαχιστοποίηση του περιθωρίου θα βελτιώσει σημαντικά τη δομή. Αφού τα τετραγωνισμένα αντικείμενα μπορούν να συσκευαστούν-πακεταριστούν ευκολότερα, τα ορθογώνια ενός επιπέδου θα δημιουργούν ακόμη μικρότερα περιβάλλοντα ορθογώνια στο αμέσως ανώτερο επίπεδο. Συμπερασματικά, η ομαδοποίηση ορθογωνίων σε περιβάλλοντα πλαίσια με μικρή διακύμανση στα μήκη των άκρων τους, οδηγεί σε μείωση της επιφάνειας των MBRs στους εσωτερικούς κόμβους της δομής..

(Π4) Η χρήση της διαθέσιμης μνήμης πρέπει να βελτιστοποιηθεί. Υψηλότερη χρήση θα μειώσει εν γένει το κόστος ερωτήματος καθώς το ύψος του δέντρου θα παραμείνει χαμηλό. Προφανώς, οι τύποι πράξεων με μεγάλα ορθογώνια ερωτήματος επηρεάζονται σημαντικότερα, δεδομένου ότι η συγκέντρωση ορθογωνίων σε περισσότερους κόμβους θα έχει μεγαλύτερη επίδραση εάν το πλήθος των κλειδιών που πρέπει να αναζητηθούν είναι μεγάλο.

Όσον αφορά τις σχέσεις και τις αλληλοεξαρτήσεις μεταξύ των παραπάνω παραμέτρων, σημειώνονται τα εξής:

(Σ1) Η διατήρηση της επιφάνειας (Π1) και της επικάλυψης (Π2) των MBRs στους εσωτερικούς κόμβους του δένδρου σε χαμηλά επίπεδα, απαιτεί περισσότερη ελευθερία στην επιλογή του πλήθους των ορθογωνίων που θα αποθηκεύονται σε αυτούς, γεγονός που αντισταθμίζεται με χειρότερη χρήση του αποθηκευτικού χώρου (Π4).

(Σ2) Η εφαρμογή των κριτηρίων (Π1) και (Π2) απαιτεί περισσότερη ελευθερία στην επιλογή του σχήματος των ορθογωνίων, με αποτέλεσμα τα τελευταία να είναι τετραγωνισμένα σε μικρότερο βαθμό (Π3).

(Σ3) Η ελαχιστοποίηση της επιφάνειας (Π1) επηρεάζει θετικά την επικάλυψη (Π2) μεταξύ των MBRs στους εσωτερικούς κόμβους, αφού μειώνεται η κάλυψη του χώρου των δεδομένων.

(Σ4) Η ελαχιστοποίηση των περιθωρίων (Π3) οδηγεί σε πιο τετραγωνισμένα MBRs και επιτυγχάνει καλύτερη χρήση της μνήμης, βελτιώνοντας την ομαδοποίησή τους (Π4).

(Σ5) Η απόδοση για τις περιπτώσεις με αρκετά μεγάλα ορθογώνια ερωτήματος θα επηρεαστεί περισσότερο από την παράμετρο της χρήσης του αποθηκευτικού χώρου (Π4) σε σχέση με τις παραμέτρους (Π1)-(Π3).

Από τα παραπάνω, γίνεται εύκολα αντιληπτή η δυσκολία να βρεθεί η κατάλληλη ισορροπία μεταξύ των παραμέτρων.

1.1.3.2. Παραλλαγές στο R*-tree

1.1.3.2.1. Αλγόριθμος ChooseSubtree

Για την επιλογή της κατάλληλης διαδρομής κατά την εισαγωγή μιας νέας εγγραφής, το *R-tree* λαμβάνει υπόψη μόνο την παράμετρο της επιφάνειας. Ο Beckmann et. al. (1990) εξέτασε τις παραμέτρους της επιφάνειας, του περιθωρίου και τις επικάλυψης σε διάφορους συνδυασμούς. Η επικάλυψη μιας εγγραφής καθορίζεται ως ακολούθως: έστω $E_1,...,E_p$ οι εγγραφές στον τρέχοντα κόμβο, τότε η επικάλυψη προκύπτει από:

$$overlap\left(E_k = \sum_{i=1, i \neq k}^{p} area\left(E_k.\mathrm{Re}\,c\tan gle \cap E_i.\mathrm{Re}\,c\tan gle\right)\right),\ 1 \leq k \leq p$$

Μετά από πειραματικές μετρήσεις, οι συγγραφείς κατέληξαν στην έκδοση με την καλύτερη απόδοση ανάκτησης (Παράρτημα Β). Ο αλγόριθμος παρουσιάζει την ίδια απόδοση με τον αρχικό αλγόριθμο του Guttman (1984) (*ChooseLeaf,* Παράρτημα Α) κατά τη διαδικασία επιλογής του κατάλληλου κόμβου του δένδρου. Ωστόσο, για την περίπτωση των φύλλων η μέθοδός τους, με ελαχιστοποίηση της επικάλυψης παρουσίασε ελαφρώς καλύτερη απόδοση. Το υπολογιστικό κόστος για τον προσδιορισμό της επικάλυψης είναι τετραγωνικό σε σχέση με τον αριθμό των εγγραφών, αφού για κάθε εγγραφή πρέπει να υπολογιστεί η επικάλυψη με όλες τις υπόλοιπες εγγραφές του κόμβου. Ωστόσο, για μεγάλα μεγέθη κόμβων ο αριθμός των εγγραφών για τις οποίες απαιτείται ο παραπάνω υπολογισμός μπορεί να μειωθεί, αφού για πολύ απομακρυσμένα ορθογώνια η πιθανότητα να επιτευχθεί η ελάχιστη επικάλυψη είναι πολύ μικρή. Έτσι, προκειμένου να μειωθεί το κόστος της *CPU*, μπορεί να εξεταστεί ένα πλήθος *p* ορθογωνίων (Παράρτημα Β).

Για δύο διαστάσεις, οι συγγραφείς κατέληξαν ότι για *p*=32 δεν υπάρχει σχεδόν καθόλου μείωση της απόδοσης ανάκτησης. Το κόστος της *CPU* παραμένει υψηλότερο σε σχέση με την αρχική έκδοση του *ChooseLeaf*, αλλά ο αριθμός των *disk I/Os* μειώνεται. Οι μετρήσεις έδειξαν ότι η βελτιστοποίηση του *ChooseSubtree* αυξάνει την απόδοση ανάκτησης και συγκεκριμένα σε περιπτώσεις *με μικρά ορθογώνια ερωτήματος σε αρχεία δεδομένων με μη-ομοιόμορφα κατανεμημένα μικρά ορθογώνια ή σημεία.* Στις υπόλοιπες περιπτώσεις, η απόδοση του αλγορίθμου του Guttman (1984) είναι παρόμοια με αυτή. Συνεπώς, μπορεί σε πρώτη φάση να καθοριστεί μια βελτίωση της ανθεκτικότητας.

1.1.3.2.2. Διάσπαση του *R*-tree*

Και στο *R*-tree* η υπερχείλιση ενός κόμβου ή φύλλου του δένδρου οδηγεί στη διάσπασή του. Ωστόσο, η μεθοδολογία που χρησιμοποιείται είναι διαφορετική από το *R-tree*. Κατά μήκος κάθε άξονα του χώρου δεδομένων, οι εγγραφές αρχικά ταξινομούνται με βάση την κατώτερη και κατόπιν την ανώτερη τιμή των ορθογωνίων τους. Για κάθε ταξινόμηση, καθορίζονται οι $M - 2 \cdot m + 2$ κατανομές των *M+1* εγγραφών στις δύο ομάδες για τις οποίες ισχύει:

k-οστή κατανομή: *η πρώτη ομάδα περιλαμβάνει τις πρώτες (m-1)+k εγγραφές και η δεύτερη ομάδα τις υπόλοιπες, όπου* (k = 1, …, ($M - 2 \cdot m + 2$)).

Για κάθε κατανομή προσδιορίζονται τιμές καταλληλότητας. Με βάση τις τιμές αυτές προσδιορίζεται η τελική βέλτιστη κατανομή όλων των εγγραφών. Ο Beckmann et. al. (1990) πραγματοποίησε πειραματικές μετρήσεις για τρεις διαφορετικές τιμές καταλληλότητας και διαφορετικές προσεγγίσεις, σε διάφορους συνδυασμούς.

- Επιφάνεια [ππ(πρώτης ομάδας)] + Επιφάνεια [ππ(δεύτερης ομάδας)]
- Περιθώριο [ππ(πρώτης ομάδας)] + Περιθώριο [ππ(δεύτερης ομάδας)]
- Επιφάνεια [ππ(πρώτης ομάδας) ∩ ππ(δεύτερης ομάδας)]

Εδώ το ππ δηλώνει το Περιβάλλον Πλαίσιο ενός συνόλου ορθογωνίων.

Με βάση τις παραπάνω τιμές καταλληλότητας οι πιθανές μέθοδοι επεξεργασίας για την βέλτιστη επιλογή είναι ο καθορισμός:

- του ελάχιστου κατά μήκος ενός άξονα ή μιας ταξινόμησης
- του ελάχιστου αθροίσματος των τιμών καταλληλότητας κατά μήκος ενός άξονα ή μιας ταξινόμησης
- του συνολικού ελάχιστου

Η τιμές αυτές καθορίζουν τον άξονα διάσπασης ή την τελική κατανομή (σε κάποιον επιλεγμένο άξονα διάσπασης). Οι συγγραφείς κατέληξαν στον αλγόριθμο που επιτυγχάνει την καλύτερη συνολική επίδοση (Παράρτημα Β). Ο αλγόριθμος διάσπασης δοκιμάστηκε για m = 20%, 30%, 40% και 45% της τιμής του μέγιστου αριθμού εγγραφών M. Όπως προέκυψε από τα πειράματα για διάφορες τιμές του M, η καλύτερη απόδοση επετεύχθη για τιμή m = 40%.

Τέλος, το κόστος του αλγορίθμου διάσπασης αναλύεται σε δύο τμήματα:

- για κάθε άξονα (διάσταση) οι εγγραφές πρέπει να ταξινομηθούν δύο φορές, το οποίο απαιτεί χρόνο $O(M \cdot \log(M))$ και αποτελεί σχεδόν το μισό χρόνο της διαδικασίας διάσπασης.
- για κάθε άξονα, πρέπει να υπολογιστούν, το περιθώριο των $2 \cdot (2 \cdot (M - 2 \cdot m + 2))$ ορθογωνίων και η επικάλυψη των $2 \cdot (M - 2 \cdot m + 2)$ κατανομών το οποίο αποτελεί το υπόλοιπο τμήμα του χρόνου.

1.1.3.2.3. Εξαναγκασμένη Επανατοποθέτηση

Το *R-tree* (Guttman (1984)) και το *R*-tree* (Beckmann et. al. (1990)) είναι μη ντετερμινιστικές δομές, όσον αφορά τον τρόπο που κατανέμουν τις εγγραφές στους κόμβους. Διαφορετικές ακολουθίες εισαγωγής δεδομένων οδηγούν στην κατασκευή διαφορετικών δένδρων. Για το λόγο αυτό το *R-tree* "χωλαίνει" από τις πρώτες εγγραφές. Τα ορθογώνια δεδομένων που εισάγονται στην αρχή της κατασκευής της δομής μπορεί να δημιουργήσουν *MBRs* στους εσωτερικούς κόμβους, τα οποία δεν εξασφαλίζουν καλή απόδοση ανάκτησης.

Ένα άλλο πρόβλημα μπορεί να προκύψει κατά τη διαγραφή εγγραφών από τη δομή. Στην περίπτωση που δημιουργηθούν κόμβοι με αριθμό εγγραφών μικρότερο του *m,* πρέπει να συγχωνευθούν με τους κόμβους του ανώτερου επιπέδου. Η αντιμετώπιση στο *R-tree* είναι η διαγραφή του κόμβου και η επαν-εισαγωγή των ορφανών εγγραφών σε διαφορετικούς κόμβους. Αντίστοιχα στην περίπτωση διάσπασης ενός κόμβου λόγω υπερχείλισης, το *R-tree* δημιουργεί δύο νέους κόμβους στους οποίους κατανέμει τις *M+1* εγγραφές.

Από τα παραπάνω γίνεται εμφανές ότι το *R-tree* σε αυτές τις περιπτώσεις κάνει μια πολύ τοπική αναδιοργάνωση των ορθογωνίων, η οποία ωστόσο δεν είναι αρκετή. Συνεπώς, είναι επιθυμητή η ανάπτυξη μιας ισχυρότερης και λιγότερο τοπικής λειτουργίας για την αναδιοργάνωση της δομής.

Το *R*-tree* προκειμένου να αντιμετωπίσει αποτελεσματικότερα αυτές τις περιπτώσεις και να δημιουργήσει μια πιο "συμπαγή" δομή, επιβάλει τη δυναμική αναδιοργάνωση της δομής αναγκάζοντας κάποιες εγγραφές να επανατοποθετηθούν κατά τη ρουτίνα εισαγωγής. Ο αλγόριθμος (Παράρτημα Β) βασίζεται στην ικανότητα της ρουτίνας εισαγωγής να τοποθετεί εγγραφές σε κάθε επίπεδο του δέντρου, όπως ακριβώς απαιτείται και από τον αλγόριθμο διαγραφής του Guttman (1984). Η διαφοροποίηση έγκειται στον τρόπο αντιμετώπισης της υπερχείλισης ενός κόμβου, όπου καλείται η υπο-ρουτίνα *OverflowTreatment* και εκτελεί την επαν-εισαγωγή των δεδομένων σε γειτονικούς κόμβους, ή προκαλεί τη διάσπαση του κόμβου.

Εάν εισαχθεί ένα νέο ορθογώνιο δεδομένων, κάθε πρώτη αντιμετώπιση υπερχείλισης κόμβου σε κάθε επίπεδο της δομής θα έχει ως αποτέλεσμα την επανατοποθέτηση *p* εγγραφών, το οποίο μπορεί να προκαλέσει διάσπαση του κόμβου που προκάλεσε την υπερχείλιση εάν όλες οι εγγραφές επαν-εισαχθούν στο ίδιο σημείο. Διαφορετικά, οι

διασπάσεις μπορούν να προκύψουν σε έναν ή περισσότερους κόμβους, αλλά σε πολλές περιπτώσεις οι διασπάσεις εμποδίζονται εντελώς.

Η παράμετρος *p* μπορεί να μεταβάλλεται ανεξάρτητα για τους κόμβους φύλλα και τους εσωτερικούς κόμβους, ως τμήμα της βελτίωσης της απόδοσης. Για το λόγο αυτό οι συγγραφείς εξέτασαν πειραματικά διάφορες τιμές της παραμέτρου και κατέληξαν ότι η καλύτερη απόδοση επιτυγχάνεται για *p*=30% του *M*, τόσο για τους κόμβους φύλλα όσο για τους υπόλοιπους κόμβους.

Επιπλέον, για όλα τα αρχεία δεδομένων και τα αρχεία ερωτημάτων, η κοντινή επανατοποθέτηση υπερτερεί της μακρινής. Η κοντινή επανατοποθέτηση επιλέγει τον κόμβο που περιελάμβανε τις εγγραφές προηγούμενα αφού το *MBR* του είχε μειωθεί σε μέγεθος. Συνεπώς, αυτός ο κόμβος έχει μικρότερη πιθανότητα να επιλεγεί από την *ChooseSubtree* ξανά.

Συνοψίζοντας, μπορούμε να πούμε ότι η εξαναγκασμένη επανατοποθέτηση:

- μεταφέρει εγγραφές μεταξύ γειτονικών κόμβων κι έτσι μειώνει την επικάλυψη,
- βελτιώνει τη χρήση αποθήκευσης,
- οδηγεί σε λιγότερες διασπάσεις λόγω μεγαλύτερης ανακατασκευής της δομής,
- επιτυγχάνει πιο τετραγωνικό σχήμα των *MBRs* στους εσωτερικούς κόμβους, γεγονός που όπως αναφέρθηκε αποτελεί επιθυμητό χαρακτηριστικό.

Τέλος, το κόστος της *CPU* θα είναι υψηλότερο αφού η ρουτίνα εισαγωγής καλείται πιο συχνά. Ωστόσο, το "μειονέκτημα" αυτό μετριάζεται από το μειωμένο πλήθος διασπάσεων. Ο Beckmann et. al. (1990) έδειξε ότι με τη χρήση της εξαναγκασμένης επανατοποθέτησης στη δομή του *R*-tree*, ο μέσος όρος των προσπελάσεων δίσκου για τις εισαγωγές αυξάνεται μονάχα περίπου 4% (και παραμένει ο χαμηλότερος σε σχέση με όλες τις παραλλαγές του *R-tree* (Guttman (1984); Greene (1989)). Αυτό οφείλεται κυρίως στη δομή μετά τη βελτίωση των ιδιοτήτων του αλγορίθμου εισαγωγής.

1.2. Ερωτήματα Κοντινότερου Γείτονα

Τα ερωτήματα Κοντινότερου Γείτονα *(Nearest Neighbor, NN)* αποτελούν ένα χρήσιμο εργαλείο στα Γεωγραφικά Πληροφοριακά Συστήματα και η αποτελεσματική υλοποίησή τους αποτέλεσε ένα ευρή πεδίο έρευνας. Ο χρήστης μπορεί να εξάγει πληροφορία της μορφής:

- Να βρεθούν τα 10 κοντινότερα αντικείμενα της ΒΔ σε ένα επιλεγμένο σημείο του χώρου.
- Να βρεθούν τα 5 κοντινότερα αντικείμενα της ΒΔ σε ένα επιλεγμένο σημείο του χώρου, τα οποία βρίσκονται σε απόσταση μεταξύ 100 και 140 μονάδες μήκους.

Επίσης, τα ερωτήματα *NN* μπορεί να συνδυαστούν και με πιο σύνθετες πληροφορίες και λειτουργίες (π.χ. χωρικά *join*) ώστε να απαντηθούν συνθετότερα ερωτήματα, όπως:

- Να βρεθούν τα 7 κοντινότερα αντικείμενα της ΒΔ που βρίσκονται δυτικά από ένα επιλεγμένο αντικείμενο-σημείο του χώρου.
- Να βρεθούν τα 4 κοντινότερα αντικείμενα της ΒΔ σε δύο επιλεγμένα σημεία του χώρου.
- Να βρεθούν τα 3 μακρινότερα αντικείμενα της ΒΔ σε ένα επιλεγμένο σημείο του χώρου.

Τέλος, η επέκταση σε χώρους με υψηλότερες διαστάσεις καθιστά τα ερωτήματα *NN* ένα χρήσιμο και αποτελεσματικό εργαλείο για Εξόρυξη Δεδομένων με πιο πολύπλοκα κριτήρια. Για παράδειγμα, εάν υποθέσουμε μια ΒΔ με χρονοσειρές που αναπαριστούν τιμές μετοχών, ή τις πωλήσεις μιας εταιρίας κατά τη διάρκεια ενός έτους, τότε μπορούμε να σκεφτούμε τις εγγραφές αυτές ως σημεία στον πολυδιάστατο χώρο. Πιο συγκεκριμένα, ένα χρονικό παράθυρο των τιμών μιας μετοχής στη διάρκεια μιας εβδομάδας, μπορεί να αναπαρασταθεί ως ένα σημείο στον 7-διάστατο χώρο. Θέτοντας λοιπόν ως αντικείμενο του ερωτήματος ένα σημείο στον πολυδιάστατο χώρο, μας δίνεται η δυνατότητα να εντοπίζουμε αντικείμενα της ΒΔ, που βρίσκονται κοντά σε αυτό, ή με άλλα λόγια παρουσιάζουν υψηλό βαθμό "ομοιότητας". Έτσι λοιπόν μας δίνεται η δυνατότητα να εξάγουμε πληροφορία με ερωτήματα της μορφής:

- Να βρεθούν 5 μετοχές που παρουσίασαν την ίδια διακύμανση με μια συγκεκριμένη μετοχή στο διάστημα μιας εβδομάδας.
- Να βρεθεί ο μήνας κατά τον οποίο οι πωλήσεις της εταιρίας παρουσίασαν την ίδια καμπύλη πωλήσεων με το μήνα Μάρτιο.

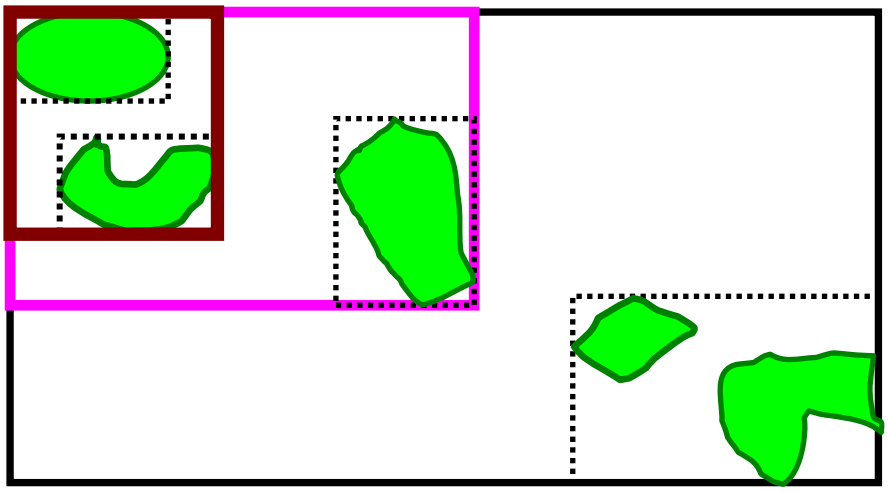
Η αποτελεσματική επεξεργασία των *NN* ερωτημάτων απαιτεί δομές χωρικών δεδομένων που αυξάνουν την εγγύτητα των αντικειμένων, ώστε να εστιάσουν στην αναζήτηση μονάχα των πιθανών γειτόνων. Στην σχετική βιβλιογραφία μπορεί να συναντήσει κανείς μια μεγάλη γκάμα από *SAMs* (Samet (1989)). Ωστόσο, πολύ λίγες έχουν χρησιμοποιηθεί για ερωτήματα *NN*. Ο Samet (1990)), παρουσίασε ευρηστικές συναρτήσεις για την εύρεση αντικειμένων στο *quadtree* (Finkel και Bentley (1974)). Το ίδιο *kNN* πρόβλημα, τίθεται επίσης για ιεραρχικές δομές χωρικές δεδομένων, όπως το *PM quadtree*. Οι Friedman, Bentley και Finkel (1977) πρότειναν ένα *NN* αλγόριθμο για *k-d-trees* (Bentley (1975)), ο οποίος επαναδιατυπώθηκε αργότερα από τον Sproull (1991).

Οι Roussopoulos, Kelley και Vincent (1995) παρουσίασαν στην εργασία τους έναν αποτελεσματικό *branch-and-bound* αλγόριθμο αναζήτησης για την επεξεργασία των *k-NN* ερωτημάτων στα *R-trees* (Guttman (1984)). Με χρήση μαθηματικών ορισμών, θεωρημάτων και λημμάτων, εισήγαγαν διάφορα μετρικά για την ταξινόμηση και το «κλάδεμα» του δέντρου αναζήτησης και εκτέλεσαν μια σειρά πειραμάτων σε συνθετικά και πραγματικά δεδομένα, για τον υπολογισμό της απόδοσης και της επεκτασιμότητας της προσέγγισής τους.

Στη συνέχεια παρατίθενται μονάχα όσοι ορισμοί, θεωρήματα και λήμματα κρίνονται αναγκαία για την κατανόηση του αλγορίθμου που υλοποίησαν. Η λεπτομερής αναφορά σε αυτά και στις αποδείξεις τους δεν κρίνεται σκόπιμη για τις ανάγκες της παρούσας εργασίας.

<u>Λήμμα 1 (Ιδιότητα της *MBR* όψης)</u>

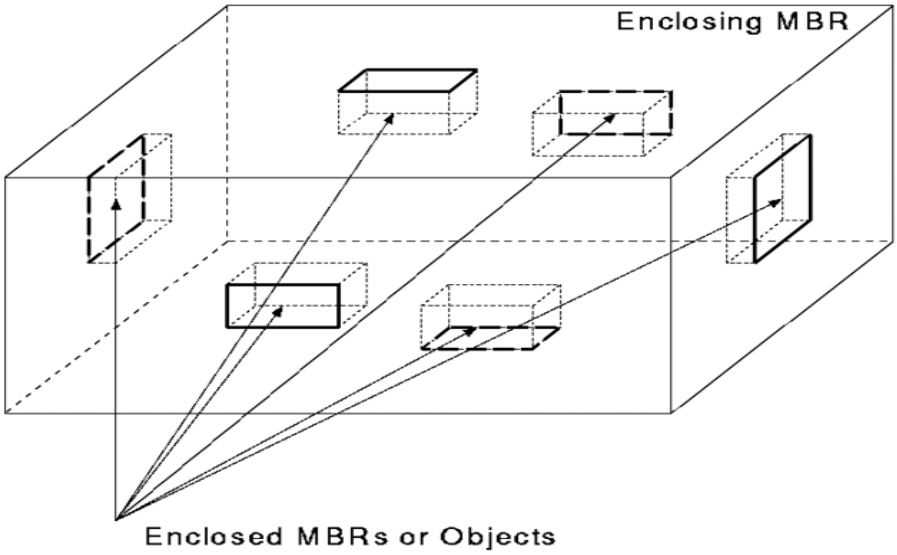
Κάθε όψη ενός *MBR* – άκρο στις δύο διαστάσεις, ορθογώνιο στις τρεις διαστάσεις και υπερ-επίπεδο σε υψηλότερες διαστάσεις – σε οποιοδήποτε επίπεδο του *R-tree*, περιέχει τουλάχιστον ένα σημείο κάποιου χωρικού αντικειμένου της ΒΔ (Εικόνες 14 και 15)



Εικόνα 14: Η ιδιότητα της *MBR* όψης στο χώρο δύο διαστάσεων.

(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

Όπως είναι εμφανές, κάθε άκρο του *MBR* στο επίπεδο *i* είναι σε επαφή με ένα γραφικό αντικείμενο στη βάση δεδομένων.



Εικόνα 15: Η ιδιότητα της *MBR* όψης στο χώρο τριών διαστάσεων

(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

Με βάση την ιδιότητα αυτή, στο στάδιο της ταξινόμησης και του κλαδέματος της αναζήτησης κατά τη διαδικασία εύρεσης του κοντινότερου γείτονα, τα μετρικά που περιγράφονται παρακάτω υπολογίζουν τις αποστάσεις ενός σημείου από τις πλευρές των *MBR*. Η εκτίμηση που γίνεται για την κοντινότερη απόσταση θα είναι αποτελεσματική, αφού η πλευρά αυτή θα εφάπτεται σε τουλάχιστον ένα σημείο με ένα χωρικό αντικείμενο της ΒΔ, που αποτελεί και το βασικό ζητούμενο της διαδικασίας αναζήτησης.

1.2.1. Μετρικά για Αναζήτηση Κοντινότερου Γείτονα

Δεδομένου ενός σημείου αναζήτησης *P* κι ενός αντικειμένου *O* που βρίσκεται μέσα στο *MBR*, παρέχονται δύο μετρικά για την επιλογή της *NN* αναζήτησης. Το πρώτο υπολογίζει την ελάχιστη απόσταση *(MINDIST)* του αντικειμένου *O* από το σημείο *P,* ενώ το δεύτερο υπολογίζει την ελάχιστη από τις μέγιστες πιθανές αποστάσεις *(MINMAXDIST)* του σημείου *P,* από μια πλευρά (ή κορυφή) του *MBR* που περικλείει το αντικείμενο *O*. Τα *MINDIST* και *MINMAXDIST* παρέχουν ένα κατώτατο κι ένα ανώτατο όριο της πραγματικής απόστασης του *O* από το *P* αντίστοιχα. Αυτά τα όρια χρησιμοποιούνται από τον αλγόριθμο αναζήτησης *NN*, ώστε να επιλέξει και να «κλαδέψει» αποτελεσματικά τα μονοπάτια του χώρου αναζήτησης στο *R-tree*.

Minimum Distance (MINDIST)

Το πρώτο μετρικό αποτελεί μια παραλλαγή της κλασσικής Ευκλείδειας απόστασης ορισμένης μεταξύ ενός σημείου κι ενός ορθογωνίου. Εάν το σημείο βρίσκεται μέσα στο ορθογώνιο η μεταξύ τους απόσταση είναι μηδέν, ενώ στην περίπτωση που το σημείο είναι έξω από το ορθογώνιο χρησιμοποιείται το τετράγωνο της Ευκλείδειας απόστασης μεταξύ του σημείου και του κοντινότερου άκρου του ορθογωνίου. Η επιλογή του τετραγώνου της Ευκλείδειας απόστασης γίνεται επειδή απαιτεί λιγότερους και μικρότερου κόστους υπολογισμούς. Στη συνέχεια παρατίθενται δύο χρήσιμοι ορισμοί για την μαθηματική αναπαράσταση και τον τρόπο υπολογισμού του *MINDIST*.

<u>*Ορισμός 1*</u>

Ένα ορθογώνιο R στο Ευκλείδειο χώρο E(n) διάστασης n, θα ορίζεται από τα δύο ακραία σημεία S και T της κύριας διαγωνίου του:

$R = (S,T)$, *όπου* $S = [s_1, s_2, ..., s_n]$ *και* $T = [t_1, t_2, ..., t_n]$ *και* $s_i \le t_i$ *για* $1 \le i \le n$

<u>*Ορισμός 2*</u>

Η απόσταση ενός σημείου P του E(n) από ένα ορθογώνιο R στον ίδιο χώρο, συμβολίζεται ως MINDIST(P,R) και είναι:

$$MINDIST(P,R) = \sum_{i=1}^{n} |p_i - r_i|^2 \text{ όπου } r_i = \begin{cases} s_i & p_i \prec s_i \\ t_i \ \text{εάν} & p_i \succ t_i \\ p_i & \text{διαφορετικά} \end{cases}$$

Αντίστοιχα, ως απόσταση του σημείου *P* από κάποιο χωρικό αντικείμενο *o* υπολογίζεται και πάλι η Ευκλείδεια Απόσταση, όπως δηλώνεται και από τον παρακάτω ορισμό.

Ορισμός 3

Η απόσταση ενός σημείου P του E(n) από ένα χωρικό αντικείμενο o στον ίδιο χώρο, συμβολίζεται ως $\|(P,o)\|$ *και υπολογίζεται από τη σχέση:*

$$\|(P,o)\| = \min\left(\sum_{i=1}^{n}|p_i - x_i|^2, \forall X = [x_1,...,x_n] \in O\right)$$

Με βάση τα παραπάνω, οι Roussopoulos, Kelley και Vincent (1995) κατέληξαν στο θεώρημα που ακολουθεί.

Θεώρημα 1

Δεδομένου ενός σημείου P κι ενός MBR R που περικλείει μια ομάδα αντικειμένων $O = \{o_i\}$ *όπου* $1 \le i \le m$*, ικανοποιείται η συνθήκη:*

$$\forall o \in O, MINDIST(P,R) \le \|(P,o)\|$$

Το θεώρημα αναφέρει ότι η απόσταση $\|(P,o)\|$ μεταξύ ενός σημείου *P* και κάθε αντικειμένου *o* εντός του *MBR R*, είναι τουλάχιστον ίση με το *MINDIST*. Η περίπτωση της ισότητας ικανοποιείται όταν ένα αντικείμενο του *R*, εφάπτεται με τον κύκλο με κέντρο το *P* και ακτίνα ίση με την τετραγωνική ρίζα του *MINDIST*. Συμπερασματικά, το *MINDIST* χρησιμοποιείται για να καθορίσει το κοντινότερο αντικείμενο στο σημείο *P*, από όλα όσα περιλαμβάνονται στο *MBR R*.

Στην πράξη όταν αναζητάμε τους *NN* σε ένα *R-tree* (Guttman (1984)), για κάθε κόμβο που επισκεπτόμαστε πρέπει να αποφασίσουμε ποιό *MBR* θα ψάξουμε πρώτο. Το *MINDIST* προσφέρει μια πρώτη προσέγγιση της *NN* απόστασης σε κάθε *MBR* του κόμβου κι έτσι, μπορεί να χρησιμοποιηθεί ώστε να κατευθύνει την αναζήτηση. Ωστόσο, η επιλογή του *MBR* που θα επισκεφτούμε πρώτα δεν είναι τόσο απλή. Μάλιστα σε πολλές περιπτώσεις, λόγω του «νεκρού χώρου» μέσα στα *MBRs*, η απόσταση του *NN* μπορεί να είναι μεγαλύτερη από το *MINDIST* και η επιλογή του *MBR* με το μικρότερο *MINDIST* μπορεί να οδηγήσει σε *false-drops*, δηλαδή επίσκεψη αχρείαστων κόμβων. Για το λόγο αυτό εισάγεται και το δεύτερο μετρικό, που ονομάζεται *MINMAXDIST*.

Minimax Distance (MINMAXDIST)

Προκειμένου να αποφευχθεί η επίσκεψη περιττών *MBRs*, πρέπει να τεθεί ένα άνω όριο της *NN* απόστασης μεταξύ του σημείου ερωτήματος *P* και κάθε αντικείμενου *o* μέσα στο *MBR*. Αυτό θα επέτρεπε ένα "κλάδεμα" των *MBRs* τα οποία έχουν *MINDIST* μεγαλύτερο από αυτό το άνω όριο. Ο ορισμός της ακόλουθης απόστασης γίνεται προκειμένου να υπολογίζεται η ελάχιστη τιμή από όλες τις μέγιστες αποστάσεις, μεταξύ του σημείου ερωτήματος *P* και των σημείων πάνω σε κάθε έναν από τους *n* άξονες αντίστοιχα. Το *MINMAXDIST* εγγυάται ότι υπάρχει ένα αντικείμενο μέσα στο *MBR* σε απόσταση μικρότερη ή ίση με την τιμή του.

<u>*Ορισμός 3*</u>

Δεδομένου ενός σημείου P στον E(n) και ενός MBR R = (S, T) της ίδιας διάστασης, το MINMAXDIST(P,R) ορίζεται ως:

$$\min_{1\le k\le n}\left(\left|p_k - rm_k\right|^2 + \sum_{\substack{i\ne k\\ 1\le i\le n}}\left|p_i - rM_i\right|^2\right)$$

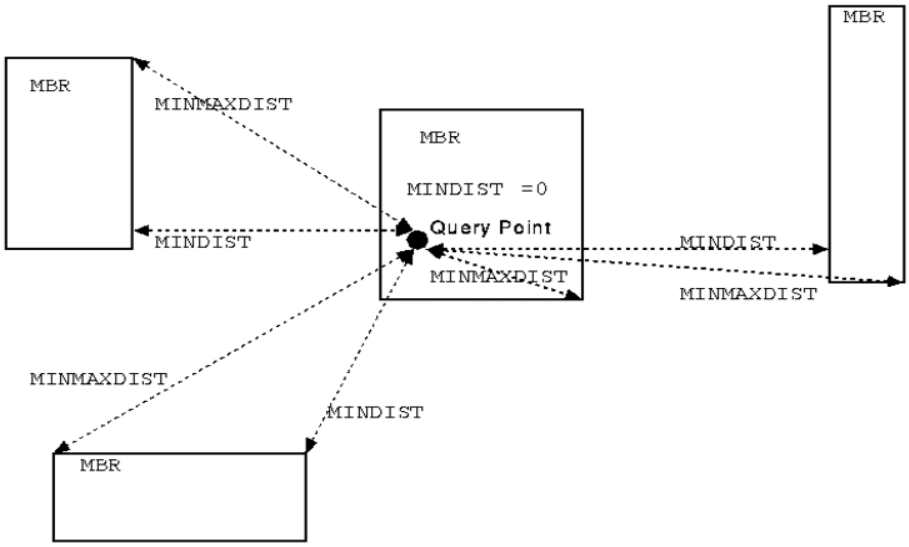
$$\text{όπου } rm_k = \begin{cases} s_k & p_k \le \dfrac{(s_k + t_k)}{2} \\ \quad \text{εάν} & \\ t_k & \text{διαφορετικ ά} \end{cases} \quad \text{και } r\mathrm{M}_i = \begin{cases} s_i & p_i \ge \dfrac{(s_i + t_i)}{2} \\ \quad \text{εάν} & \\ t_i & \text{διαφορετικ ά} \end{cases}$$

<u>*Θεώρημα 2*</u>

Δεδομένου ενός σημείου P κι ενός MBR R το οποίο περικλείει μια ομάδα αντικειμένων $O = \{o_i\}$ *όπου* $1 \le i \le m$*, ικανοποιείται η συνθήκη:*

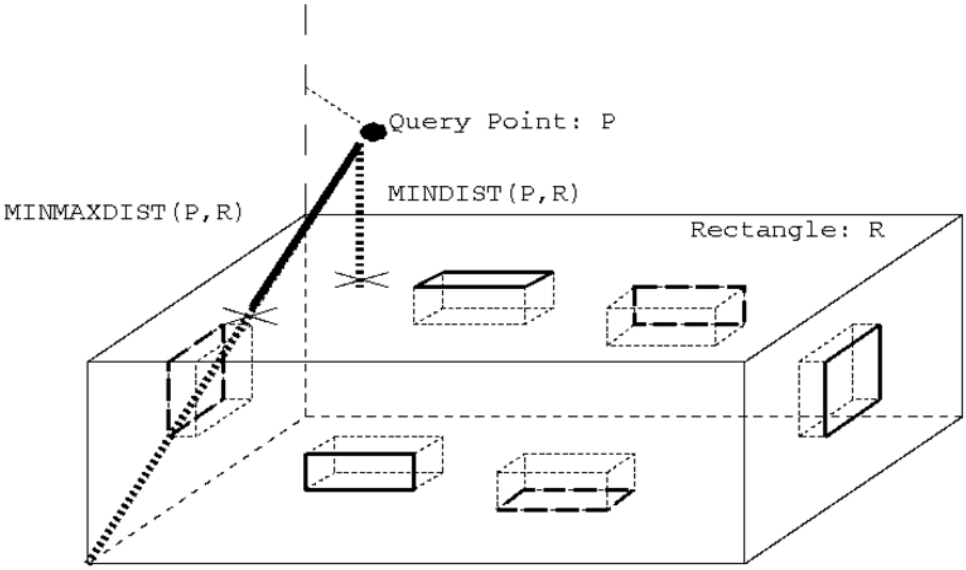
$$\exists o \in O, \|(P,o)\| \le MINMAXDIST(P,R).$$

Το θεώρημα αυτό αναφέρει ότι η *MINMAXDIST* είναι η ελάχιστη απόσταση, που εξασφαλίζει την ύπαρξη ενός αντικειμένου *o* μέσα στο *R* με απόσταση από το *P* $\|(P,o)\|$ μικρότερη από το *MINMAXDIST*. Μια τιμή μεγαλύτερη ή ίση με αυτό θα μπορούσε να "πιάσει" μερικά αντικείμενα εντός ενός *MBR*, αλλά μια μικρότερη απόσταση θα μπορούσε να "χάσει" μερικά από αυτά. Οι εικόνες 16 και 17 αναπαριστούν τα *MINDIST* και *MINMAXDIST* στις δύο και τρεις διαστάσεις αντίστοιχα.



Εικόνα 16: Τα *MINDIST* και *MINMAXDIST* στο χώρο δύο διαστάσεων.

(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)



Εικόνα 17: Τα *MINDIST* και *MINMAXDIST* στο χώρο τριών διαστάσεων.

(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

Συμπερασματικά, μετά τον ορισμό των *MINDIST* και *MINMAXDIST* και με τη βοήθεια και των παραπάνω εικόνων, καταλήγουμε στη σχέση που ισχύει ανάμεσα στο δύο μετρικά:

$$MINDIST(P,R) \leq NN(P) \leq MINMAXDIST(P,R)$$

όπου, με *NN(P)* συμβολίζεται η απόσταση του κοντινότερου γείτονα στο σημείο *P*.

1.2.2. Αλγόριθμος Κοντινότερου Γείτονα

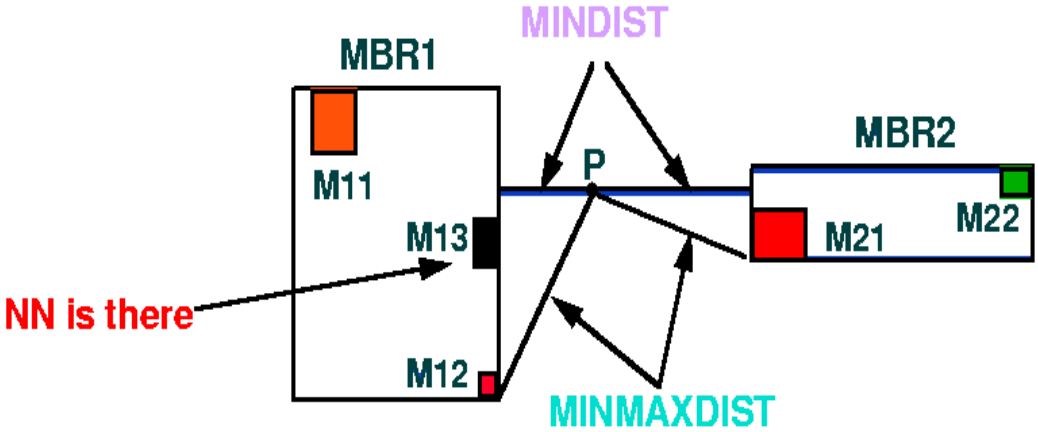
Στη συνέχεια παρουσιάζεται ο *branch and bound* αλγόριθμος διάσχισης ενός *R-tree* για την εύρεση των *k-NN* αντικειμένων σε ένα δεδομένο σημείο ερωτήματος, όπως ορίστηκε από τους Roussopoulos, Kelley και Vincent (1995). Αρχικά αναλύεται η χρήση των μετρικών *MINDIST* και *MINMAXDIST* για την ταξινόμηση και το κλάδεμα του δέντρου αναζήτησης, κατόπιν παρουσιάζεται ο αλγόριθμος για την εύρεση του 1-NN και τέλος, η γενίκευση του αλγορίθμου για την εύρεση των *k-NN*.

1.2.2.1. Ταξινόμηση και Κλάδεμα της Αναζήτησης

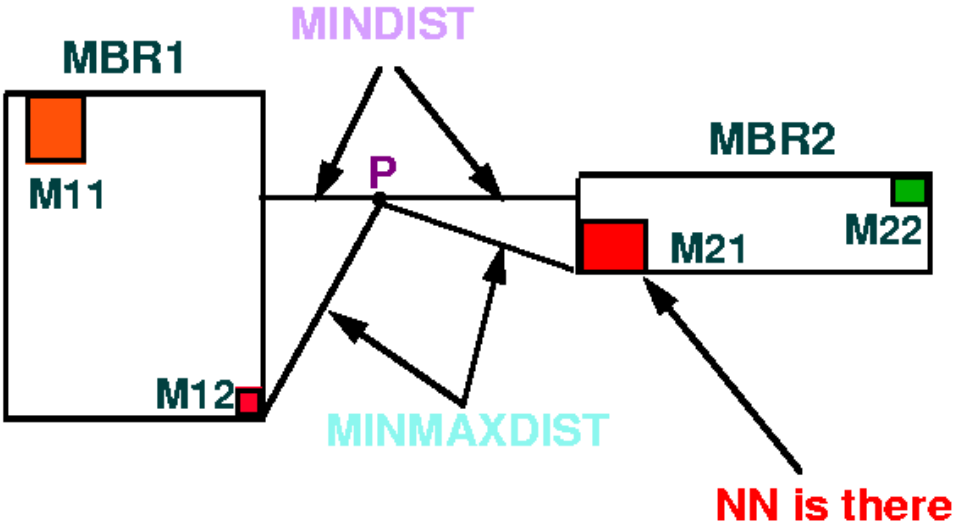
Εάν οι ευριστικές συναρτήσεις για την ταξινόμηση και το κλάδεμα επιλεγούν σωστά, μπορούν να μειώσουν σημαντικά τον αριθμό των κόμβων που επισκέπτονται κατά την αναζήτηση.

Ταξινόμηση Αναζήτησης: Οι ευριστικές συναρτήσεις που χρησιμοποιούνται στον αλγόριθμο των Roussopoulos, Kelley και Vincent (1995) βασίζονται στις ταξινομήσεις λόγω των *MINDIST* και *MINMAXDIST* μετρικών. Η ταξινόμηση με το *MINDIST* μετρικό αποτελεί την αισιόδοξη επιλογή, αφού εκτιμά την απόσταση από το σημείο ερωτήματος σε κάθε *MBR* ή αντικείμενο δεδομένων, ως την ελάχιστη απόσταση από το σημείο στο *MBR* που τα περιλαμβάνει. Λόγω των ιδιοτήτων των *MBRs* και βάσει του ορισμού της, το *MINMAXDIST* μετρικό παράγει την πλέον απαισιόδοξη ταξινόμηση που μπορεί να θεωρήσει κανείς.

Ωστόσο, η βέλτιστη ταξινόμηση της επίσκεψης των *MBR* δεν εξαρτάται μονάχα από την απόσταση μεταξύ του σημείου ερωτήματος και καθενός από τα *MBRs* που βρίσκονται στη διαδρομή (διαδρομές) της αναζήτησης (φύλλα), αλλά και από το μέγεθος και το σχήμα των τελευταίων. Στην πράξη λοιπόν μπορεί κανείς να δημιουργήσει παραδείγματα στα οποία η ταξινόμηση με το *MINDIST* μετρικό παράγει πιο πολύπλοκες διασχίσεις του δέντρου, όσον αφορά το πλήθος των επισκεπτόμενων κόμβων, σε σχέση με το *MINMAXDIST* μετρικό. Οι εικόνες 18 και 19 απεικονίζουν δύο διαφορετικές περιπτώσεις, όπου η κατάλληλη επιλογή του *MBR* που θα επισκεφθεί ο αλγόριθμος γίνεται με τη βοήθεια του *MINDIST* και του *MINMAXDIST* αντίστοιχα.



Εικόνα 18: Η ταξινόμηση με χρήση του *MINDIST* μετρικού καταλήγει γρηγορότερα στον κοντινότερο γείτονα.
(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)



Εικόνα 19: Η ταξινόμηση με χρήση του *MINMAXDIST* μετρικού καταλήγει γρηγορότερα στον κοντινότερο γείτονα.
(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

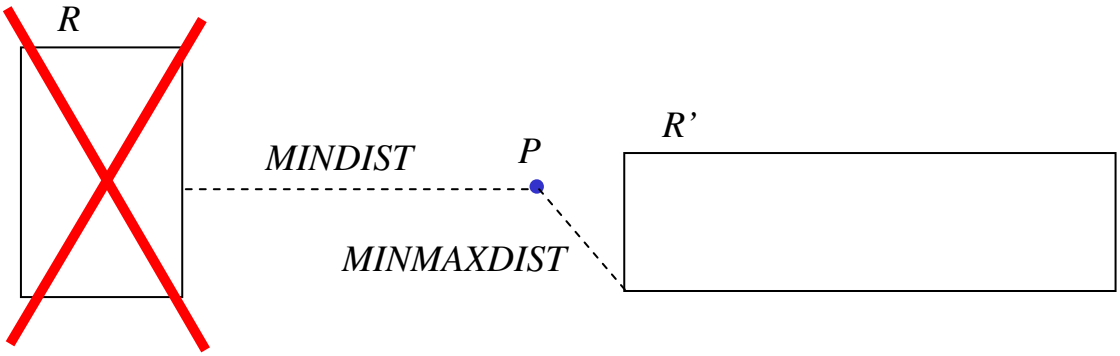
Από την εικόνα 19, φαίνεται ότι η ταξινόμηση με *MINDIST* μετρικό θα οδηγούσε την αναζήτηση στο *MBR1* και την εξέταση των *M11* και *M12*, δημιουργώντας πιο πολύπλοκες διασχίσεις και αυξάνοντας το χρόνο εκτέλεσης. Εάν, όμως χρησιμοποιηθεί η ταξινόμηση του *MINMAXDIST* μετρικού, η επίσκεψη του *MBR2* οδηγεί σε μικρότερη τιμή της απόστασης του *NN* (το οποίο θα βρισκόταν στο *M21*), το οποίο εξαλείφει την ανάγκη να εξεταστεί το *M11* και το *M12*.

Συμπερασματικά λοιπόν, η *MINDIST* ταξινόμηση κάνει την αισιόδοξη υπόθεση ότι οι Κοντινότεροι Γείτονες του σημείου *P* μέσα στο *MBR M* θα είναι πολύ κοντά στο *MINDIST(M,P),* το οποίο όμως δεν είναι πάντα η σωστή περίπτωση. Από την άλλη, η το *MINMAXDIST(P,M)* μετρικό υπολογίζει την ελάχιστη απόσταση μεταξύ του σημείου *P* και του *MBR M*, η οποία εξασφαλίζει την εύρεση ενός αντικειμένου στο *M* σε Ευκλείδεια απόσταση μικρότερη ή ίση με το *MINMAXDIST(P,M)*.

Κλάδεμα Αναζήτησης: τα δύο θεωρήματα που αναπτύχθηκαν παραπάνω, οδηγούν στις ακόλουθες στρατηγικές για το κλάδεμα των *MBR* κατά την αναζήτηση.

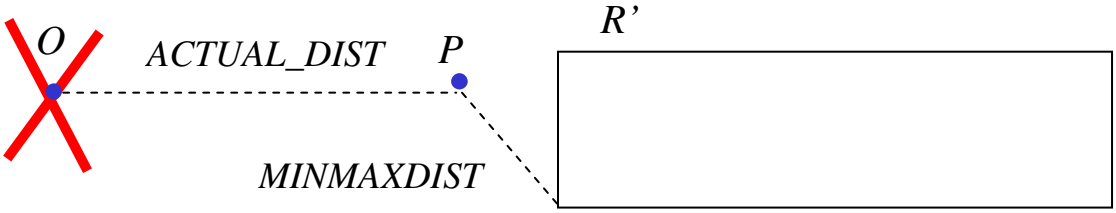
Κλάδεμα "προς τα κάτω"

1. Ένα *MBR R* με απόσταση από το σημείο *P* ίση με *MINDIST(P,R)* απορρίπτεται εάν υπάρχει κάποιο *MBR R'* με απόσταση από το σημείο *P* ίση με *MINDIST(P,R')* τέτοια ώστε *MINDIST(P,R)>MINDIST(P,R')*, αφού δεν μπορεί να περιέχει τους Κοντινότερους Γείτονες. (Θεώρημα 1, 2) (Εικόνα 10)



Εικόνα 20: Το *MBR R* απορρίπτεται από τη στιγμή που η απόσταση *MINDIST(P,R)* από το σημείο *P* είναι μεγαλύτερη από την απόσταση *MINMAXDIST(P,R')* του σημείου από το *MBR R'*.
(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

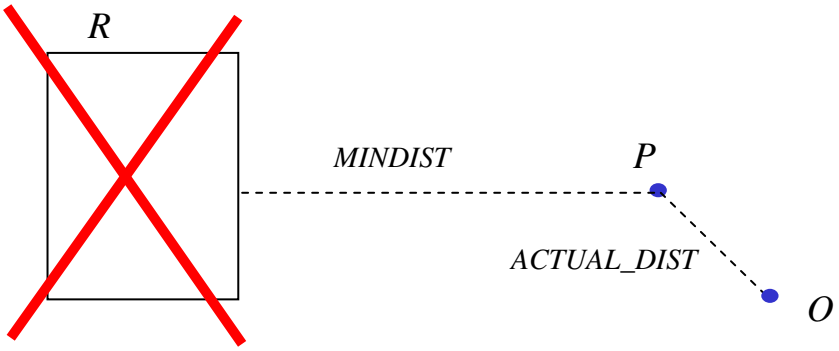
2. Ένα αντικείμενο *O* με πραγματική απόσταση από το σημείο *P* *ACTUAL_DIST(P,O)* απορρίπτεται εάν υπάρχει *MBR R* με απόσταση από το σημείο *P* ίση με *MINMAXDIST(P,R)* τέτοια ώστε *ACTUAL_DIST(P,O)> MINMAXDIST(P,R)*, αφού σε αυτή την περίπτωση το *R* θα περιέχει ένα αντικείμενο *O'* το οποίο θα βρίσκεται κοντινότερα στο σημείο *P*. (Θεώρημα 2) (Εικόνα 21)



Εικόνα 21: Το αντικείμενο *O* απορρίπτεται από τη στιγμή που η πραγματική του απόσταση *ACTUAL_DIST(P,O)* από το σημείο *P* είναι μεγαλύτερη από την απόσταση *MINMAXDIST(P,R')* του σημείου από το *MBR R'*.
(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

Κλάδεμα "προς τα πάνω"

3. Ένα *MBR R* με απόσταση από το σημείο *P* ίση με *MINDIST(P,R)* απορρίπτεται εάν υπάρχει αντικείμενο *O* με πραγματική απόσταση από το σημείο *P* ίση με *ACTUAL_DIST(P,O)* τέτοια ώστε *MINDIST(P,R)* > *ACTUAL_DIST(P,O)*, αφού δεν μπορεί να περιλαμβάνει ένα αντικείμενο *O'* που θα βρίσκεται κοντινότερα στο σημείο *P*. (Θεώρημα 2) (Εικόνα 22)



Εικόνα 22: Το *MBR R* απορρίπτεται από τη στιγμή που η απόσταση *MINDIST(P,R)* από το σημείο *P* είναι μεγαλύτερη από την πραγματική απόσταση *ACTUAL_DIST(P,O)* του σημείου από το αντικείμενο *O*.
(csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

Στην πράξη υπάρχουν περιπτώσεις που είναι καλύτερη η εφαρμογή του *MINDIST* και πιο συγκεκριμένα της τρίτης στρατηγικής. Για παράδειγμα, όταν δεν υπάρχει "νεκρός χώρος" (ή είναι πάρα πολύ μικρός) στους κόμβους του *R-tree* – όπως συμβαίνει στην περίπτωση του *R* tree* (Beckmann et. al. (1990)) – το *MINDIST* αποτελεί μια πολύ καλύτερη εκτίμηση της πραγματικής απόστασης με τους Κοντινότερους Γείτονες, σε

σχέση με το *MINMAXDIST*, για όλα τα επίπεδα του δέντρου. Έτσι θα κλαδέψει σωστά περισσότερα υποψήφια *MBRs* απ' ότι το *MINMAXDIST*.

1.2.2.2. Αλγόριθμος Αναζήτησης Κοντινότερων Γειτόνων

Ο αλγόριθμος αναζήτησης των Roussopoulos, Kelley και Vincent (1995) υλοποιεί μια ταξινομημένη κατά βάθος διάσχιση και περιλαμβάνει τα παρακάτω βήματα:

1. **Αρχικοποίηση της κοντινότερης απόστασης ίση με το άπειρο.**
2. **Προς τα κάτω διάσχιση του δένδρου ξεκινώντας από τη ρίζα του. Σε κάθε νέο επισκεπτόμενο κόμβο (όχι φύλλο), ταξινόμηση όλων των *MBR* σε μια Λίστα Ενεργών Κλάδων *(Active Branch List, ABL)* με τη χρήση των μετρικών ταξινόμησης των *MBR* (*MINDIST* ή *MINMAXDIST*).**
3. **Εφαρμογή των στρατηγικών κλαδέματος 1 και 2 πάνω στην *ABL* για την απομάκρυνση περιττών κλάδων.**
4. **Επανάληψη στην *ABL* έως ότου αυτή να είναι άδεια. Σε κάθε επανάληψη, εφαρμογή του βήματος 3 αναδρομικά στον κόμβο παιδί που αντιστοιχεί στο *MBR* του τρέχοντα κλάδου.**
5. **Σε έναν κόμβο-φύλλο, υπολογισμός της απόστασης κάθε αντικειμένου από το σημείο ερωτήματος, σύγκριση με την τρέχουσα τιμή της κοντινότερης απόστασης και ενημέρωσή της εάν αυτό είναι απαραίτητο.**
6. **Κατά την επιστροφή από την αναδρομή, εφαρμογή της στρατηγικής κλαδέματος 3 για την απόρριψη περιττών κλάδων.**
7. **Όταν η *ABL* αδειάσει, επιστρέφεται ο κοντινότερος γείτονας.**

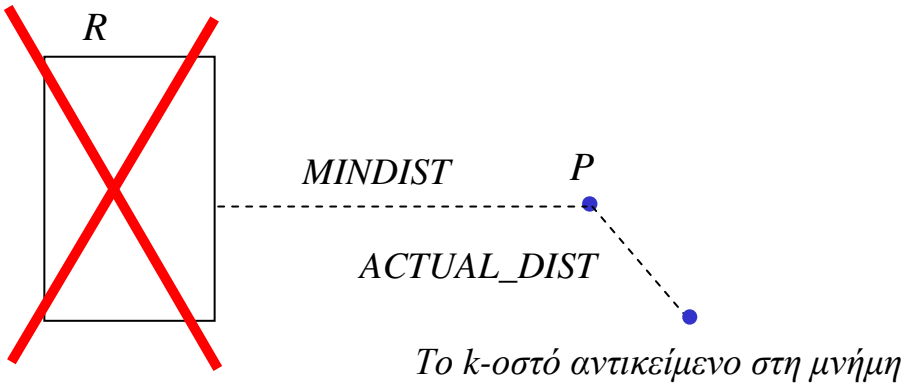
1.2.2.3. Γενίκευση: Εύρεση των *k* Κοντινότερων Γειτόνων

Ο αλγόριθμος που παρουσιάστηκε παραπάνω, μπορεί εύκολα να γενικευτεί για να επιστρέφει μεγαλύτερο αριθμό κοντινότερων γειτόνων. Με άλλα λόγια είναι δυνατή η επέκτασή του ώστε να απαντά σε ερωτήματα του τύπου: βρες τους *k* κοντινότερους γείτονες σε ένα δεδομένο σημείο ερωτήματος, όπου το *k* είναι μεγαλύτερο του μηδενός.

Η μόνες διαφορές είναι:

- Η χρήση μιας ενδιάμεσης μνήμης *(buffer)* για την ταξινόμηση των *k* (το μέγιστο) κοντινότερων γειτόνων κάθε στιγμή.
- Το κλάδεμα των *MBRs* γίνεται με βάση την απόσταση του πιο απομακρυσμένου κοντινότερου γείτονα μέσα σε αυτή τη μνήμη. (Εικόνα 23)

Με βάση τα παραπάνω, η εικόνα 22 τροποποιείται ώστε να περιγράφει τη γενική περίπτωση και η νέα μορφή παρατίθεται στη συνέχεια. (Εικόνα 23)



Εικόνα 23: Για την απόρριψη ενός *MBR* γίνεται η σύγκριση της απόστασής του *MINDIST(P,R)* από το σημείο *P*, με την πραγματική απόσταση *ACTUAL_DIST(P,*O_k*)* του σημείου από το *k*-οστό αντικείμενο O_k που βρίσκεται στη μνήμη. (csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt)

1.3. Ομαδοποίηση Δεδομένων Εισαγωγής

Το *R*-tree* (Beckmann et. al. (1990)) ως εξέλιξη του *R-tree* (Guttman (1984)) διατήρησε το δυναμικό χαρακτήρα της δομής. Η κατασκευή του δέντρου ξεκινά από τη ρίζα και γίνεται σταδιακά με την εισαγωγή κάποιου δεδομένου, επιτρέποντας έτσι προσθαφαιρέσεις δεδομένων σε κάθε στάδιο της δημιουργίας του. Αυτή η λειτουργία κρίνεται απαραίτητη για την ευρετηριοποίηση Δυναμικών ΧΒΔ, αφού οι εγγραφές τους μπορούν να τροποποιούνται ανά πάσα στιγμή, κάτι που συνεπάγεται και αντίστοιχη τροποποίηση του ευρετηρίου.

Στον αντίποδα, η σταδιακή κατασκευή της δομής με την εισαγωγή ενός αντικειμένου κάθε φορά, παρουσιάζει τα παρακάτω μειονεκτήματα:

- υψηλό χρόνο εισαγωγής δεδομένων
- μειωμένη χρήση του χώρου στη μνήμη
- πιθανή ακατάλληλη δομή

Πράγματι ο χρόνος που απαιτείται για την κατασκευή της δομής είναι μεγάλος, αφού για κάθε εισαγωγή απαιτείται η εύρεση της κατάλληλης διαδρομής από τη ρίζα έως τα φύλλα του δένδρου. Έτσι λοιπόν ακόμη κι αν δύο εγγραφές είναι γειτονικές, η αναζήτηση της κατάλληλης θέσης στο δέντρο θα γίνει εξ' ολοκλήρου για κάθε μια από αυτές. Όσον αφορά τη χρήση της διαθέσιμης μνήμης, η κατασκευή του δέντρου βήμα προς βήμα οδηγεί στη δημιουργία κόμβων και φύλλων με τον ελάχιστο αριθμό εγγραφών. Η δομή διατηρεί αυτή τη μορφή, αφού τα κριτήρια πληρότητας για τους κόμβους και τα φύλλα ικανοποιούνται, χωρίς να λαμβάνει υπόψιν πιθανές συγχωνεύσεις κόμβων ή φύλλων, με στόχο τη μείωση του χώρου που καταλαμβάνεται στη μνήμη. Είναι φανερό λοιπόν ότι η χρήση του αποθηκευτικού χώρου δεν γίνεται αποδοτικά. Τέλος, ο τρόπος με τον οποίο κατασκευάζεται το δένδρο καθιστά τη δομή επιρρεπή σε "εσφαλμένο χτίσιμο" αφού οι πρώτες εισαγωγές καθορίζουν σε μεγάλο ποσοστό την τελική της μορφή. Συνεπώς, πιθανές ακατάλληλες εγγραφές στα πρώτα στάδια της δημιουργίας του δένδρου θα οδηγήσουν σε μια δομή που θα είναι αναποτελεσματική, αφού σε μια τέτοια περίπτωση θα απαιτείται η ανάκτηση ενός υπερβολικά μεγάλου αριθμού κόμβων για να απαντηθεί ένα ερώτημα.

Για να επιλυθούν τα παραπάνω προβλήματα, αναπτύχθηκαν κάποιοι δυναμικοί αλγόριθμοι (Beckmann et. al. (1990); Selis, Roussopoulos και Faloutsos (1987)) με

στόχο τη βελτίωση της ποιότητας της δομής κατά την κατασκευή της. Παράλληλα όμως, αναπτύχθηκε και η ιδέα ότι μια κατάλληλη προ-επεξεργασία των δεδομένων πριν την εισαγωγή τους στη δομή, θα οδηγούσε σε σημαντική βελτίωσή της. Ως αποτέλεσμα υλοποιήθηκαν κάποιοι αλγόριθμοι εισαγωγής δεδομένων που υποστηρίζουν την προ-επεξεργασία τους πριν την αποθήκευση στη δομή. Συγκριτικές μελέτες των δύο παραπάνω διαφορετικών προσεγγίσεων έδειξαν ότι, οι δυναμικοί αλγόριθμοι με στόχο τη "διόρθωση" της δομής δεν είναι ανταγωνιστικοί σε σχέση με το χρόνο που απαιτείται για την εκτέλεση ενός ερωτήματος όταν συγκρίνονται με τους αλγορίθμους φορτώματος που επιτρέπουν προ-επεξεργασία των δεδομένων.

Η προ-επεξεργασία είναι λογική για εφαρμογές στις οποίες τα δεδομένα είναι αρκετά στατικά (δηλ. δεν μεταβάλλονται συχνά) ή διαθέσιμα εκ των προτέρων, όπως στις Αποθήκες Δεδομένων. Όταν η προ-επεξεργασία των δεδομένων υλοποιηθεί κατάλληλα, οδηγεί σε δομές με σχεδόν 100% χρήση του χώρου και βελτιωμένους χρόνους ερωτημάτων. Το τελευταίο είναι εύλογο εξαιτίας του γεγονότος ότι χρειάζεται να προσπελαστούν λιγότεροι κόμβοι κατά την εκτέλεση ενός ερωτήματος. Τέτοιου είδους αλγόριθμοι "πακεταρίσματος" προτάθηκαν αρχικά από τους Roussopoulos και Leifker (1985) και αργότερα από τους Kamel και Faloutsos (1993).

Οι Kamel και Faloutsos (1993) πρότειναν έναν αλγόριθμο "πακεταρίσματος" που βασίζεται στην *Hilbert Curve* ταξινόμηση και έκαναν τη σύγκριση με τον αλγόριθμο *Nearest-X* που προτάθηκε από τους Roussopoulos και Leifker (1985). Ο τελευταίος είναι απλούστερος στην εφαρμογή και σε κάποιες περιπτώσεις οδηγεί σε καλύτερα δέντρα για ερωτήματα σημείων. Ωστόσο, λόγω της μικρότερης περιμέτρου των *MBRs* στους εσωτερικούς κόμβους, ο αλγόριθμος των Kamel και Faloutsos (1993) υπερτερεί σημαντικά σε ερωτήματα περιοχής. Ως συνέπεια, ο *Hilbert-based* αλγόριθμος ομαδοποίησης αποτελεί την καλύτερη επιλογή για ερωτήματα περιοχής ενώ παραμένει ανταγωνιστικός και για ερωτήματα σημείων.

Τα επόμενα χρόνια, η έρευνα σχετικά με την ομαδοποιημένη και μαζική εισαγωγή δεδομένων έχει απασχολήσει μεγάλο κομμάτι της ερευνητικής κοινότητας και οδήγησε σε πληθώρα νέων μεθόδων. Οι Arge, Hinrichs, Vahrenhold και Vitter (2002) εισήγαγαν μια τεχνική με χρήση *buffer* και υλοποίησαν το *BR-tree* για μαζική ενημέρωση δεδομένων και μαζικά ερωτήματα σε πολυδιάστατες ΒΔ. Οι Bercken, Seeger και Widmayer (1996) όρισαν ρουτίνες διάσπασης και συγχώνευσης σελίδων, καθώς και μεθόδους ταυτόχρονης εισαγωγής εγγραφών. Οι *Lee, Moon* και *Lee* (2006)

χρησιμοποίησαν ένα *seed tree* για την ταξινόμηση των δεδομένων σε *clusters* από τα οποία κατασκευάζονται τα αντίστοιχα *R-trees* και εισήγαγαν τα τελευταία σε ένα *target R-tree* για την κατασκευή της δομής. Κάποιες εργασίες αντιμετωπίζουν το πρόβλημα της ομαδοποιημένης εισαγωγής με μεθόδους παράλληλης επεξεργασίας (Qin et. al. (2009); Papadopoulos και Manolopoulos (2003)). Ο αναγνώστης μπορεί να ανατρέξει και σε εργασίες που περιγράφουν νέες δομές για ομαδική εισαγωγή δεδομένων, όπως το *TPR-tree* (Lin, Su (2004)) που ορίζεται για την περίπτωση κινούμενων αντικειμένων και το *CRB-tree* (Govindarajan, Agarwal και Arge (2003)) που απαντά σε ερωτήματα συνολικού εύρους.

Οι Leutenegger, Lopez και Edgington (1997) πρότειναν έναν αλγόριθμο με όνομα *Sort-Tile-Recursive*, ο οποίος είναι απλός στην εφαρμογή του και συγκρίνεται με τους *Hilbert* και *Nearest-X* αλγορίθμους για ένα ευρύ πεδίο δεδομένων και μεγεθών της ενδιάμεσης μνήμης. Στην εργασία τους οι συγγραφείς ασχολήθηκαν με την περίπτωση του *R-tree* (Guttman (1984)), θεωρώντας ότι το αρχείο δεδομένων αποτελείται από *r* ορθογώνια και κάθε κόμβος του δένδρου μπορεί να περιέχει μέχρι *n* ορθογώνια. Η γενική διαδικασία είναι παρόμοια με την κατασκευή ενός *B-tree* (Comer (1979)) από μια συλλογή κλειδιών, δημιουργώντας αρχικά το επίπεδο φύλλων και ακολούθως κάθε διαδοχικό ανώτερο επίπεδο έως ότου φτάσουμε στη ρίζα.

1.3.1. Γενικός Αλγόριθμος

Οι αλγόριθμοι ομαδοποίησης που αναφέρθηκαν παραπάνω, διαφέρουν μονάχα στον τρόπο ταξινόμησης των ορθογωνίων σε κάθε επίπεδο. Τα βασικά στάδια σε κάθε έναν από αυτούς είναι τα ακόλουθα.

1. Προ-επεξεργασία του αρχείου δεδομένων έτσι ώστε τα *r* ορθογώνια να είναι ταξινομημένα σε *[r/n]* διαδοχικές ομάδες *n* ορθογωνίων, όπου κάθε ομάδα προορίζεται να τοποθετηθεί στον ίδιο κόμβο του επιπέδου φύλλων (η τελευταία ομάδα μπορεί να περιλαμβάνει λιγότερα από *n* ορθογώνια).

2. "Φόρτωμα" των *[r/n]* ομάδων ορθογωνίων σε σελίδες μνήμης και καταγραφή του (*MBR*, αριθμός-σελίδας) για κάθε σελίδα επιπέδου φύλλων σε ένα προσωρινό αρχείο. Οι αριθμοί-σελίδων χρησιμοποιούνται ως δείκτες στους κόμβους "παιδία", των κόμβων του αμέσως ανώτερου επιπέδου.

3. Ομαδοποίηση αναδρομικά των MBRs σε κόμβους στο επόμενο επίπεδο, ενεργώντας προς τα πάνω, έως ότου κατασκευαστεί ο κόμβος ρίζα.

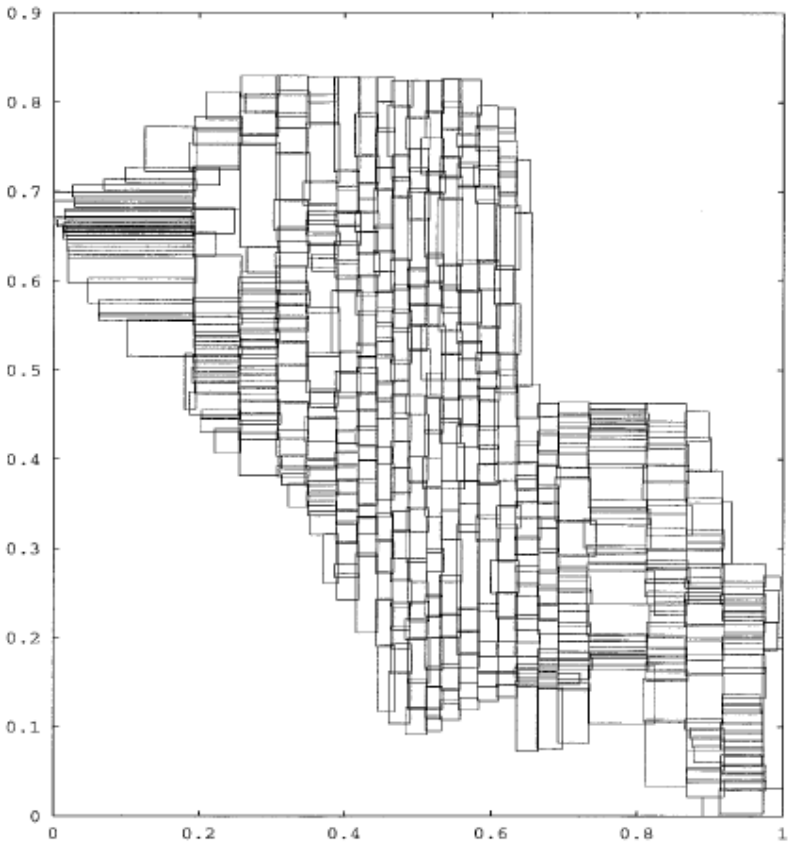
1.3.2. Ο Αλγόριθμος *Sort-Tile-Recursive (STR)*

Αρχικά θεωρείται ένα *k*-διάστατο σύνολο δεδομένων από *r* υπερ-ορθογώνια. Ένα υπερ-ορθογώνιο προσδιορίζεται από *k* διαστήματα της μορφής $[A_i, B_i]$ και αποτελεί το χώρο σημείων των οποίων η *i*-οστή συντεταγμένη βρίσκεται μέσα στο *i*-οστό διάστημα, για όλα τα $1 \le i \le k$.

Ο αλγόριθμος *STR* περιγράφεται καλύτερα αναδρομικά με το $k = 2$ να αποτελεί την βασική περίπτωση. Επομένως, θεωρείται σε πρώτη φάση ένα σύνολο ορθογωνίων στο επίπεδο. Η βασική ιδέα είναι η κάλυψη του χώρου δεδομένων με "πλάκες" χρησιμοποιώντας $\sqrt{r/n}$ κάθετες φέτες, έτσι ώστε κάθε φέτα να περιέχει αρκετά ορθογώνια για να πακετάρει το πολύ $\sqrt{r/n}$ κόμβους. Και πάλι γίνεται η θεώρηση ότι οι συντεταγμένες αναφέρονται στα κεντρικά σημεία των ορθογωνίων. Καθορίζεται ο αριθμός των σελίδων στο επίπεδο των φύλλων $P = \lceil r/n \rceil$ και $S = \lceil \sqrt{P} \rceil$. Κατόπιν γίνεται ταξινόμηση των ορθογωνίων κατά την *x*-συντεταγμένη και ο διαχωρισμός τους σε *S* κάθετες φέτες. Μια φέτα αποτελείται από μια προσπέλαση $S \cdot n$ συνεχόμενων ορθογωνίων από την ταξινομημένη λίστα (η τελευταία μπορεί να περιλαμβάνει λιγότερα από $S \cdot n$ ορθογώνια). Στη συνέχεια γίνεται ταξινόμηση των ορθογωνίων κάθε φέτας κατά την *y*-συντεταγμένη και πακετάρισμα σε κόμβους ομαδοποιώντας τα ορθογώνια σε προσπελάσεις μήκους *n* (τα πρώτα *n* ορθογώνια στον πρώτο κόμβο, τα επόμενα *n* στον δεύτερο κόμβο και ούτω καθεξής).

Η περίπτωση για $k > 2$ είναι μια απλή γενίκευση της προσέγγισης που περιγράφηκε παραπάνω. Αρχικά, γίνεται ταξινόμηση των υπερ-ορθογωνίων με βάση την πρώτη συντεταγμένη του κέντρου τους. Κατόπιν το σύνολο εισόδου διαχωρίζεται σε $S = \left\lceil P^{\frac{1}{k}} \right\rceil$ πλάκες, όπου κάθε μια αποτελείται από μια προσπέλαση από $n \cdot \left\lceil P^{\frac{k-1}{k}} \right\rceil$ συνεχόμενα υπερ-ορθογώνια από την ταξινομημένη λίστα. Κάθε πλάκα πλέον επεξεργάζεται αναδρομικά χρησιμοποιώντας τις υπόλοιπες $k-1$ συντεταγμένες (δηλ, αντιμετωπίζεται ως ένα $(k-1)$-διάστατο σύνολο δεδομένων).

Για την ευκολότερη κατανόηση της λειτουργίας του αλγορίθμου, οι Leutenegger, Lopez και Edgington (1997) υλοποίησαν μια σειρά από πειραματικές μετρήσεις και οπτικοποίησαν τα αποτελέσματά τους. Ως δεδομένα θεώρησαν τους κόμβους του επιπέδου των φύλλων που λαμβάνονται χρησιμοποιώντας το *Long Beach Tiger* σύνολο δεδομένων, υποθέτοντας 100 ορθογώνια σε κάθε κόμβο. Η εικόνα 24 απεικονίζει τα *MBR* του επιπέδου των φύλλων που προκύπτουν για το παραπάνω σύνολο δεδομένων. Στην εικόνα αυτή είναι ευκρινείς οι κάθετες φέτες που σχηματίστηκαν από τον *STR* αλγόριθμο ομαδοποίησης.

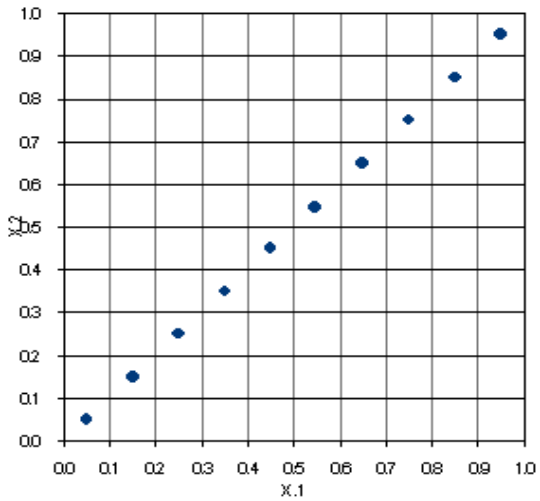


Εικόνα 24: Τα ορθογώνια οριοθέτησης στο επίπεδο των φύλλων για τα *Long Beach Data* με τη χρήση του *STR*..(Leutenegger, Lopez και Edgington (1997))

1.4. Η "Κατάρα της Διαστατικότητας"

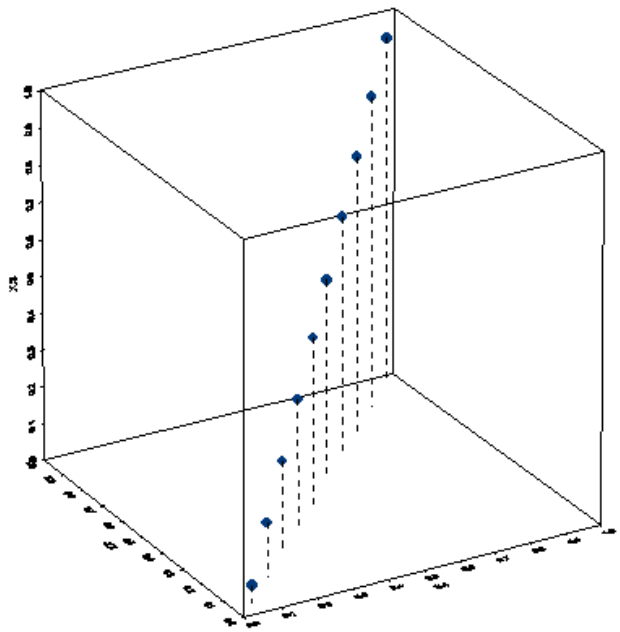
Η "κατάρα της διαστατικότητας" αναφέρεται σε διάφορα φαινόμενα που προκύπτουν κατά την ανάλυση και οργάνωση πολυδιάστατων χώρων, τα οποία δεν συναντώνται σε περιπτώσεις χαμηλών διαστάσεων, όπως ο φυσικός χώρος στον οποίο κινούμαστε ο οποίος μοντελοποιείται με τρεις διαστάσεις. Υπάρχουν διάφορα φαινόμενα που αναφέρονται με αυτό τον όρο σε διάφορα πεδία όπως η δειγματοληψία, η συνδυαστική ανάλυση, η μηχανική μάθηση και η εξόρυξη δεδομένων. Το κοινό σημείο αυτών των προβλημάτων είναι το γεγονός ότι όταν αυξάνεται η διαστατικότητα, ο όγκος του χώρου αυξάνεται τόσο γρήγορα ώστε τα διαθέσιμα δεδομένα να αραιώνουν και να καθίστανται προβληματικά για κάθε μέθοδο που απαιτεί στατιστική σημασία. Για να επιτευχθεί ένα στατιστικά ασφαλές και αξιόπιστο αποτέλεσμα, το ποσό των δεδομένων που απαιτείται, συχνά αυξάνεται εκθετικά με βάση τη διαστατικότητα.

Στο http://www.statisticalengineering.com/curse_of_dimensionality.htm θεωρείται ως χώρος πιθανότητας μιας μεταβλητής το μοναδιαίο διάστημα (0, 1), και μέσα σε αυτόν δέκα δείγματα. Καθένα από αυτά αντιπροσωπεύει το 10% του χώρου πιθανοτήτων (κατά μέσο όρο). Στη συνέχεια θεωρείται μια δεύτερη μεταβλητή που ορίζεται σε ένα ορθογώνιο διάστημα (0, 1) και αποτελείται κι αυτή από δέκα δείγματα. Δημιουργούνται έτσι 10 (x_1, x_2) σημεία στο επίπεδο που ορίζεται από τους ορθογώνιους άξονες x_1 , x_2 και αναπαριστά το νέο χώρο πιθανοτήτων. Όμως ο τελευταίος έχει χώρο 10 x 10 = 100 μονάδες επιφάνειας, κι έτσι καθένα από τα δέκα σημεία αντιπροσωπεύει πλέον το 1% του χώρου πιθανοτήτων (Εικόνα 25).



Εικόνα 25: Ο χώρος πιθανοτήτων στην περίπτωση των δύο διαστάσεων.

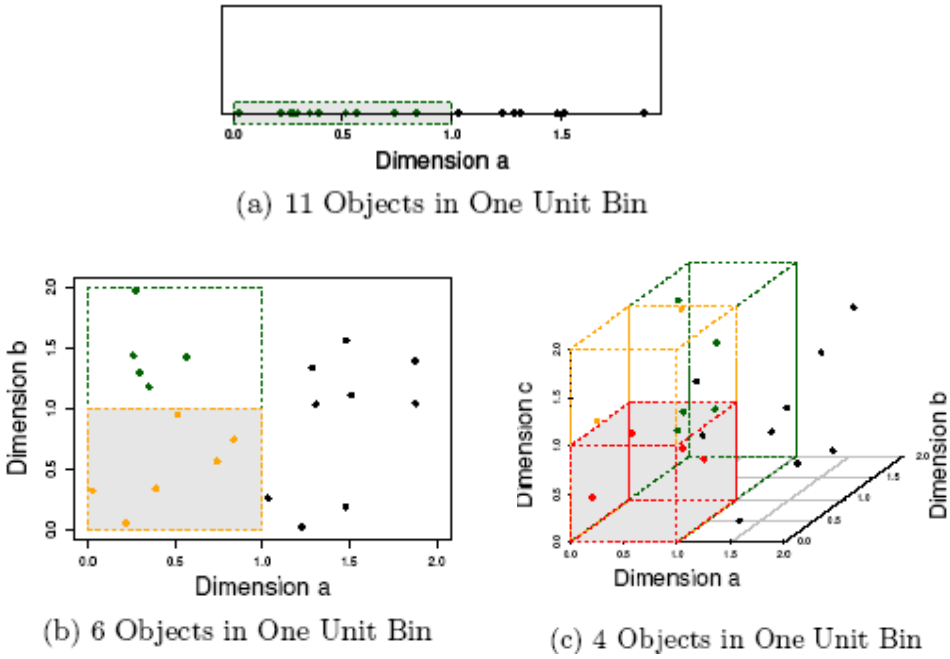
Για την αναπαράσταση του 10% θα ήταν απαραίτητα 100 σημεία και συνεπώς, ο αριθμός των σημείων πρέπει να αυξηθεί εκθετικά ώστε να διατηρήσει μια δεδομένη ακρίβεια. Στη συνέχεια θεωρείται ότι προστίθεται μια ακόμη διάσταση, δημιουργώντας έτσι χώρο πιθανοτήτων που αναπαρίσταται από ένα κύβο με 10 μονάδες απόστασης σε κάθε πλευρά και 1000 μονάδες πιθανότητας στο εσωτερικό του. Τώρα απαιτείται το στοίβαγμα 10 επιπέδων (x_1, x_2) ώστε να δημιουργηθεί ο νέος κύβος (x_1, x_2, x_3) και φυσικά 10 φορές το πλήθος των δειγμάτων σε κάθε άξονα, ώστε να διατηρηθεί το προηγούμενο επίπεδο ακρίβειας, ή η κάλυψη πιθανότητας. Είναι φανερό ότι απαιτούνται 10^n δείγματα για ένα n-διάστατο πρόβλημα, το οποίο στη βιβλιογραφία αναφέρεται ως "κατάρα της διαστατικότητας" και θέτει ένα πρακτικό όριο περίπου στις 6 διαστάσεις για τη απευθείας δειγματοληψία Monte Carlo.



Εικόνα 26: Ο χώρος πιθανοτήτων στην περίπτωση των τριών διαστάσεων.

Η οργάνωση και η αναζήτηση δεδομένων συχνά στηρίζεται στην εύρεση περιοχών όπου τα αντικείμενα δημιουργούν ομάδες με κοινά χαρακτηριστικά. Στα πολυδιάστατα δεδομένα ωστόσο, όλα τα αντικείμενα εμφανίζονται να είναι αραιά και ανόμοια με πολλούς τρόπους, με αποτέλεσμα οι κοινές στρατηγικές οργάνωσης δεδομένων να καθίστανται αναποτελεσματικές. Στις παρακάτω εικόνες, τα δεδομένα εμφανίζονται σχετικά πυκνά σε μια διάσταση. Η πρόσθεση μιας διάστασης "εκτείνει" τα σημεία κατά μήκος μιας διάστασης, κάνοντάς τα πιο απομακρυσμένα. Αυξάνοντας το πλήθος των διαστάσεων τα σημεία τοποθετούνται ακόμη πιο μακριά, με αποτέλεσμα τα πολυδιάστατα δεδομένα να είναι εξαιρετικά αραιά μεταξύ τους. Τα

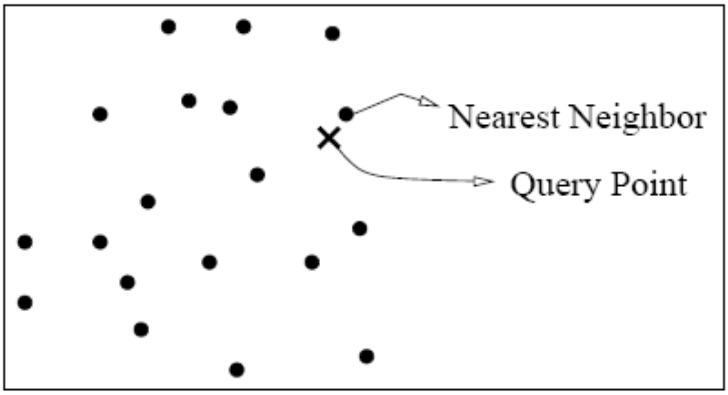
μετρικά απόστασης καθίστανται αναποτελεσματικά, λόγω του ότι όλα τα σημεία φαίνονται να βρίσκονται στην ίδια απόσταση.



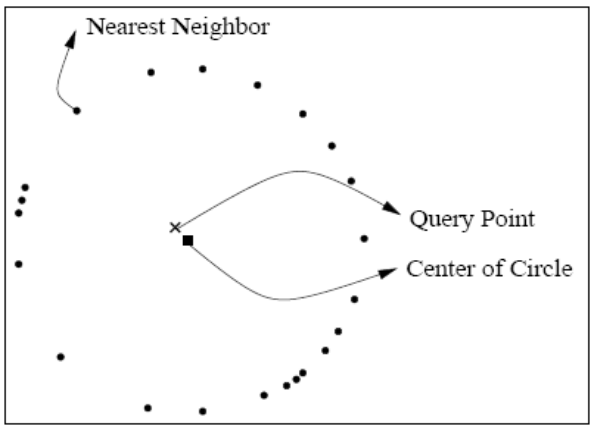
Εικόνα 27: Πυκνότητα σημείων για χώρο (a) μίας, (b) δύο, (c) τριών διαστάσεων. (Parsons et al. (2004))

1.4.1. Το ερώτημα Κοντινότερου Γείτονα σε υψηλές διαστάσεις

Το ερώτημα *NN* αφορά τον προσδιορισμό του σημείου του χώρου δεδομένων, που βρίσκεται πλησιέστερα σε ένα σημείο ερωτήματος. Ωστόσο, δεν υπάρχει πάντοτε μια απάντηση με πραγματική σημασία σε κάποια τέτοια ερώτηση (Εικόνες 28 και 29).



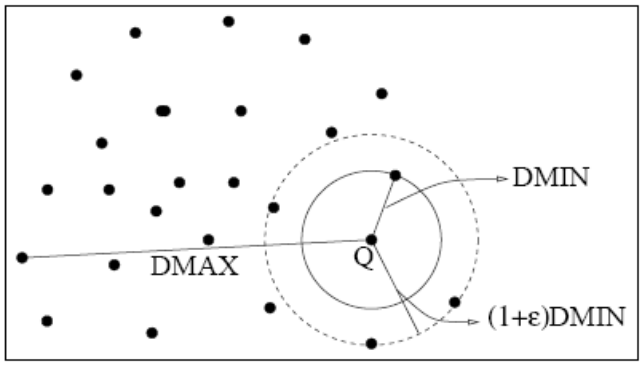
Εικόνα 28: Το σημείο ερωτήματος και ο κοντινότερος γείτονας. (Beyer et. al. (1999))



Εικόνα 29: Το σημείο ερωτήματος και ο κοντινότερος γείτονας. (Beyer et. al. (1999))

Στην παραπάνω εικόνα, αν και υπάρχει ένας διακριτός κοντινότερος γείτονας η διαφορά της απόστασης από το σημείο ερωτήματος μεταξύ αυτού και οποιουδήποτε άλλου σημείου είναι πολύ μικρή, με αποτέλεσμα η απόδοση σε κάποια προβλήματα να είναι χαμηλή. Το σενάριο αυτό είναι πολύ σπάνιο σε ένα *GIS* (και σε οποιοδήποτε πρακτική εφαρμογή του *NN* σε δύο διαστάσεις), ωστόσο στο (Beyer et. al. (1999)) αποδεικνύεται ότι αποτελεί τον κανόνα για ένα πλήθος κατανομών δεδομένων, στις υψηλές διαστάσεις.

Οι συγγραφείς, εξετάζουν το πλήθος των σημείων που περικλείονται από μια σφαίρα ερωτήματος μεγεθυμένη κατά έναν παράγοντα ε (Εικόνα 30). Εάν τα σημεία που βρίσκονται εντός της μεγεθυμένης σφαίρας είναι λίγα, σημαίνει ότι το σημείο που είναι κοντινότερα στο σημείο ερωτήματος διαχωρίζεται από τα υπόλοιπα δεδομένα ορθά. Από την άλλη, εάν τα περισσότερα (ή και όλα) σημεία εμπίπτουν μέσα σε αυτή τη σφαίρα, ο διαχωρισμός του κοντινότερου γείτονα από τα υπόλοιπα σημεία δεν έχει κανένα νόημα στην περίπτωση που το ε είναι μικρό.

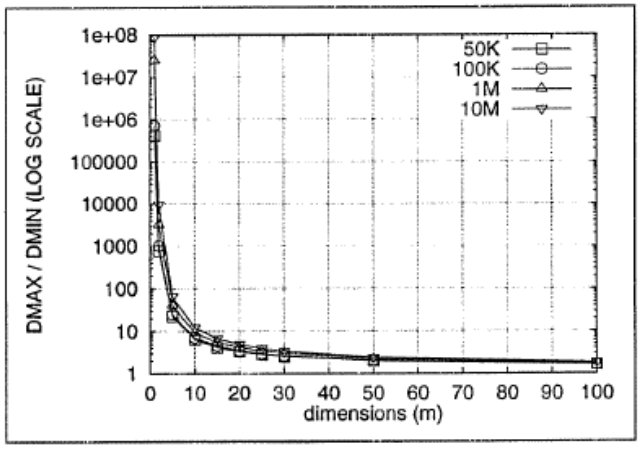


Εικόνα 30: Περιοχή ερωτήματος και μεγεθυμένη περιοχή. Με *DMIN* αναπαρίσταται η απόσταση του κοντινότερου γείτονα και *DMAX* η απόσταση του πιο απομακρυσμένου σημείου. (Beyer et. al. (1999))

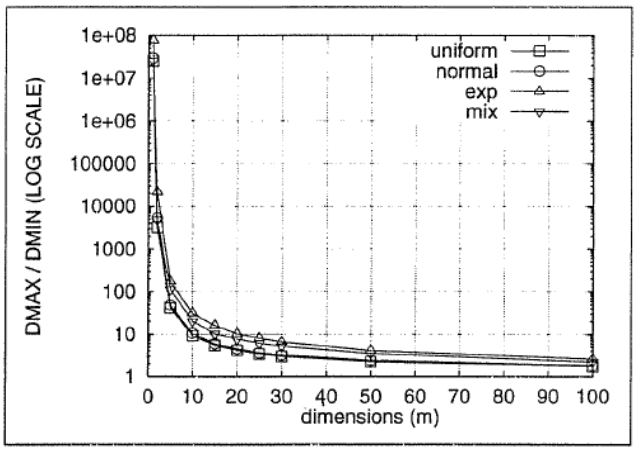
Για την περιγραφή της θεωρίας τους, παραθέτουν τον ορισμό:

Ορισμός 1: *Ένα ερώτημα κοντινότερου γείτονα είναι "ασταθές" για δεδομένο ε εάν η απόσταση από το σημείο ερωτήματος προς τα περισσότερα σημεία δεδομένων είναι μικρότερη κατά (1 + ε) φορές, από την απόσταση μεταξύ του σημείου ερωτήματος και του κοντινότερου γείτονα.*

Οι Beyer et. al. (1999) αποδεικνύουν ότι σε πολλές περιπτώσεις, για οποιοδήποτε σταθερό ε > 0, η πιθανότητα ενός ερωτήματος να είναι "ασταθές" με την αύξηση της διαστατικότητας, συγκλίνει στο 1. Συγκεκριμένα μελέτησαν την επίδραση της διαστατικότητας σε ερωτήματα Κοντινότερου Γείτονα. Οι κατανομές που εξέτασαν ήταν η Ομοιόμορφη, η Κανονική και η Εκθετική. Ως μέγεθος του αρχείου δεδομένων επιλέχθηκαν οι 50.000, 100.000, 1.000.000 και 10.000.000 εγγραφές. Σε κάθε περίπτωση, υπολόγισαν τον λόγο *DMAX / DMIN* και τα αποτελέσματά τους απεικονίζονται στα παρακάτω διαγράμματα.



Εικόνα 31: Η Ομοιόμορφη κατανομή για τα εξεταζόμενα πλήθη δεδομένων. (Beyer et. al. (1999))



Εικόνα 32: Οι κατανομές στην περίπτωση 1.000.000 εγγραφών. (Beyer et. al. (1999))

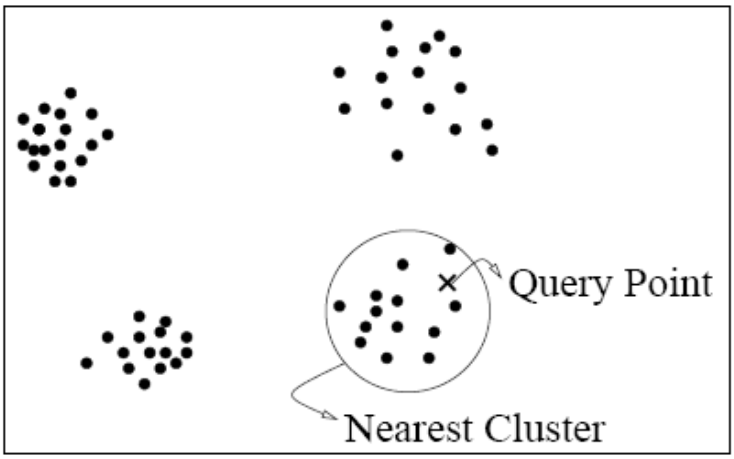
Η εικόνες αυτές δείχνουν σε λογαριθμική κλίμακα την τιμή του *DMAX / DMIN* καθώς αυξάνεται η διάσταση *m*. Ο λόγος αυτός στην περίπτωση της μιας διάστασης είναι της τάξης του 10^8 , προσδίδοντας μεγάλη διαφορά ανάμεσα στο κοντινότερο και το μακρινότερο αντικείμενο. Στις 10 διαστάσεις, η διαφορά αυτή εμφανίζεται μειωμένη κατά 7 φορές, ενώ τα πράγματα δείχνουν ακόμη χειρότερα όταν οι διαστάσεις ξεπερνούν τις 20 και τις 30. Η εκτίμηση των συγγραφέων είναι ότι η διαφοροποίηση μεταξύ κοντινότερων και μακρινότερων γειτόνων γίνεται δυσδιάκριτη μετά από περίπου 15 διαστάσεις. Επίσης, ένα ενδιαφέρον στοιχείο που απορρέει από τα γραφήματα αυτά, είναι ότι η μεταβολή του πλήθους των εγγραφών και της κατανομής των δεδομένων είχε μικρή επίδραση στη συνολική εικόνα. Συγκεκριμένα μονάχα η εκθετική κατανομή πέτυχε μεγαλύτερη διαφοροποίηση από τις υπόλοιπες. Ωστόσο, και σε αυτή την περίπτωση η διαφοροποίηση μειώνεται τόσο δραστικά που η απόλυτη διαφορά γίνεται γρήγορα αμελητέα.

Οι Beyer et. al. (1999) κατέληξαν στο συμπέρασμα ότι η ακαταλληλότητα των μετρικών απόστασης σε υψηλές διαστάσεις δεν γενικεύεται σε όλες τις περιπτώσεις, αλλά εξαρτάται από τη φύση των δεδομένων και κάτω από συγκεκριμένες συνθήκες μπορεί να αποδειχθούν χρήσιμα. Συγκεκριμένα έχουν νόημα εφαρμογές για πολυδιάστατη αναζήτηση *NN* σε δύο περιπτώσεις:

- Ταξινόμηση και Σχετική Ομοιότητα

Η απόλυτη και η σχετική ομοιότητα δεδομένων μπορεί να εξυπηρετηθεί αποδοτικά από πολυδιάστατο *NN*. Για παράδειγμα, εάν υπάρχει συσχέτιση μεταξύ του σημείου ερωτήματος και των σημείων δεδομένων έτσι ώστε κάποιο από αυτά να συμπίπτει με το σημείο ερωτήματος, τότε *DMIN* = 0. Υποθέτοντας ότι τα δεδομένα δεν είναι πανομοιότυπα μπορεί να προκύψει μια λογική απάντηση. Επιπλέον, εάν το πρόβλημα διατυπώνεται έτσι ώστε να επιτρέπει στο σημείο ερωτήματος να βρίσκεται σε μια μικρή απόσταση από το σημείο δεδομένων (αντί να είναι πανομοιότυπο), η αναζήτηση επιστρέφει και πάλι μια ικανοποιητική λύση. Πρέπει ωστόσο να σημειωθεί ότι η διατήρηση αυτής της μικρής απόστασης με την αύξηση των διαστάσεων είναι πολύ δύσκολη, αφού προστίθενται συνεχώς νέοι παράγοντες στο μετρικό της απόστασης. Συνεπώς για τέτοιου είδους προβλήματα, το σημείο ερωτήματος πρέπει να έρχεται όλο και πιο κοντά στο σημείο δεδομένων όσο αυξάνονται οι διαστάσεις.

Η κατάσταση αυτή γενικεύεται εάν τα δεδομένα αποτελούνται από μια ομάδα τυχαία επιλεγμένων σημείων, μαζί με σημεία που είναι κατανεμημένα σε *clusters* κάποιας ακτίνας δ, γύρω από ένα ή περισσότερα σημεία και το σημείο ερωτήματος συμπίπτει με κάποιο από αυτά τα *clusters* (Εικόνα 33). Το παραπάνω αποτελεί πρόβλημα κατηγοριοποίησης, όπου τα δεδομένα από τη φύση τους διαχωρίζονται σε διακριτές ομάδες ή *clusters* για κάποιο πολυδιάστατο χώρο χαρακτηριστικών.



Εικόνα 33: Ερώτημα *NN* σε *clustered* δεδομένα. (Beyer et. al. (1999))

Στην περίπτωση αυτή, υπάρχει ένα *cluster* (αυτό που περιλαμβάνει το σημείο ερωτήματος) το οποίο βρίσκεται κοντινότερα στο σημείο σε σχέση με τα υπόλοιπα που τοποθετούνται σε δυσδιάκριτες αποστάσεις. Η κατάλληλη απόκριση σε ένα τέτοιο ερώτημα είναι η επιστροφή όλων των σημείων του κοντινότερου *cluster* και όχι μόνο το κοντινότερο σημείο, το οποίο γρήγορα χάνει τη διακριτικότητά του σε σχέση με τα υπόλοιπα σημεία του *cluster* καθώς αυξάνεται η διάσταση. Ωστόσο, εάν δεν υπάρχει η δυνατότητα το σημείο ερωτήματος να βρίσκεται μέσα σε κάποια όρια απόστασης, η παραπάνω προσέγγιση δεν είναι αποδοτική.

- Έμμεση Χαμηλή Διαστατικότητα

Όταν η "χρήσιμη" διάσταση των δεδομένων είναι πολύ χαμηλότερη από την πραγματική, διάφορες Μέθοδοι Μείωσης Διαστατικότητας *(Dimensionality Reduction Methods)* επιτρέπουν στα πολυδιάστατα ερωτήματα *NN* να έχουν νόημα. Σε διάφορες έρευνες (Belussi και Faloutsos (1995); Faloutsos και Gaede (1996); Faloutsos και Kamel (1994)) έχουν αναπτυχθεί μέθοδοι για τον εντοπισμό των χρήσιμων διαστάσεων τέτοιων δεδομένων, όπως για παράδειγμα με χρήση *Principal Components Analysis, Factor Analysis* και *Random Projections*. Ωστόσο, αυτές οι τεχνικές έχουν αξία μονάχα εάν η αναζήτηση *NN* έχει νόημα στις νέες διαστάσεις.

ΚΕΦΑΛΑΙΟ 2: Μεθοδολογία προσέγγισης

Στη σχετική βιβλιογραφία, υπάρχουν πολλές εργασίες που μελετούν την αποδοτικότητα των δομών ευρετηριοποίησης χωρικών δεδομένων. Η αποτελεσματικότητα εξετάζεται με διάφορες παραμέτρους, ανάλογα με τον αντικειμενικό σκοπό της έρευνας. Οι βασικότερες αφορούν την αποτελεσματικότητα της διαδικασίας κατασκευής της δομής και την απόδοσή της κατά την εκτέλεση ερωτημάτων σε αυτή. Για τη μελέτη της αποδοτικότητας κατά την κατασκευή της δομής υιοθετούνται ως μετρικά:

- η χρήση του αποθηκευτικού χώρου *(storage utilization)*
- το μέγεθος του τελικού αρχείου *(index size)*
- ο χρόνος κατασκευής του ευρετηρίου *(construction time)*
- η χρήση της μνήμης *(disk I/Os)*

Για τη μελέτη της αποδοτικότητας κατά την εκτέλεση ερωτημάτων στα δεδομένα, εξετάζεται η χρήση της μνήμης σε όρους *disk I/Os*, για ερωτήματα σημείου, εύρους, χωρικής σύζευξης, τομής και κοντινότερου γείτονα

Η παρούσα μελέτη στοχεύει στη συγκριτική μελέτη της απόδοσης της δομής του *R*-tree* (Beckmann et. al. (1990)), για τις περιπτώσεις που η κατασκευή του πραγματοποιείται με σειριακή και μαζική εισαγωγή των δεδομένων. Στη δεύτερη περίπτωση γίνεται προ-επεξεργασία για την ομαδοποίησή των δεδομένων με τον αλγόριθμο *STR* (Leutenegger, Lopez και Edgington (1997)). Στα πλαίσια της μελέτης, εξετάζονται δύο βασικές παράμετροι:

- η απόδοση της δομής κατά τη διαδικασία κατασκευής
- η απόδοση της δομής κατά την εκτέλεση ερωτημάτων στα δεδομένα.

Τα μετρικά που λαμβάνονται υπόψη για την αξιολόγηση της αποδοτικότητας ως προς τη διαδικασία κατασκευής είναι:

- η χρήση του αποθηκευτικού χώρου *(storage utilization),*
- το μέγεθος του τελικού αρχείου *(index size)* και
- η χρήση της μνήμης *(disk I/Os).*

Οι παράμετροι αυτές, υπολογίζονται και αναλύονται για κάθε μια από τις περιπτώσεις που καθορίζονται στο πλαίσιο της πειραματικής διαδικασίας. Στόχος είναι μια όσο το δυνατόν πιο πολύπλευρη ανάλυση των χαρακτηριστικών της δομής, ώστε να οδηγηθούμε σε ασφαλή συμπεράσματα.

Η αποτελεσματικότητα ως προς την διεκπεραίωση ερωτημάτων στα δεδομένα μετριέται με τον υπολογισμό των παραμέτρων:

- Αριθμός προσπελάσεων σελίδων μνήμης *(# Disk Accesses)*
- Αριθμός προσπελάσεων σελίδων ευρετηρίου *(# Index Accesses)*
- Αριθμός προσπελάσεων σελίδων δεδομένων *(# Leaf Accesses)*
- Ποσοστό χρήσης-επισκεψιμότητας ευρετηρίου *(% Index Level Usage)*
- Ποσοστό χρήσης-επισκεψιμότητας δεδομένων *(% Leaf Level Usage)*

Εκτός λοιπόν από το πλήθος των προσπελάσεων σελίδων της μνήμης *(disk I/Os)* κατά την εκτέλεση του ερωτήματος, εξετάζονται κάποια μετρικά που περιγράφουν λεπτομερέστερα τον τρόπο με τον οποίο χρησιμοποιείται η δομή. Πιο συγκεκριμένα, τα αποτελέσματα αυτών των παραμέτρων χρησιμοποιούνται ώστε να εξεταστεί πιο σφαιρικά η απόδοση της δομής, με βάση το ποσοστό των δεδομένων που χρειάζεται να επισκεφτεί κατά την εκτέλεση ενός ερωτήματος. Για τον υπολογισμό του παραπάνω, είναι απαραίτητες οι μετρήσεις του πλήθους των προσπελάσεων των σελίδων μνήμης που αντιστοιχίζονται στις εγγραφές της ΒΔ. Όπως θα περιγραφεί στη συνέχεια, τα δεδομένα αποθηκεύονται στο επίπεδο των φύλλων της δομής. Εξετάζοντας λοιπόν την απόδοση της δομής με όρους *Index* και *Leaf Level Access*, έχουμε τη δυνατότητα να μελετήσουμε αυτή την παράμετρο αποτελεσματικά.

Οι σχέσεις που διέπουν τα μετρικά αυτά είναι οι ακόλουθες:

(# Disk Accesses) = (# Index Accesses) + (# Leaf Accesses)

(% Index Level Usage) = (# Index Accesses) / (Index Level Size)

(% Leaf Level Usage) = (# Leaf Accesses) / (Leaf Level Size)

Στοχεύοντας σε μια πολύπλευρη και αναλυτική μελέτη της αποδοτικότητας της δομής στις δύο περιπτώσεις, καθιερώνονται στο πλαίσιο των πειραματικών μετρήσεων οι παράμετροι και τα δεδομένα που περιγράφονται στο Κεφάλαιο 3.

ΚΕΦΑΛΑΙΟ 3: Ανάλυση Δεδομένων και Ερμηνεία Ευρημάτων

Στο παρόν κεφάλαιο περιγράφεται η πειραματική διαδικασία και τα αποτελέσματα που προέκυψαν. Αρχικά παρουσιάζονται οι παράμετροι και τα δεδομένα των μετρήσεων, στη συνέχεια παρατίθενται τα αποτελέσματα των δοκιμών και τέλος δίνεται η ερμηνεία των ευρημάτων της πειραματικής μελέτης. Το υπολογιστικό σύστημα που χρησιμοποιήθηκε στα πειράματα περιλαμβάνει τα παρακάτω χαρακτηριστικά:

- Επεξεργαστής Intel Core Duo T5300 1.76 GHz
- Μνήμη RAM 2GB
- Λειτουργικό Σύστημα Windows Vista 32 bit

Η γλώσσα προγραμματισμού είναι η C++ και ο κώδικας που επιλέχθηκε για τις μετρήσεις μας αποτελεί προϊόν εργασίας του κ. Μάριου Χατζηελευθερίου. Η χρήση του κώδικα έγινε με τη σύμφωνη γνώμη του και ο κώδικας είναι διαθέσιμος στην ιστοσελίδα: https://github.com/libspatialindex/libspatialindex.

3.1. Παράμετροι Κατασκευής του R* tree

Στην ενότητα αυτή, αρχικά παραθέτουμε τις παραμέτρους που συμβάλλουν στον τρόπο με τον οποίο θα κατασκευαστεί το *R*-tree*. Για κάθε μια από αυτές περιγράφεται η σημασία της και ο τρόπος επιλογής της κατάλληλης τιμής. Οι παράμετροι είναι οι ακόλουθες:

- *Fan-Out*
- *Fill Factor*
- *Near Minimum Overlap Enlargement Factor*
- *Split Distribution Factor*
- *Reinsert Factor*

Fan-Out

Το *Fan-Out* αναφέρεται στο πλήθος των στοιχείων που περιλαμβάνουν οι κόμβοι και τα φύλλα του δένδρου. Στη σχετική βιβλιογραφία, η επιλογή της τιμής γίνεται έτσι ώστε να διευκολύνεται η πειραματική διαδικασία και η ερμηνεία των αποτελεσμάτων με βάση την παρακάτω λογική. Όπως προαναφέρθηκε, τα μετρικά που αφορούν την αποδοτικότητα μιας δομής κατά την εκτέλεση ενός ερωτήματος είναι ο χρόνος απόκρισης και ο αριθμός επισκέψεων στις σελίδες μνήμης του Η/Υ. Όσον αφορά το δεύτερο, προκειμένου να εξασφαλιστεί ένας εύκολος τρόπος καταμέτρησης η χωρητικότητα των κόμβων και των φύλλων του δένδρου επιλέγεται έτσι ώστε κάθε κόμβος ή φύλλο να καταλαμβάνει μια σελίδα μνήμης. Έτσι λοιπόν, το πλήθος των στοιχείων σε κάθε κόμβο-φύλλο του δένδρου προκύπτει από την παρακάτω σχέση:

Χωρητικότητα Κόμβου-Φύλλου = Μέγεθος Σελίδας Μνήμης / Μέγεθος Στοιχείου

Το μέγεθος της σελίδας μνήμης στο υπολογιστικό σύστημα που χρησιμοποιήθηκε είναι ίσο με 4096 bytes. Το μέγεθος του στοιχείου αφορά το μέγεθος των αντικειμένων που θα αποθηκευτούν στη δομή και εξαρτάται από τις διαστάσεις του χώρου δεδομένων. Για τον υπολογισμό του, παραθέτουμε στη συνέχεια ένα τμήμα του αρχείου δεδομένων, στην απλή περίπτωση των δύο διαστάσεων (Εικόνα 34). Κάθε γραμμή του αρχείου αναπαριστά τον δείκτη του αντικειμένου και τις τιμές (x_1, y_1) και (x_2, y_2) του κάτω αριστερά και του άνω δεξιά σημείου του.

		…		
20	0.667846	0.195083	0.757537	0.234225
21	0.952584	0.459529	0.961698	0.546906
22	0.502589	0.881762	0.527033	0.911598
23	0.750989	0.736782	0.825303	0.797356
		…		

Εικόνα 34: Η μορφή καταγραφής του *MBR* κάθε αντικειμένου στο αρχείο δεδομένων, για την περίπτωση του χώρου δύο διαστάσεων.

Οι τιμές αυτές αποθηκεύονται με τη βοήθεια μεταβλητών τύπου double και συνεπώς ο χώρος που απαιτείται για την αποθήκευση ενός αντικειμένου είναι 40 *bytes*. Χρησιμοποιώντας την παραπάνω σχέση, έχουμε:

Χωρητικότητα Κόμβου-Φύλλου = 4096 / 40 = 102.4

Η τιμή αυτή στρογγυλοποιείται προς τα κάτω και σε αντίστοιχες εργασίες που μελετήθηκαν από τη βιβλιογραφία επιλέγεται χωρητικότητα κόμβων-φύλλων ίση με 100. Με ανάλογη διαδικασία, υπολογίζεται η χωρητικότητα των κόμβων και των φύλλων του δένδρου για κάθε έναν από τους χώρους υψηλότερης διάστασης που μελετήθηκαν. Συγκεκριμένα:

- 4 διαστάσεις: $4096 / (8 + 8 \cdot 8) = 4096 / 72 \approx 56$
- 6 διαστάσεις: $4096 / (8 + 12 \cdot 8) = 4096 / 104 \approx 39$
- 8 διαστάσεις: $4096 / (8 + 16 \cdot 8) = 4096 / 136 \approx 30$
- 10 διαστάσεις: $4096 / (8 + 20 \cdot 8) = 4096 / 168 \approx 24$
- 12 διαστάσεις: $4096 / (8 + 24 \cdot 8) = 4096 / 200 \approx 20$
- 14 διαστάσεις: $4096 / (8 + 28 \cdot 8) = 4096 / 232 \approx 17$
- 16 διαστάσεις: $4096 / (8 + 32 \cdot 8) = 4096 / 264 \approx 15$

Fill Factor

Το *Fill Factor* συνιστά το ποσοστό πληρότητας των κόμβων ή φύλλων της δομής και συνεπώς του *Fan-Out*. Η τιμή του και κυμαίνεται από 1% έως 100%. Μικρή τιμή του *Fill Factor* οδηγεί σε μεγαλύτερο ευρετήριο, το οποίο όμως μπορεί ευκολότερα να υποστηρίξει την εισαγωγή νέων εγγραφών. Αντίθετα, μεγάλη τιμή του *Fill Factor* εξασφαλίζει πιο συμπαγές ευρετήριο, το οποίο ωστόσο υστερεί στην εισαγωγή νέων εγγραφών λόγω των περισσοτέρων διασπάσεων που προκαλούνται κατά τη ρουτίνα εισαγωγής. Συνήθως, ως "χρυσή τομή" επιλέγεται τιμή 70%. Ωστόσο, για στατικά δεδομένα η βέλτιστη επιλογή για την τιμή της παραμέτρου είναι το 100% και οδηγεί σε ελαχιστοποίηση του μεγέθους του ευρετηρίου.

Στα πλαίσια της πειραματικής διαδικασίας, τα δεδομένα θεωρούνται αμετάβλητα ώστε να έχει νόημα η χρήση μεθόδων μαζικής εισαγωγής στη δομή. Έτσι η τιμή που επιλέγεται για το *Fill Factor* είναι 0.99 και αποτελεί το ανώτατο όριο με βάση τον κώδικα που χρησιμοποιήθηκε.

Near Minimum Overlap Enlargement Factor

Η παράμετρος του *Near Minimum Overlap Enlargement Factor* ορίστηκε από τους Beckmann et. al. (1990) στον αλγόριθμο επιλογής του κατάλληλου υπο-δένδρου (*ChooseSubtree*, Παράρτημα Β). Όπως περιγράφηκε στην ενότητα 1.1.3.2.1. οι συγγραφείς προκειμένου να μειώσουν το υπολογιστικό κόστος για τη μέτρηση των επικάλυψης μεταξύ όλων των εγγραφών ενός κόμβου, θεώρησαν ότι μπορεί να γίνει κατάλληλη επιλογή ενός αριθμού *p* από αυτές χωρίς να οδηγηθούν σε μείωση της αποτελεσματικότητας (Παράρτημα Β). Μετά από μια σειρά πειραμάτων κατέληξαν ότι για τιμή *p=32* η μείωση κατά την απόδοση είναι μηδαμινή. Η τιμή αφορά το χώρο δεδομένων δύο διαστάσεων, ενώ μετρήσεις για χώρους υψηλότερων διαστάσεων δεν έχουν υλοποιηθεί. Στα πειράματά μας, υιοθετήσαμε την ίδια τιμή για τις περιπτώσεις των 2, 4 και 6 διαστάσεων. Στις υπόλοιπες διαστάσεις εξετάζονται όλες οι εγγραφές του κόμβου, αφού ο αριθμός τους είναι μικρότερος της τιμής του *p*.

Split Distribution Factor

Το *Split Distribution Factor* δεν είναι τίποτε άλλο παρά η τιμή της μεταβλητής *m* του ελάχιστου αριθμού εγγραφών ανά κόμβο ή φύλλο του δένδρου, ως ποσοστό του μέγιστου αριθμού εγγραφών ανά κόμβο ή φύλλο *M*. Οι Beckmann et. al. (1990) στα πλαίσια της εργασίας τους εξέτασαν διάφορες τιμές της παραμέτρου και κατέληξαν ότι τα καλύτερα αποτελέσματα επιτυγχάνονται για τιμή *m = 40%*. Αυτή είναι η τιμή που θεωρούμε στην πειραματική διαδικασία μας.

Reinsert Factor

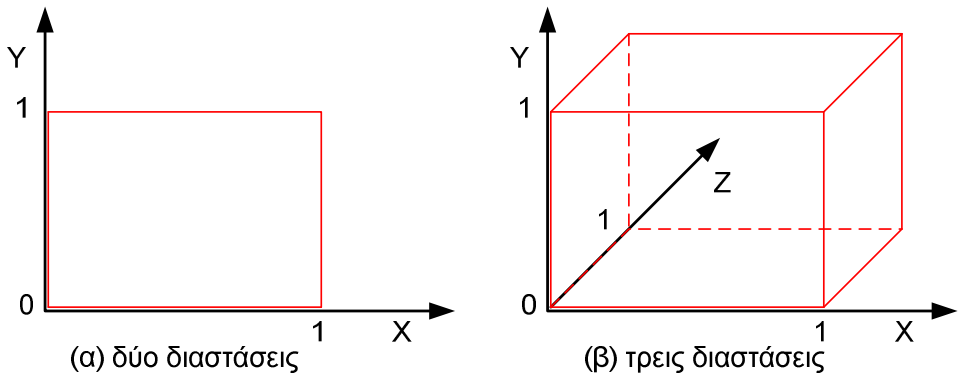
Το *Reinsert Factor* ορίζεται από τους Beckmann et. al. (1990) κατά την περιγραφή της εξαναγκασμένης επαν-εισαγωγής εγγραφών. Στην εργασία τους, η παράμετρος που συμβολίζεται με *p* καθορίζει το πλήθος των εγγραφών ενός κόμβου, ως ποσοστό του μέγιστου αριθμού *M*, οι οποίες πρέπει να επανατοποθετηθούν κατά την εισαγωγή μιας νέας εγγραφής. Τα πειράματα των συγγραφέων έδειξαν ότι για τιμή *p = 30%,* τόσο για τους κόμβους όσο και για τα φύλλα, επιτυγχάνεται η καλύτερη απόδοση κι έτσι στις μετρήσεις μας υιοθετείται αυτή η τιμή.

3.2. Δεδομένα Πειραματικής Διαδικασίας

Στην παρακάτω ενότητα, αναλύονται τα πειραματικά δεδομένα που συνθέτουν το πλαίσιο των μετρήσεων και της διερεύνησης που εκπονήθηκε. Τα δεδομένα αυτά διαμορφώνονται από την επιλογή (α) του χώρου δεδομένων, (β) του τύπου δεδομένων, (γ) του τύπου ερωτήματος και (δ) της μεθόδου εισαγωγής.

Χώρος Δεδομένων

Στην εικόνα που ακολουθεί απεικονίζεται ο χώρος δεδομένων, για δύο και τρεις διαστάσεις. Οι τιμές σε κάθε άξονα, βρίσκονται στο διάστημα [0,1]. Επεκτείνοντας τα παραπάνω, δημιουργείται ο χώρος δεδομένων για υψηλότερες διαστάσεις. Στα πειράματα θεωρούμε χώρο δεδομένων από 2 έως 16 διαστάσεις, λαμβάνοντας υπόψη τα πορίσματα της ανάλυσης των Beyer et. al. (1999).



Εικόνα 35: Ο χώρος δεδομένων στις (α) δύο διαστάσεις και (β) τρεις διαστάσεις.

Δεδομένα Εισαγωγής

Το πλήθος των δεδομένων είναι 100.000 και ο τύπος του είναι:

- σημεία,
- ορθογώνια, όπου το μήκος των πλευρών τους είναι σταθερό και ίσο με ένα ποσοστό του μήκους του άξονα συντεταγμένων (θεωρούμε μήκος ίσο με το 0.1% και το 1% του μήκους του άξονα) και
- ορθογώνια, όπου το μήκος των πλευρών τους ακολουθεί την ομοιόμορφη κατανομή στο διάστημα [0 , 1].

Ερωτήματα

Για τη μελέτη της αποδοτικότητας του *R*-tree*, επιλέχθηκαν μετρήσεις για ερωτήματα Κοντινότερου Γείτονα στην περίπτωση που αναζητείται:

- ο κοντινότερος γείτονας (1NN)
- οι 10 κοντινότεροι γείτονες (10NN)
- οι 100 κοντινότεροι γείτονες (100NN)

Το πλήθος των ερωτημάτων που εκτελούνται σε κάθε περίπτωση είναι ίσο με 100 και η τελική τιμή για καθένα από τα μετρικά, προκύπτει από τον αριθμητικό μέσο όρο των τιμών του στις 100 μετρήσεις.

Εισαγωγή Δεδομένων

Στο κεφάλαιο 1 περιγράφηκε ο τρόπος κατασκευής του *R*-tree*, όταν η εισαγωγή των δεδομένων γίνεται σειριακά και μαζικά. Στην πρώτη περίπτωση, η ανάπτυξη του δένδρου γίνεται κάθε φορά που εισάγεται ένα αντικείμενο στα φύλλα του. Στη δεύτερη περίπτωση τα δεδομένα υπόκεινται σε επεξεργασία προκειμένου να ομαδοποιηθούν πριν την εισαγωγή τους στο δένδρο, βελτιώνοντάς με αυτό τον τρόπο τη δομή του. Με δεδομένη τη θεώρηση ότι τα δεδομένα μας είναι στατικά, συγκρίνουμε την αποδοτικότητα της δομής στις δύο περιπτώσεις.

3.3. Αποτελέσματα

Η ανάλυση των πειραματικών μετρήσεων χωρίζεται σε δύο τμήματα. Στο πρώτο μέρος, στόχος είναι η "οριοθέτηση" της διαστατικότητας του χώρου δεδομένων. Με άλλα λόγια αναζητείται ένα ανώτατο κατώφλι για το πλήθος των διαστάσεων του χώρου των δεδομένων, άνω του οποίου η αποδοτικότητα της δομής κρίνεται χαμηλή. Η τιμή αυτή θα καθορίσει το εύρος των διαστάσεων που μελετώνται στη συνέχεια. Στο δεύτερο τμήμα γίνεται η ανάλυση της λειτουργίας της δομής και η σύγκριση μεταξύ των δύο μεθόδων εισαγωγής των δεδομένων σε αυτή. Στα πλαίσια της σύγκρισης, μελετώνται οι:

- Απόδοση κατά την κατασκευή της δομής.
- Απόδοση κατά την εκτέλεση ερωτημάτων.

Στους πίνακες και τα γραφήματα που ακολουθούν, ισχύει ο συμβολισμός:

Τύπος δεδομένων

- Case 1: Σημειακά χωρικά αντικείμενα
- Case 2: Ορθογώνια με σταθερό μήκος πλευράς, το 0.1% του μήκους του άξονα σε κάθε διάσταση.
- Case 3: Ορθογώνια με σταθερό μήκος πλευράς, το 1% του μήκους του άξονα σε κάθε διάσταση.
- Case 4: Ορθογώνια με τυχαίο μήκος πλευράς, που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.

Τρόπος εισαγωγής

- Serial Load: Σειριακή εισαγωγή δεδομένων στη δομή ένα προς ένα.
- Bulk Load: Μαζική εισαγωγή δεδομένων στη δομή κατόπιν ομαδοποίησης.

ΜΕΡΟΣ 1: Κατώφλι Διάστασης Χώρου Δεδομένων

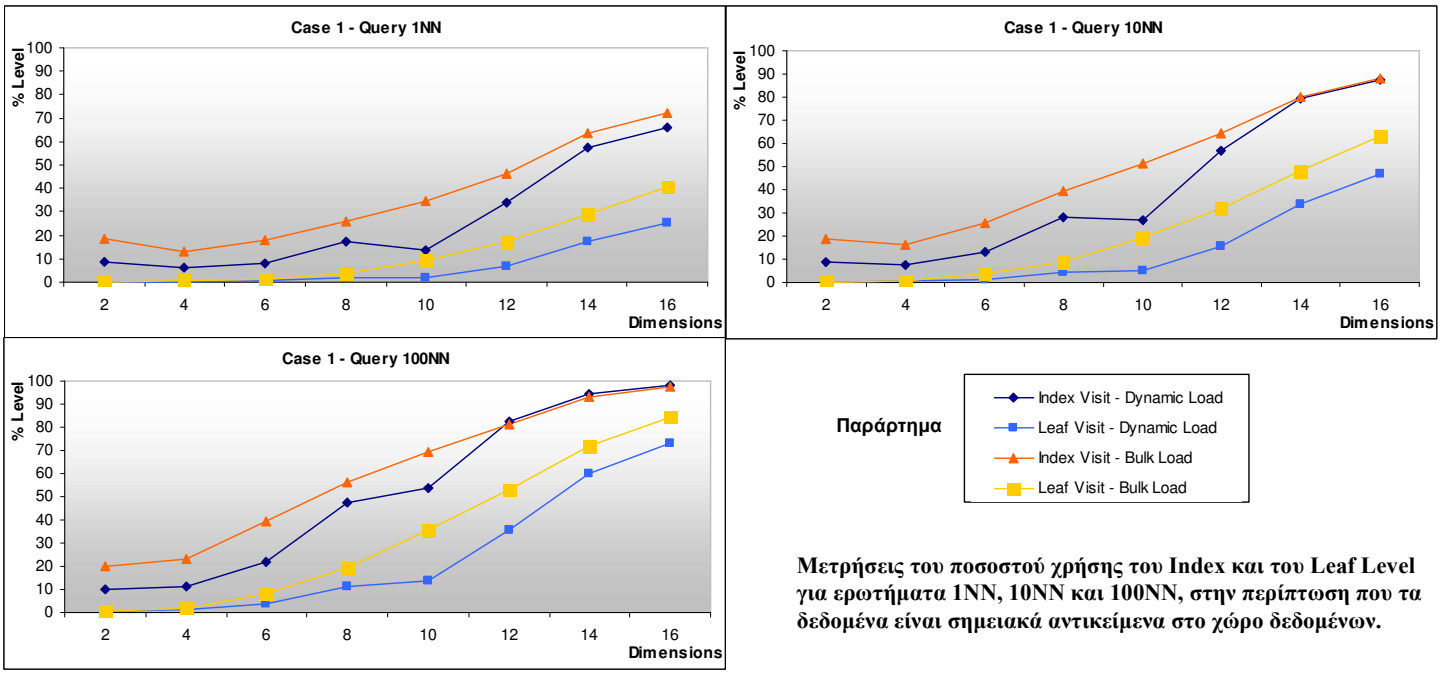
Το *R*-tree* αποτελεί ένα διατεταγμένο ευρετήριο όπου τα ανώτερα επίπεδα της δομής *(Index Level)* οδηγούν στα φύλλα *(Leaf Level),* τα οποία αποθηκεύουν την πληροφορία των χωρικών δεδομένων. Είναι προφανές λοιπόν, ότι ο αριθμός των φύλλων αποτελεί το μεγαλύτερο τμήμα της δομής, γεγονός που επαληθεύεται και από τις πειραματικές μετρήσεις που συνοψίζονται στον πίνακα 1. Ο πίνακας περιλαμβάνει το πλήθος των κόμβων στο *Index* και στο *Leaf Level* και υπολογίζει το ποσοστό του πρώτου ως προς το σύνολο της δομής, για κάθε διάσταση. Με βάση τις μετρήσεις το 95%, κατά μέσο όρο, των κόμβων που κατασκευάζονται βρίσκονται στο επίπεδο των φύλλων και μονάχα το 5% αποτελεί τα ανώτερα επίπεδα του δένδρου έως τη ρίζα.

Συμπερασματικά, ως κριτήριο για τη μέτρηση της απόδοσης του *R*-tree* στις διαστάσεις που μελετώνται, θεωρείται το ποσοστό χρήσης του *Leaf Level* κατά την εκτέλεση των ερωτημάτων. Το κατώτατο κατώφλι αποδοτικής λειτουργίας για τη δομή καθορίζεται στο 20%.

Πίνακας 1: Ποσοτική σχέση μεταξύ των φύλλων και των εσωτερικών κόμβων στο στη δομή του *R*-tree* για κάθε διάσταση.

	Serial Load				Bulk Load			
	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 1**	**Case 2**	**Case 3**	**Case 4**
2-Dimensions								
Index Nodes	23	21	22	20	12	12	12	12
Leaf Nodes	1456	1448	1447	1384	1011	1011	1011	1011
Index Nodes Part	1.56%	1.43%	1.50%	1.42%	1.17%	1.17%	1.17%	1.17%
4-Dimensions								
Index Nodes	71	70	68	73	35	35	35	35
Leaf Nodes	2535	2557	2559	2507	1819	1819	1819	1819
Index Nodes Part	2.72%	2.66%	2.59%	2.83%	1.89%	1.89%	1.89%	1.89%
6-Dimensions								
Index Nodes	144	141	148	141	73	73	73	73
Leaf Nodes	3578	3636	3661	3553	2632	2632	2632	2632
Index Nodes Part	3.87%	3.73%	3.89%	3.82%	2.70%	2.70%	2.70%	2.70%
8-Dimensions								
Index Nodes	243	242	246	230	125	125	125	125
Leaf Nodes	4672	4713	4780	4532	3449	3449	3449	3449
Index Nodes Part	4.94%	4.88%	4.89%	4.83%	3.50%	3.50%	3.50%	3.50%
10-Dimensions								
Index Nodes	373	372	381	367	200	200	200	200
Leaf Nodes	5849	5714	5709	5584	4348	4348	4348	4348
Index Nodes Part	5.99%	6.11%	6.26%	6.17%	4.40%	4.40%	4.40%	4.40%
12-Dimensions								
Index Nodes	542	533	541	541	294	294	294	294
Leaf Nodes	6829	6696	6650	6649	5264	5264	5264	5264
Index Nodes Part	7.35%	7.37%	7.52%	7.52%	5.29%	5.29%	5.29%	5.29%
14-Dimensions								
Index Nodes	766	760	760	755	419	419	419	419
Leaf Nodes	7826	7755	7794	7619	6250	6250	6250	6250
Index Nodes Part	8.92%	8.93%	8.88%	9.02%	6.28%	6.28%	6.28%	6.28%
16-Dimensions								
Index Nodes	995	998	995	995	552	552	552	552
Leaf Nodes	8782	8845	8725	8465	7143	7143	7143	7143
Index Nodes Part	10.18%	10.14%	10.24%	10.52%	7.17%	7.17%	7.17%	7.17%

Στις τέσσερις ομάδες γραφημάτων και τους αντίστοιχους πίνακες που ακολουθούν, απεικονίζονται τα αποτελέσματα των μετρήσεων του ποσοστού χρήσης του *Index* και του *Leaf Level,* στις περιπτώσεις της σειριακής και της μαζικής εισαγωγής δεδομένων. Κάθε ομάδα αναφέρεται σε έναν τύπο δεδομένων για όλα τα ερωτήματα και τις διαστάσεις που εξετάζονται. Στους πίνακες, με κόκκινο χρώμα σημειώνονται οι ανώτατες τιμές που βρίσκονται κάτω από το κατώφλι που θέσαμε.

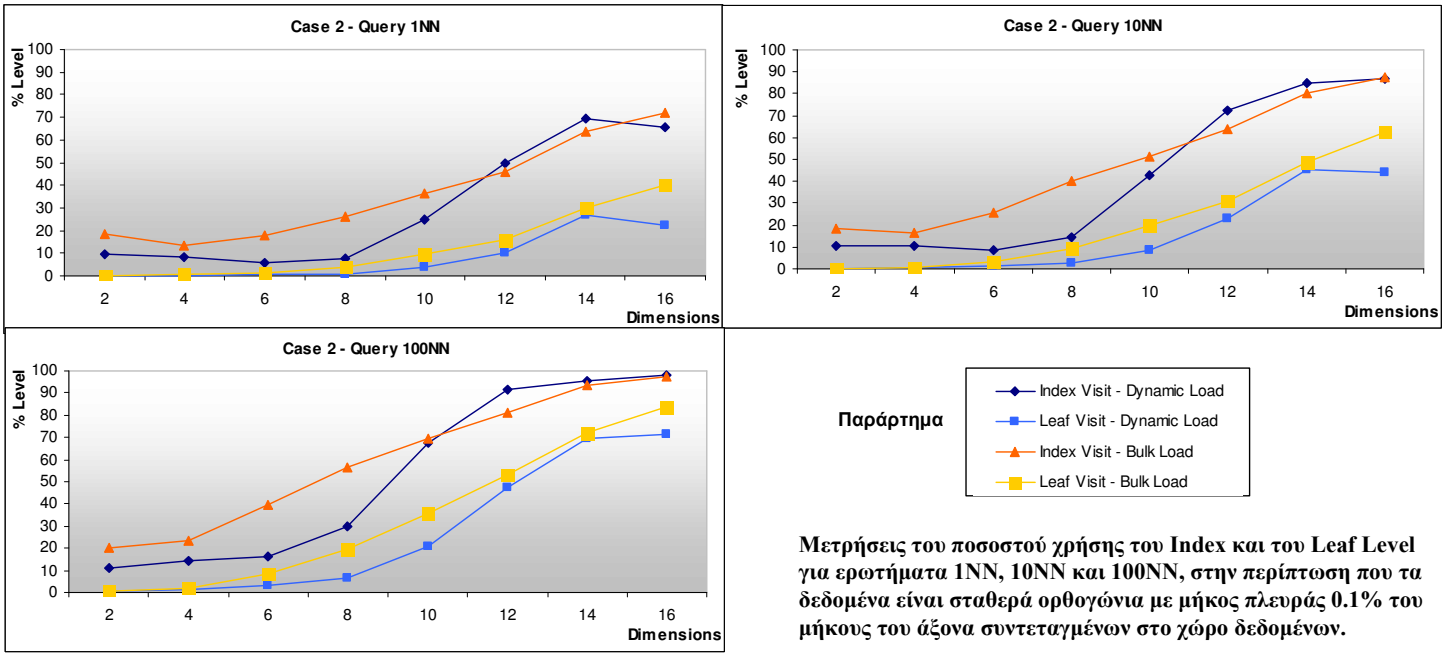


Μετρήσεις του ποσοστού χρήσης του Index και του Leaf Level για ερωτήματα 1NN, 10NN και 100NN, στην περίπτωση που τα δεδομένα είναι σημειακά αντικείμενα στο χώρο δεδομένων.

Εικόνα 36: Τιμές *Index* και *Leaf Level Usage* για τον πρώτο τύπο δεδομένων.

Πίνακας 2: Τιμές *Index* και *Leaf Level Usage* για την πρώτη περίπτωση δεδομένων σε ερωτήματα *1NN, 10NN, 100NN* και τις εξεταζόμενες διαστάσεις.

Case 1									
	1NN								
	Dimension	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Serial	Index Level Usage	8.91%	6.08%	7.83%	17.13%	13.73%	34.22%	57.66%	66.15%
	Leaf Level Usage	0.09%	0.15%	0.51%	1.79%	1.86%	6.9%	**17.30%**	25.61%
Bulk	Index Level Usage	18.25%	12.97%	17.86%	25.98%	34.84%	46.58%	63.32%	72.38%
	Leaf Level Usage	0.11%	0.31%	1.41%	3.91%	9.33%	**17.12%**	28.73%	40.99%
	10NN								
Serial	Index Level Usage	9.04%	7.77%	12.94%	28.14%	26.82%	56.64%	79.19%	87.35%
	Leaf Level Usage	0.13%	0.38%	1.37%	4.39%	4.99%	**15.77%**	34.00%	46.96%
Bulk	Index Level Usage	18.58%	16.37%	25.82%	39.52%	51.51%	64.35%	80.17%	87.94%
	Leaf Level Usage	0.18%	0.74%	3.48%	9.05%	**19.39%**	31.71%	48.22%	63.27%
	100NN								
Serial	Index Level Usage	9.83%	11.39%	22.12%	47.64%	54.04%	82.8%	94.42%	98.05%
	Leaf Level Usage	0.34%	1.13%	3.83%	11.08%	**13.85%**	35.38%	60.29%	73.3%
Bulk	Index Level Usage	20.08%	23.4%	39.48%	56.34%	69.49%	81.14%	93.40%	97.21%
	Leaf Level Usage	0.41%	2.03%	8.12%	**19.28%**	35.75%	53.31%	71.98%	84.24%

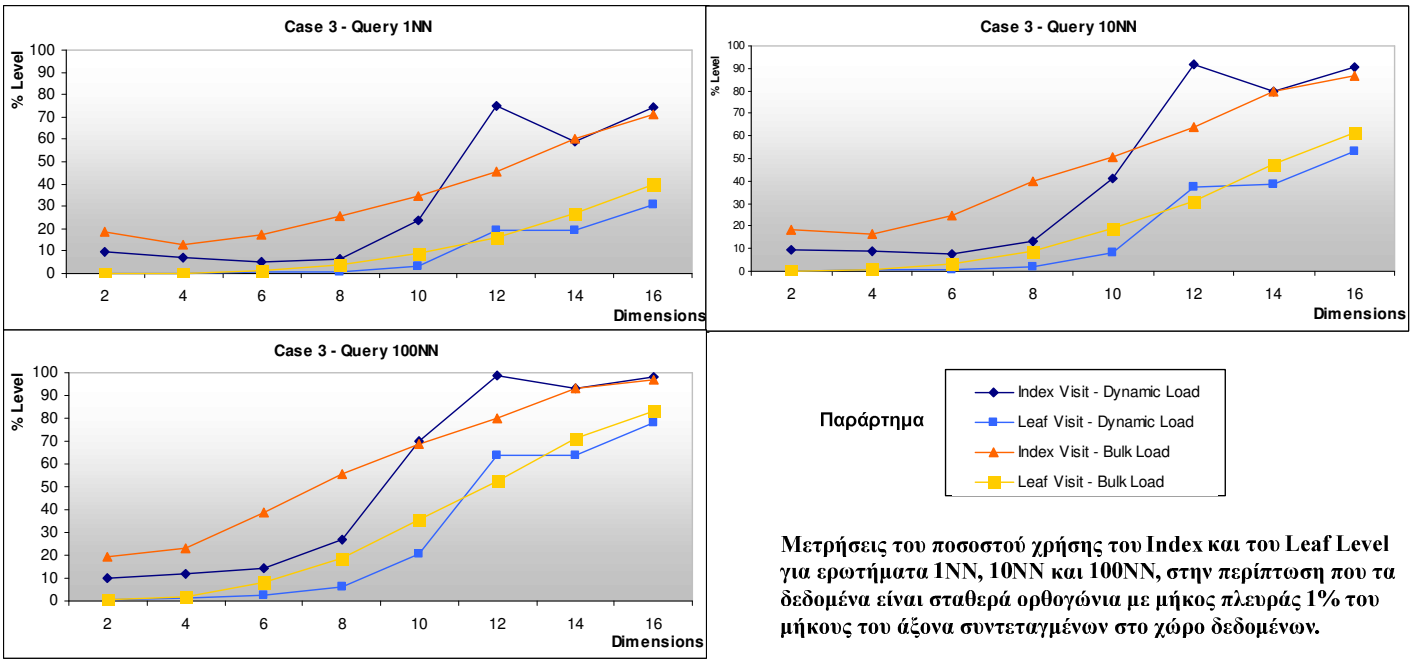


Μετρήσεις του ποσοστού χρήσης του Index και του Leaf Level για ερωτήματα 1NN, 10NN και 100NN, στην περίπτωση που τα δεδομένα είναι σταθερά ορθογώνια με μήκος πλευράς 0.1% του μήκους του άξονα συντεταγμένων στο χώρο δεδομένων.

Εικόνα 37: Τιμές *Index* και *Leaf Level Usage* για το δεύτερο τύπο δεδομένων.

Πίνακας 3: Τιμές *Index* και *Leaf Level Usage* για τη δεύτερη περίπτωση δεδομένων σε ερωτήματα *1NN, 10NN, 100NN* και τις εξεταζόμενες διαστάσεις.

Case 2									
	1NN								
	Dimension	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Serial	Index Level Usage	9.86%	8.21%	5.99%	7.83%	24.70%	49.38%	69.19%	65.30%
	Leaf Level Usage	0.08%	0.23%	0.43%	0.89%	3.54%	**10.35%**	26.92%	22.45%
Bulk	Index Level Usage	18.25%	13.09%	17.86%	26.26%	36.23%	45.80%	63.96%	71.89%
	Leaf Level Usage	0.12%	0.33%	1.47%	4.09%	9.72%	**16.09%**	30.10%	40.16%
	10NN								
Serial	Index Level Usage	10.33%	10.46%	8.73%	14.26%	42.48%	72.54%	84.79%	86.70%
	Leaf Level Usage	0.14%	0.48%	1.04%	2.39%	**8.40%**	23.18%	45.17%	44.14%
Bulk	Index Level Usage	18.67%	16.77%	25.52%	39.86%	51.53%	63.95%	80.48%	87.54%
	Leaf Level Usage	0.18%	0.75%	3.50%	9.21%	**19.66%**	31.17%	48.78%	62.59%
	100NN								
Serial	Index Level Usage	11.29%	14.27%	16.11%	29.69%	67.42%	91.79%	95.60%	97.83%
	Leaf Level Usage	0.35%	1.30%	3.04%	6.75%	**20.59%**	47.33%	69.26%	71.68%
Bulk	Index Level Usage	20.00%	23.37%	39.56%	56.46%	69.18%	80.94%	93.51%	97.13%
	Leaf Level Usage	0.41%	2.02%	8.16%	**19.39%**	35.80%	53.07%	72.08%	83.96%

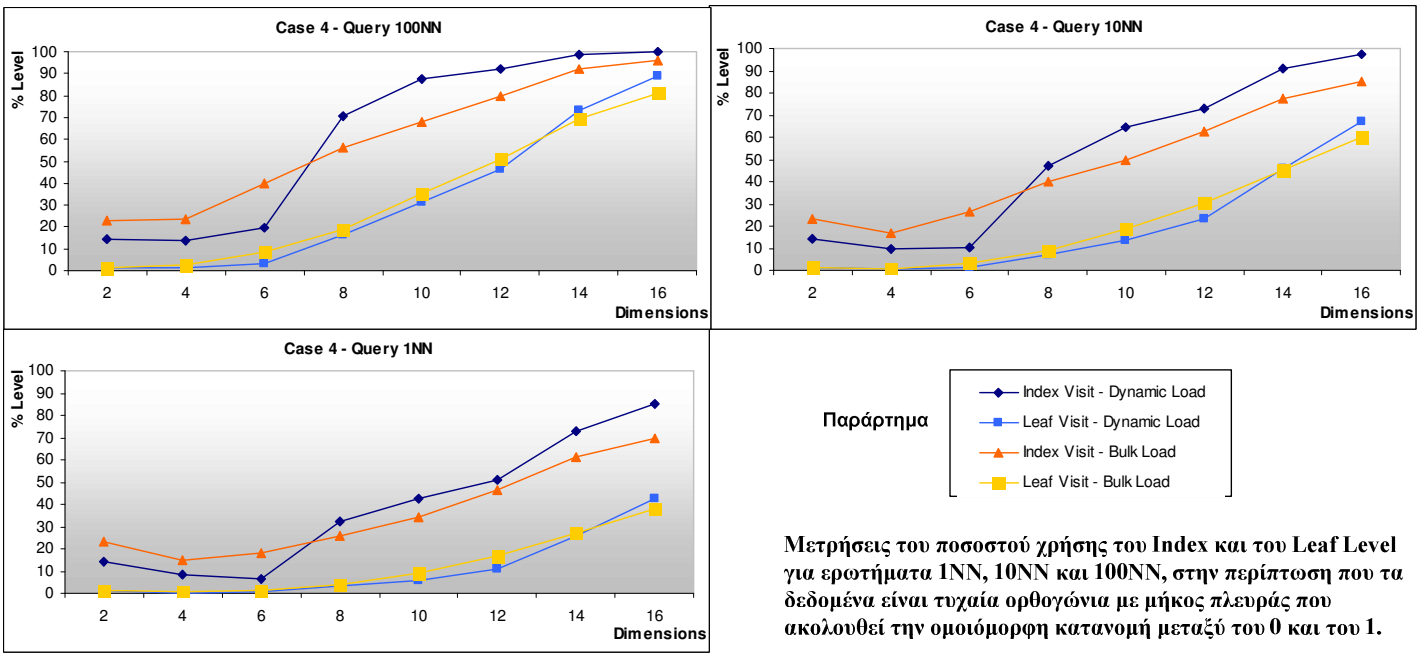


Μετρήσεις του ποσοστού χρήσης του Index και του Leaf Level για ερωτήματα 1NN, 10NN και 100NN, στην περίπτωση που τα δεδομένα είναι σταθερά ορθογώνια με μήκος πλευράς 1% του μήκους του άξονα συντεταγμένων στο χώρο δεδομένων.

Εικόνα 38: Τιμές *Index* και *Leaf Level Usage* για τον τρίτο τύπο δεδομένων.

Πίνακας 4: Τιμές *Index* και *Leaf Level Usage* για την τρίτη περίπτωση δεδομένων σε ερωτήματα *1NN, 10NN, 100NN* και τις εξεταζόμενες διαστάσεις.

Case 3									
	1NN								
	Dimension	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Serial	Index Level Usage	9.36	7.21	5.03	6.67	23.56	74.74	59.21	74.43
	Leaf Level Usage	0.12	0.17	0.34	0.79	3.27	19.12	**19.26**	30.77
Bulk	Index Level Usage	18.33	13.14	17.15	25.57	34.50	45.69	60.48	71.02
	Leaf Level Usage	0.18	0.31	1.34	3.83	9.16	**16.12**	26.96	39.45
	10NN								
Serial	Index Level Usage	9.36	8.87	7.85	12.98	40.85	91.50	79.46	90.59
	Leaf Level Usage	0.13	0.38	0.91	2.19	**8.00**	37.51	38.44	53.07
Bulk	Index Level Usage	18.33	16.51	25.00	39.60	50.34	63.66	79.49	86.55
	Leaf Level Usage	0.18	0.71	3.29	8.90	**18.86**	31.05	47.43	61.30
	100NN								
Serial	Index Level Usage	10.23	11.96	14.30	27.18	70.30	98.61	93.16	98.31
	Leaf Level Usage	0.32	1.05	2.80	**6.21**	20.93	63.88	63.48	78.37
Bulk	Index Level Usage	19.42	22.97	38.88	55.73	68.65	80.09	93.04	96.72
	Leaf Level Usage	0.40	1.97	7.91	**18.99**	35.38	52.25	71.07	83.17



Μετρήσεις του ποσοστού χρήσης του Index και του Leaf Level για ερωτήματα 1NN, 10NN και 100NN, στην περίπτωση που τα δεδομένα είναι τυχαία ορθογώνια με μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.

Εικόνα 39: Τιμές *Index* και *Leaf Level Usage* για τον τέταρτο τύπο δεδομένων.

Πίνακας 5: Τιμές *Index* και *Leaf Level Usage* για την τέταρτη περίπτωση δεδομένων σε ερωτήματα *1NN, 10NN, 100NN* και τις εξεταζόμενες διαστάσεις.

Case 4									
	1NN								
	Dimension	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Serial	Index Level Usage	14.50	8.41	6.45	31.94	42.40	51.20	73.07	85.03
	Leaf Level Usage	1.13	0.27	0.46	3.18	6.02	**11.17**	25.65	42.30
Bulk	Index Level Usage	22.92	14.60	18.07	25.84	34.37	46.17	61.09	69.65
	Leaf Level Usage	1.41	0.50	1.56	4.11	9.07	**16.50**	27.05	38.12
	10NN								
Serial	Index Level Usage	14.50	9.89	10.28	47.22	64.51	73.14	91.03	97.30
	Leaf Level Usage	1.13	0.53	1.16	7.06	**13.68**	23.26	46.07	66.90
Bulk	Index Level Usage	22.92	17.09	26.25	39.98	49.77	62.56	77.62	85.11
	Leaf Level Usage	1.41	0.94	3.48	9.10	**18.51**	30.11	45.30	59.74
	100NN								
Serial	Index Level Usage	14.50	13.81	19.37	70.35	87.55	92.24	98.60	99.85
	Leaf Level Usage	1.13	1.46	3.41	**16.45**	31.55	46.72	73.43	88.74
Bulk	Index Level Usage	22.92	23.83	39.62	56.06	68.29	79.65	92.48	95.99
	Leaf Level Usage	1.41	2.32	8.43	**19.23**	35.03	51.24	69.43	81.26

Από τα παραπάνω γραφήματα και πίνακες προκύπτει το ανώτατο κατώφλι της διάστασης του χώρου δεδομένων, για κάθε ένα από τα ερωτήματα που εξετάστηκαν. Προκειμένου να γενικεύσουμε, στη συνέχεια σημειώνεται το πλήθος των αντικειμένων που αναζητούνται ως ποσοστό του πλήθους των αρχικών δεδομένων (100.000). Οι τιμές που προκύπτουν είναι:

- Αναζήτηση Κοντινότερου Γείτονα (0.01‰) → 12 διαστάσεις
- Αναζήτηση 10 Κοντινότερων Γειτόνων (0.1‰) → 10 διαστάσεις
- Αναζήτηση 100 Κοντινότερων Γειτόνων (1‰) → 8 διαστάσεις

Ως γενικό συμπέρασμα για την αποτελεσματικότητα του *R*-tree*, καταλήγουμε στο ακόλουθο: *η δομή του R*-tree μπορεί να χρησιμοποιηθεί εξασφαλίζοντας υψηλή απόδοση σε χώρους δεδομένων έως 8 διαστάσεων, ενώ το όριο αυτό αυξάνεται και μπορεί να φτάσει στις 12, όταν το πλήθος των αντικειμένων προς αναζήτηση είναι μικρότερο του (1‰) των διαθέσιμων δεδομένων.*

Στο σημείο αυτό αξίζει να τονιστεί ότι, τα παραπάνω γραφήματα και πίνακες δεν μπορούν να αποτελέσουν κριτήριο για τη σύγκριση της αποδοτικότητας μεταξύ των δύο τρόπων κατασκευής της δομής. Η μαζική εισαγωγή δεδομένων δημιουργεί μια πιο συμπαγή δομή με πολύ μικρότερο αριθμό κόμβων, με συνέπεια το ποσοστό χρήσης του *Leaf Level* να είναι υψηλό ακόμη κι αν η αναζήτηση επισκέπτεται μικρό αριθμό κόμβων. Η μεθοδική ανάλυση των αποτελεσμάτων για την παραπάνω σύγκριση γίνεται στο Μέρος 2 που ακολουθεί.

ΜΕΡΟΣ 2: Σύγκριση Μεθόδων Κατασκευής

Οι δύο πιθανοί τρόποι κατασκευής της δομής, σχετίζονται με τον τρόπο εισαγωγής των δεδομένων και είναι:

- Σειριακά, το ένα μετά το άλλο.
- Μαζικά, κατόπιν ομαδοποίησής με τον αλγόριθμο *STR*.

Όπως αναφέρθηκε, για τη σύγκριση των μεθόδων υιοθετούνται τα δύο κριτήρια:

1. Απόδοση κατά την κατασκευή της δομής.
2. Απόδοση κατά την εκτέλεση ερωτημάτων στη δομή.

Όσον αφορά το πρώτο κριτήριο, η σύγκριση θα γίνει βάσει της χρήση του αποθηκευτικού χώρου, μετρώντας (α) το *storage utilization* και (β) το πλήθος των *Disk Accesses*, το οποίο χρησιμοποιείται ως μετρικό και για το δεύτερο κριτήριο.

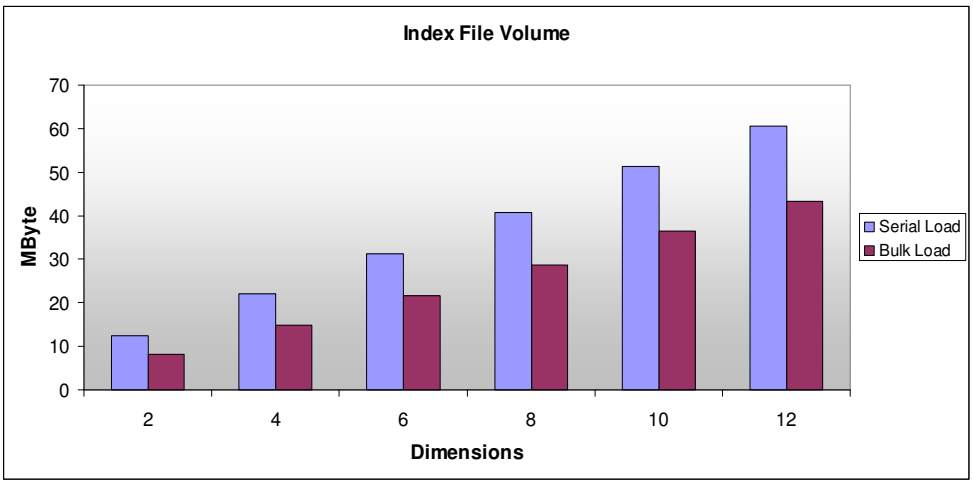
<u>Κριτήριο 1:</u> Απόδοση κατά την κατασκευή της δομής

Ο πίνακας που ακολουθεί περιλαμβάνει τις τιμές του *storage utilization,* για καθεμιά από τις διαστάσεις και τους τύπους δεδομένων που εξετάζουμε. Είναι εμφανές ότι η κατασκευή της δομής με μαζική εισαγωγή δεδομένων οδηγεί σε πολύ υψηλότερο ποσοστό χρήσης της διαθέσιμης μνήμης, το οποίο συνεπάγεται μικρότερο μέγεθος ευρετηρίου, με κόμβους που εμφανίζουν υψηλότερο ποσοστό πληρότητας.

Πίνακας 6: Τιμές του *storage utilization* για τους δύο τρόπους εισαγωγής δεδομένων σε όλες τις διαστάσεις.

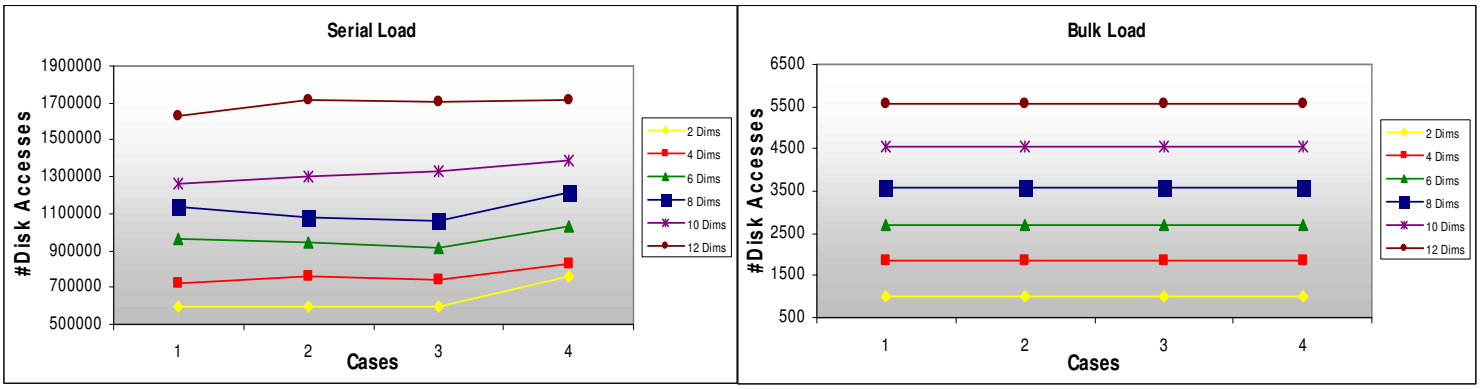
Utilization	Serial Load				Bulk Load			
	Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
2-Dimensional	68%	69%	69%	72%	98%	98%	98%	98%
4-Dimensional	70%	69%	69%	71%	98%	98%	98%	98%
6-Dimensional	71%	70%	70%	72%	97%	97%	97%	97%
8-Dimensional	71%	70%	69%	73%	96%	96%	96%	96%
10-Dimensional	71%	72%	72%	74%	95%	95%	95%	95%
12-Dimensional	73%	74%	75%	75%	94%	94%	94%	94%

Οι διαφορές αυτές απεικονίζονται σε απόλυτες τιμές στο γράφημα που ακολουθεί και απεικονίζει τα μεγέθη, σε Mbytes, των αρχείων ευρετηρίου που δημιουργούν οι δύο μέθοδοι για κάθε μια από τις διαστάσεις του χώρου δεδομένων.

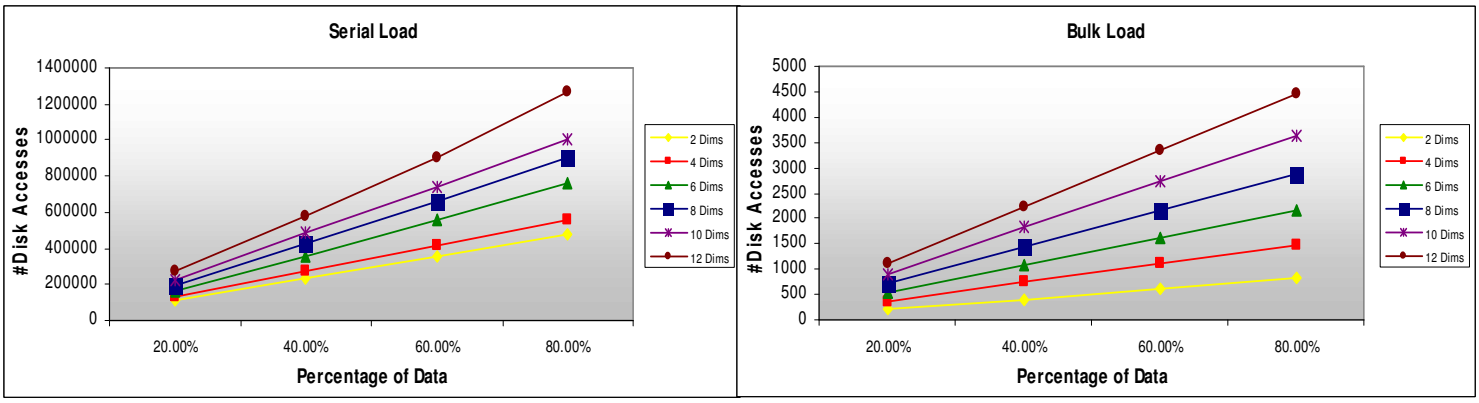


Εικόνα 40: Μέγεθος των αρχείων ευρετηρίου για τις δύο μεθόδους εισαγωγής και όλες τις εξεταζόμενες διαστάσεις.

Όσον αφορά τον αριθμό προσπελάσεων στη μνήμη, παρατίθεται στη συνέχεια δύο ομάδες γραφημάτων. Η πρώτη ομάδα (Εικόνα 41) απεικονίζει το πλήθος των *Disk Accesses* για κάθε διάσταση και τύπο δεδομένων, όταν η δομή κατασκευάζεται με σειριακή (αριστερή στήλη) και μαζική (δεξιά στήλη) εισαγωγή. Η δεύτερη ομάδα (Εικόνα 42) περιλαμβάνει μετρήσεις του πλήθους των *Disk Accesses* ανάλογα με το ποσοστό των δεδομένων που έχουν εισαχθεί στη δομή, εξετάζοντας την περίπτωση των σημειακών αντικειμένων.



Εικόνα 41: Πλήθος των *Disk Accesses* κατά την κατασκευή της δομής όταν τα δεδομένα εισάγονται σειριακά (αριστερά) και μαζικά (δεξιά).



Εικόνα 42: Πλήθος των *Disk Accesses* κατά την εισαγωγή του 20, 40, 60 και 80% των δεδομένων όταν εισάγονται σειριακά (αριστερά) και μαζικά (δεξιά).

Από τα παραπάνω γραφήματα φαίνεται ξεκάθαρα η διαφορά στην απόδοση μεταξύ των δύο μεθόδων εισαγωγής. Στην περίπτωση της σειριακής εισαγωγής, ο αριθμός των *Disk Accesses* είναι της τάξης του 10^6, ενώ στην περίπτωση που η εισαγωγή γίνεται μαζικά η τάξη μεγέθους είναι το 10^3. Με υπολογισμούς από τις ακριβείς τιμές των γραφημάτων, η μείωση στον αριθμό των προσπελάσεων μνήμης που επιτυγχάνεται με τη μαζική εισαγωγή, είναι πάνω από 99.5% σε όλες τις διαστάσεις.

Συμπερασματικά, η καλύτερη απόδοση ως προς την κατασκευή της δομής εξασφαλίζεται από τη μέθοδο μαζικής εισαγωγής των δεδομένων. Η μέθοδος πετυχαίνει σχεδόν βέλτιστη χρήση του διαθέσιμου αποθηκευτικού χώρου με ποσοστά 93%-98%, σε αντίθεση με τη σειριακή εισαγωγή που κυμαίνεται στο 68%-78%. Η δομή που κατασκευάζεται είναι πολύ πιο συμπαγής, με μέγεθος μειωμένο κατά 30% σε σχέση με την δομή που δημιουργείται από τη σειριακή εισαγωγή των δεδομένων. Η υπεροχή αυτή εντοπίζεται και σε όρους προσπελάσεων της μνήμης, αφού η μέθοδος μαζικής εισαγωγής απαιτεί μόλις το 0.5% του πλήθους των *Disk I/Os* που λαμβάνουν χώρα όταν τα δεδομένα εισάγονται σειριακά Έτσι, αποτελεί ξεκάθαρα την καλύτερη επιλογή για την κατασκευή της δομής του *R*-tree*, με την προϋπόθεση ότι τα δεδομένα είναι στατικά ή σχετικά αμετάβλητα.

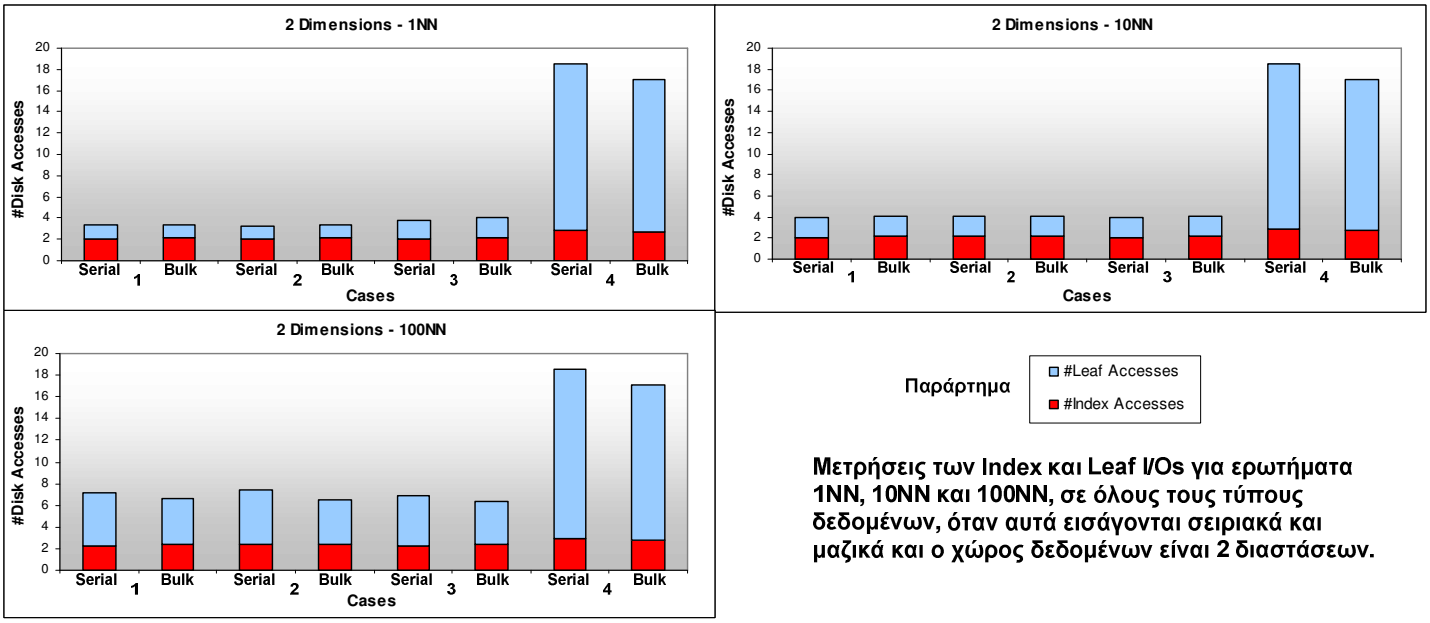
<u>Κριτήριο 2:</u> Απόδοση κατά την εκτέλεση ερωτημάτων στη δομή

Όπως αναφέρθηκε, ως μετρικό για την απόδοση της δομής κατά την εκτέλεση ερωτημάτων θεωρήθηκε το πλήθος των *Disk Accesses*. Στα γραφήματα και τους πίνακες που ακολουθούν η τιμή αυτή αναλύεται στις συνιστώσες τις, δηλαδή τα *Index* και *Leaf Level Accesses*. Σκοπός είναι να δοθεί μια πιο λεπτομερής εικόνα του τρόπου με τον οποίο χρησιμοποιείται η δομή σε κάθε περίπτωση, διαχωρίζοντας τις προσπελάσεις σελίδων της μνήμης σε αυτές που αφορούν την αναζήτηση του κατάλληλου φύλλου (*Index Level Accesses)* και σε αυτές που αφορούν την αναζήτηση του κατάλληλου αντικειμένου σε αυτό (*Leaf Level Accesses).* Η σύγκριση της αποτελεσματικότητας του *R*-tree* στις δύο περιπτώσεις κατασκευής γίνεται με ανάλυση των πειραματικών μετρήσεων, (α) κατά διάσταση του χώρου δεδομένων και (β) κατά τύπο δεδομένων.

Ανάλυση κατά διάσταση του χώρου των δεδομένων

Η ανάλυση που ακολουθεί αποσκοπεί στην εξαγωγή συμπερασμάτων για την αποδοτικότητα των δύο μεθόδων κατασκευής της δομής, για κάθε μια από τις διαστάσεις που εξετάζονται, υπο το πρίσμα των διαφορετικών τύπων δεδομένων.

➢ Χώρος Δεδομένων 2 διαστάσεων



Μετρήσεις των Index και Leaf I/Os για ερωτήματα 1NN, 10NN και 100NN, σε όλους τους τύπους δεδομένων, όταν αυτά εισάγονται σειριακά και μαζικά και ο χώρος δεδομένων είναι 2 διαστάσεων.

Εικόνα 43: Τιμές μετρικών για χώρο δεδομένων 2 διαστάσεων.

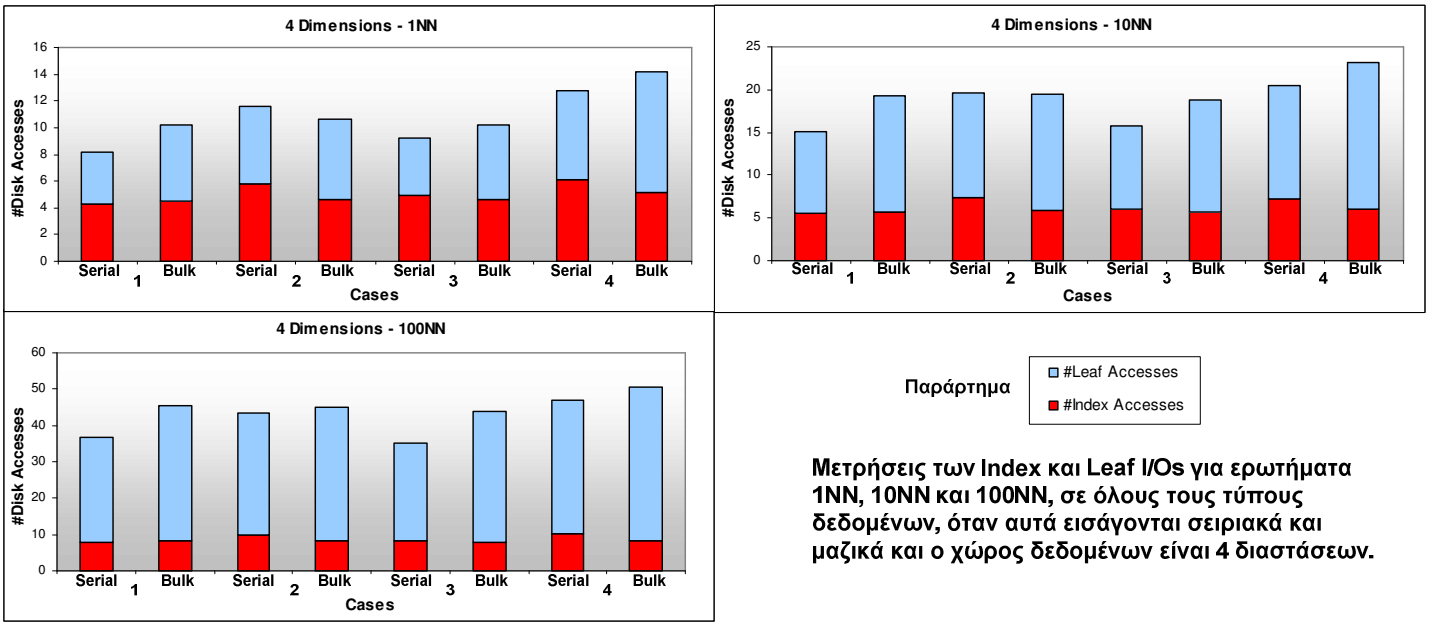
Πίνακας 7: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 2 διαστάσεις.

2-Dimensions	**Case 1**		**Case 2**		**Case 3**		**Case 4**	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	2.05	2.19	2.07	2.19	2.06	2.20	2.90	2.75
# Leaf Accesses	1.31	1.16	1.21	1.18	1.79	1.81	15.58	14.27
# Disk Accesses	3.37	**3.36**	**3.29**	3.38	**3.86**	4.02	18.49	**17.03**
10NN								
# Index Accesses	2.08	2.23	2.17	2.24	2.06	2.20	2.90	2.75
# Leaf Accesses	1.88	1.83	1.96	1.84	1.87	1.87	15.58	14.27
# Disk Accesses	**3.97**	4.07	4.14	**4.09**	**3.94**	4.08	18.49	**17.03**
100NN								
# Index Accesses	2.26	2.41	2.37	2.40	2.25	2.33	2.90	2.75
# Leaf Accesses	4.91	4.15	5.04	4.13	4.64	4.03	15.61	14.30
# Disk Accesses	7.18	**6.57**	7.42	**6.54**	6.90	**6.37**	18.52	**17.06**

Στην περίπτωση χώρου δεδομένων 2 διαστάσεων, η απόδοση των δύο μεθόδων κατασκευής κυμαίνεται στα ίδια επίπεδα. Η μεγαλύτερες διαφορές εντοπίζονται όταν (α) ο αριθμός των κοντινότερων γειτόνων είναι μεγάλος (βλέπε 100NN), ή (β) τα δεδομένα έχουν πιο πολύπλοκη μορφή (βλέπε *Case* 4). Στις περιπτώσεις αυτές η κατασκευή της δομής με μαζική εισαγωγή δεδομένων οδηγεί σε μείωση του αριθμού προσπελάσεων στη μνήμη, από 7.5% έως 12% σε σχέση με τη σειριακή εισαγωγή.

Συμπέρασμα: *η απόδοση του R*-tree κυμαίνεται στα ίδια επίπεδα για τις δύο μεθόδους κατασκευής, με τη μαζική εισαγωγή να εξασφαλίζει αισθητά καλύτερη απόδοση (7.5%-12%) όσο αυξάνεται το πλήθος των κοντινότερων γειτόνων και η πολυπλοκότητα των αντικειμένων.*

- Χώρος Δεδομένων 4 διαστάσεων



Μετρήσεις των Index και Leaf I/Os για ερωτήματα 1NN, 10NN και 100NN, σε όλους τους τύπους δεδομένων, όταν αυτά εισάγονται σειριακά και μαζικά και ο χώρος δεδομένων είναι 4 διαστάσεων.

Εικόνα 44: Τιμές μετρικών για χώρο δεδομένων 4 διαστάσεων.

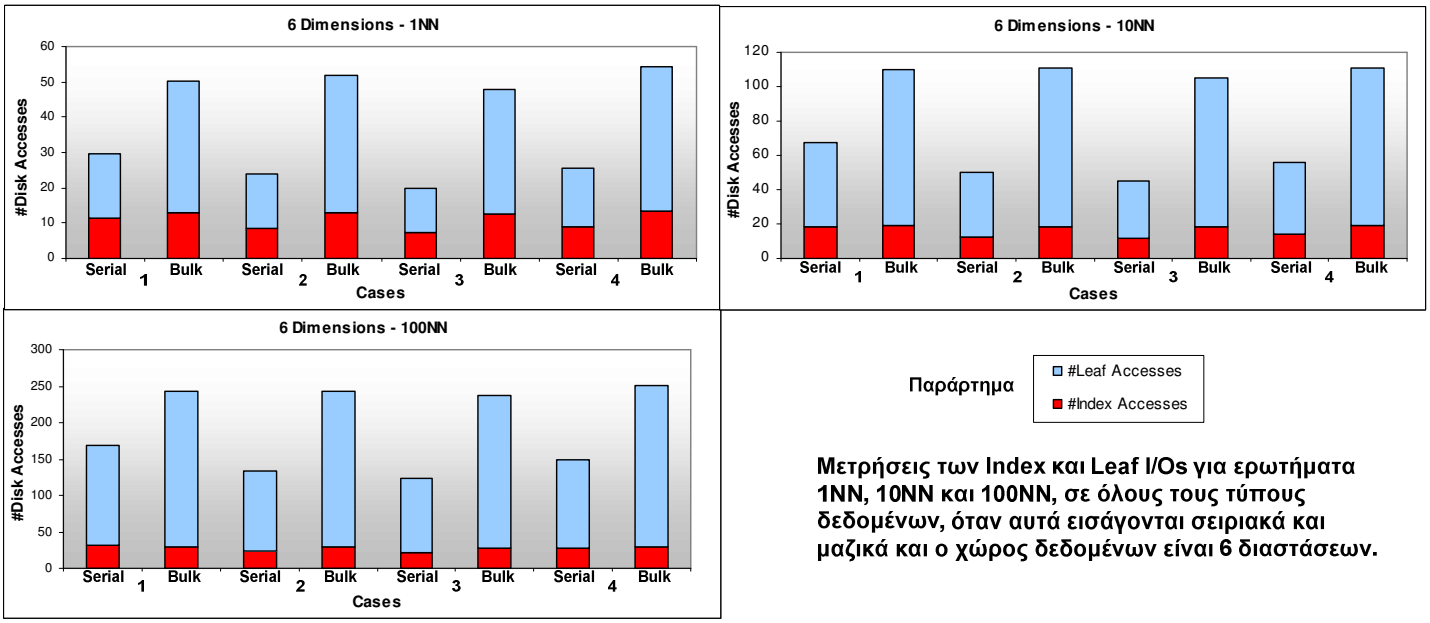
Πίνακας 8: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 4 διαστάσεις.

4-Dimensions	**Case 1**		**Case 2**		**Case 3**		**Case 4**	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	4.32	4.54	5.75	4.58	4.90	4.60	6.14	5.11
# Leaf Accesses	3.83	5.65	5.90	6.00	4.29	5.61	6.67	9.03
# Disk Accesses	**8.16**	10.20	11.66	**10.59**	**9.20**	10.22	**12.82**	14.15
10NN								
# Index Accesses	5.52	5.73	7.32	5.87	6.03	5.78	7.22	5.98
# Leaf Accesses	9.57	13.50	12.27	13.59	9.79	12.93	13.27	17.18
# Disk Accesses	**15.10**	19.24	19.60	**19.47**	**15.83**	18.72	**20.50**	23.17
100NN								
# Index Accesses	8.09	8.19	9.99	8.18	8.13	8.04	10.08	8.34
# Leaf Accesses	28.53	37.01	33.35	36.78	26.96	35.83	36.71	42.23
# Disk Accesses	**36.63**	45.21	**43.35**	44.97	**35.10**	43.88	**46.80**	50.58

Στην περίπτωση χώρου δεδομένων 4 διαστάσεων, η μέθοδος της σειριακής εισαγωγής δεδομένων παρουσιάζει καλύτερη απόδοση σε όλες σχεδόν τις περιπτώσεις. Το κέρδος σε όρους *Disk I/Os*, σχετίζεται με τον τύπο των δεδομένων και (α) για χωρικά αντικείμενα κυμαίνεται γύρω στο 20%, (β) για σταθερά ορθογώνια με μικρές πλευρές είναι σχεδόν μηδενικό, (γ) για σταθερά ορθογώνια με μεγαλύτερες πλευρές αυξάνεται από 10% σε 20% με την αύξηση των κοντινότερων γειτόνων και (δ) για τυχαία ορθογώνια βρίσκεται μεταξύ 7.5% και 11.5%.

<u>Συμπέρασμα:</u> *η απόδοση του R*-tree με σειριακή εισαγωγή δεδομένων υπερτερεί της μαζικής, με τη μέγιστη διαφορά στη διαχείριση σημειακών δεδομένων (20%) και μικρότερες διαφορές (10%-20%) για αύξηση του όγκου και της πολυπλοκότητας των χωρικών αντικειμένων (7.5%-11.5%).*

- Χώρος Δεδομένων 6 διαστάσεων



Εικόνα 45: Τιμές μετρικών για χώρο δεδομένων 6 διαστάσεων.

Πίνακας 9: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 6 διαστάσεις.

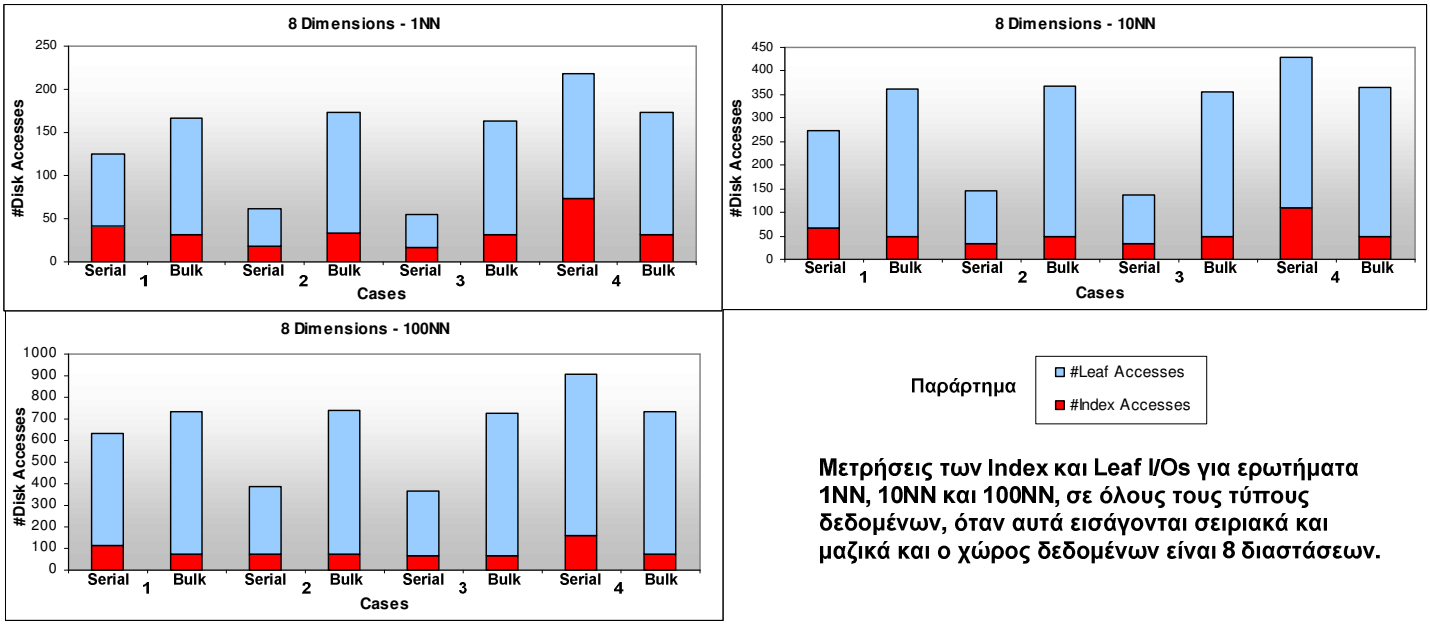
6-Dimensions	Case 1		Case 2		Case 3		Case 4	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	11.27	13.04	8.45	13.04	7.44	12.52	9.10	13.19
# Leaf Accesses	18.35	37.05	15.62	38.80	12.43	35.31	16.42	41.03
# Disk Accesses	**29.63**	50.10	**24.08**	51.85	**19.88**	47.84	**25.53**	54.23

10NN								
# Index Accesses	18.64	18.85	12.31	18.63	11.62	18.25	14.49	19.16
# Leaf Accesses	49.14	91.56	37.84	92.00	33.46	86.61	41.39	91.71
# Disk Accesses	**67.79**	110.42	**50.16**	110.64	**45.09**	104.87	**55.89**	110.88
100NN								
# Index Accesses	31.85	28.82	22.71	28.88	21.17	28.38	27.31	28.92
# Leaf Accesses	136.94	213.60	110.41	214.86	102.46	208.28	121.32	221.83
# Disk Accesses	**168.80**	242.43	**133.13**	243.75	**123.64**	236.67	**148.64**	250.76

Στην περίπτωση χώρου δεδομένων 6 διαστάσεων, η μέθοδος της σειριακής εισαγωγής δεδομένων παρουσιάζει καθολικά αποτελεσματικότερη συμπεριφορά σε σχέση με τη μαζική εισαγωγή. Το κέρδος σε όρους *Disk I/Os* είναι σε κάθε περίπτωση άνω του 30% και φτάνει έως 58.5%. Η μέγιστη διαφορά εντοπίζεται για σταθερά ορθογώνια μεγάλου μήκους πλευράς και αναζήτηση του κοντινότερου γείτονα, παρουσιάζει μείωση με την αύξηση του αριθμού των κοντινότερων γειτόνων, αλλά παραμένει σε σημαντικά επίπεδα για όλες τις περιπτώσεις.

<u>Συμπέρασμα:</u> *η κατασκευή του R*-tree με σειριακή εισαγωγή δεδομένων παρουσιάζει εμφανώς καλύτερη αποτελεσματικότητα (30%-58.5%) για κάθε τύπο δεδομένων και όλα τα πλήθη κοντινότερων γειτόνων.*

- Χώρος Δεδομένων 8 διαστάσεων



Εικόνα 46: Τιμές μετρικών για χώρο δεδομένων 8 διαστάσεων.

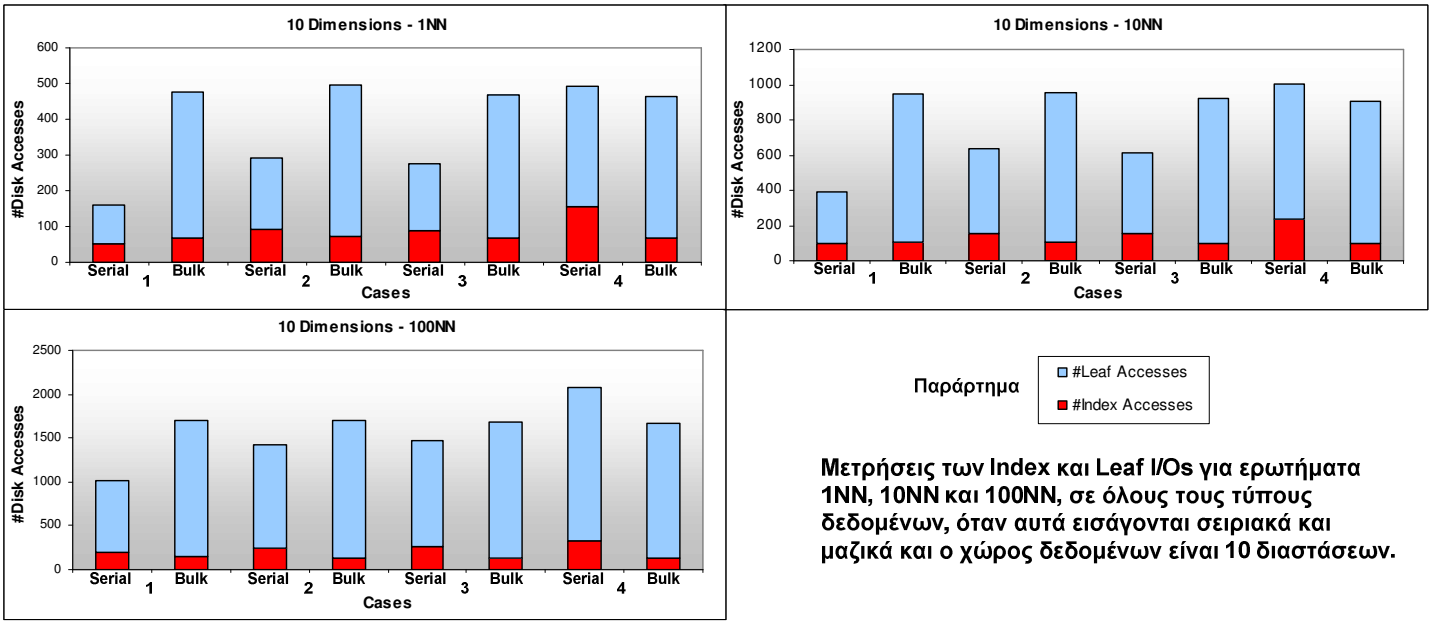
Πίνακας 10: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 8 διαστάσεις.

8-Dimensions	**Case 1**		**Case 2**		**Case 3**		**Case 4**	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	41.62	32.47	18.95	32.82	16.40	31.96	73.46	32.30
# Leaf Accesses	83.62	134.71	42.00	141.08	37.98	131.98	144.18	141.81
# Disk Accesses	**125.25**	167.19	**60.96**	173.91	**54.39**	163.95	217.65	**174.12**
10NN								
# Index Accesses	68.37	49.40	34.52	49.82	31.94	49.50	108.60	49.98
# Leaf Accesses	205.23	312.14	112.43	317.56	104.52	306.98	319.76	313.90
# Disk Accesses	**273.61**	361.55	**146.96**	367.39	**136.47**	356.49	428.37	**363.89**
100NN								
# Index Accesses	115.76	70.43	71.84	70.57	66.87	69.66	161.80	70.07
# Leaf Accesses	517.54	664.89	318.16	668.60	296.65	655.03	745.71	663.17
# Disk Accesses	**633.31**	735.33	**390.01**	739.18	**363.53**	724.70	907.52	**733.25**

Στην περίπτωση χώρου δεδομένων 8 διαστάσεων, παρατηρείται μια νέα "ισορροπία" μεταξύ των δύο μεθόδων εισαγωγής. Η σειριακή υπερέχει όταν τα δεδομένα είναι σημεία ή σταθερά ορθογώνια, ενώ η μαζική σε περισσότερο πολύπλοκες μορφές. Το κέρδος της πρώτης κυμαίνεται από 14% έως 25% για σημειακά δεδομένα και από 47% έως 66% για σταθερά ορθογώνια. Αντίστοιχα, η μαζική εισαγωγή οδηγεί σε μειωμένο αριθμό προσπελάσεων από 15% έως 20% όταν τα ορθογώνια έχουν τυχαίο μήκος πλευρών. Όπως και στις προηγούμενες περιπτώσεις, η αύξηση του πλήθους των κοντινότερων γειτόνων οδηγεί σε μείωση των παραπάνω τιμών.

<u>Συμπέρασμα:</u> *η διαχείριση "κανονικοποιημένων" αντικειμένων, όπως σημείων ή ορθογωνίων με σταθερά μήκη πλευρών ωφελείται από τη σειριακή εισαγωγή (14%-25% και 47%-66% αντίστοιχα), ενώ για πολύπλοκα αντικείμενα προτιμάται η μέθοδος της μαζικής εισαγωγής (15%-20%).*

➢ Χώρος Δεδομένων 10 διαστάσεων



Εικόνα 47: Τιμές μετρικών για χώρο δεδομένων 10 διαστάσεων.

Πίνακας 11: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 10 διαστάσεις.

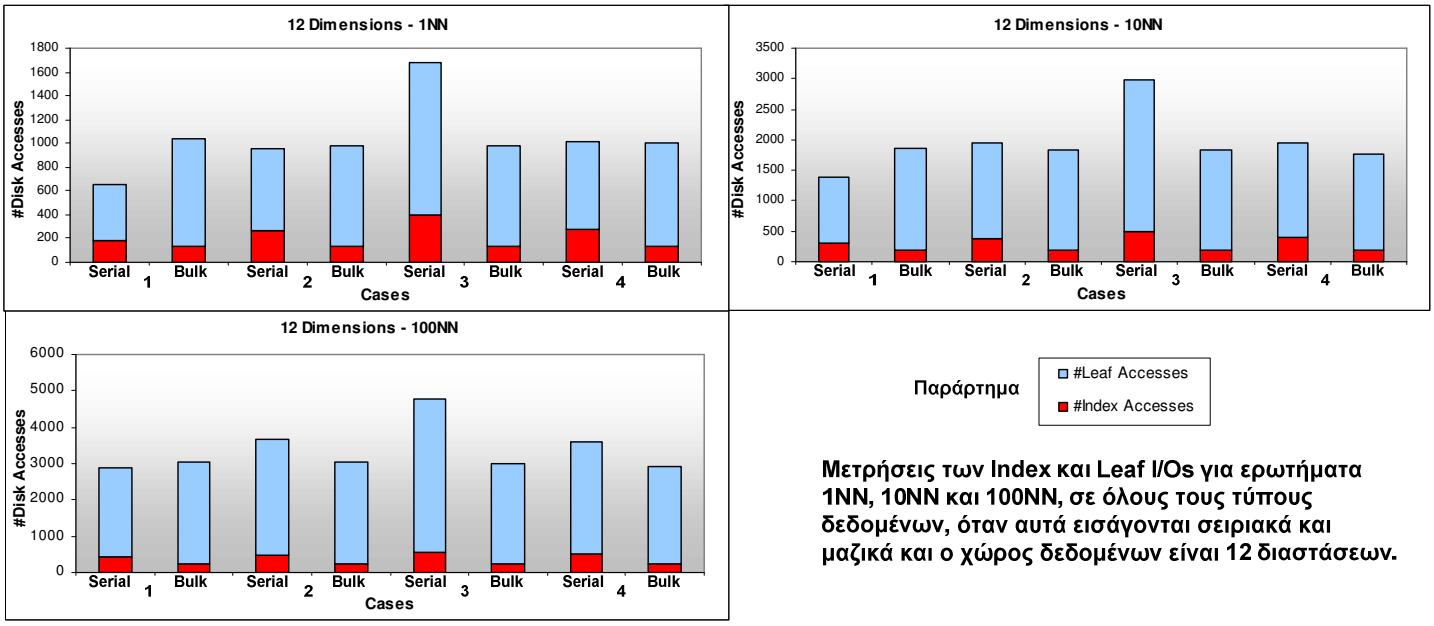
10-Dimensions	Case 1		Case 2		Case 3		Case 4	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	51.20	69.67	91.87	72.46	89.75	68.99	155.59	68.73
# Leaf Accesses	108.56	405.62	202.01	422.76	186.73	398.29	336.17	394.43
# Disk Accesses	**159.77**	475.30	**293.89**	495.23	**276.49**	467.29	491.77	**463.17**
10NN								
# Index Accesses	100.02	103.01	158.01	103.05	155.63	100.68	236.76	99.53
# Leaf Accesses	291.75	843.11	479.79	854.63	456.66	820.21	763.62	804.64
# Disk Accesses	**391.78**	946.13	**637.81**	957.69	**612.30**	920.90	1000.39	**904.18**
100NN								
# Index Accesses	201.57	138.98	250.79	138.36	267.85	137.29	321.31	136.57
# Leaf Accesses	809.87	1554.60	1176.34	1556.61	1194.85	1538.12	1761.86	1522.92
# Disk Accesses	**1011.45**	1693.59	**1427.14**	1694.98	**1462.71**	1675.42	2083.18	**1659.50**

Στην περίπτωση χώρου δεδομένων 10 διαστάσεων, παρατηρείται η ίδια κατάσταση με το χώρο 8 διαστάσεων. Η μεγαλύτερη διαφορά στην απόδοση ωστόσο εντοπίζεται για σημειακά αντικείμενα με κέρδος από 40% έως 66%, ενώ για σταθερά ορθογώνια η τιμή του διαμορφώνεται στο εύρος 12% έως 40%. Η μαζική εισαγωγή υπερτερεί και πάλι στην περίπτωση που τα ορθογώνια έχουν τυχαίο μήκος πλευρών, με μείωση των *Disk Accesses* από 5% έως 20%. Η διαφορά των δύο μεθόδων μειώνεται με την

αύξηση των κοντινότερων γειτόνων για τη σειριακή εισαγωγή, ενώ έχει αντίστροφη τάση στην περίπτωση της μαζικής.

<u>Συμπέρασμα:</u> *η σειριακή εισαγωγή κρίνεται καταλληλότερη για διαχείριση σημείων (40%-66%) και ορθογωνίων με σταθερά μήκη πλευρών (12%-40%), ενώ η μαζική εισαγωγή είναι αποδοτικότερη στην περίπτωση τυχαίων ορθογωνίων (5%-20%).*

- Χώρος Δεδομένων 12 διαστάσεων



Εικόνα 48: Τιμές μετρικών για χώρο δεδομένων 12 διαστάσεων.

Πίνακας 12: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, όταν ο χώρος δεδομένων έχει 12 διαστάσεις.

12-Dimensions	**Case 1**		**Case 2**		**Case 3**		**Case 4**	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN								
# Index Accesses	185.48	136.95	263.18	134.65	404.34	134.33	277.01	135.73
# Leaf Accesses	471.30	901.00	692.73	846.77	1271.34	848.49	742.67	868.55
# Disk Accesses	**656.79**	1037.96	**955.92**	981.43	1675.69	**982.83**	1019.69	**1004.29**
10NN								
# Index Accesses	307.01	189.19	386.65	188.00	495.02	187.17	395.70	183.93
# Leaf Accesses	1077.00	1669.29	1551.99	1640.71	2494.35	1634.44	1546.88	1585.07
# Disk Accesses	**1384.02**	1858.49	1938.65	**1828.72**	2989.38	**1821.62**	1942.59	**1769.01**
100NN								
# Index Accesses	448.79	238.55	489.26	237.96	533.49	235.46	499.00	234.16
# Leaf Accesses	2416.34	2806.36	3169.32	2793.37	4248.15	2750.49	3106.22	2697.30
# Disk Accesses	**2865.14**	3044.92	3658.59	**3031.34**	4781.65	**2985.96**	3605.23	**2931.47**

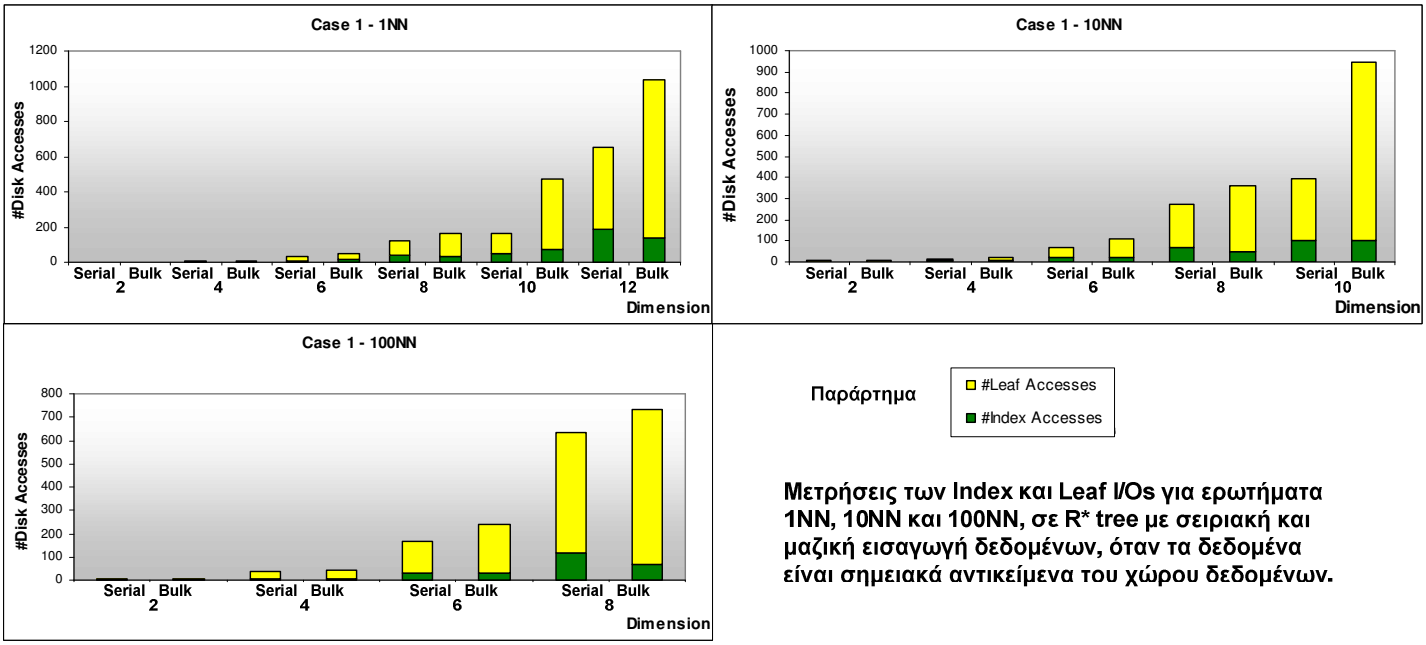
Στην περίπτωση χώρου δεδομένων 12 διαστάσεων, η μέθοδος σειριακής εισαγωγής υπερτερεί μονάχα στην περίπτωση που τα δεδομένα είναι σημειακά, ενώ στις υπόλοιπες περιπτώσεις η μαζική εισαγωγή εξασφαλίζει παρόμοια ή καλύτερη απόδοση. Συγκεκριμένα, το κέρδος της σειριακής εισαγωγής φτάνει το 36% για σημειακά αντικείμενα. Από την άλλη, η απόδοση είναι περίπου ίδια για σταθερά ορθογώνια με μικρό μήκος πλευρών, ενώ το πλεονέκτημα της μαζικής εισαγωγής εμφανίζεται στις περιπτώσεις σταθερών ορθογωνίων με μεγάλο και τυχαίο μήκος πλευρών όπου οδηγεί σε κέρδος 37%-41% και έως 18% αντίστοιχα.

Συμπέρασμα: *για σημειακά αντικείμενα επιλέγεται η μέθοδος σειριακής εισαγωγής (6%-36%), ενώ όσο αυξάνεται ο όγκος και η πολυπλοκότητα του σχήματος των αντικειμένων, τόσο περισσότερο ενδείκνυται η επιλογή της μαζικής εισαγωγής (37-41% και έως 18% αντίστοιχα).*

Ανάλυση κατά τύπο δεδομένων

Η ανάλυση που ακολουθεί αποσκοπεί στην εξαγωγή συμπερασμάτων για την αποδοτικότητα των δύο μεθόδων κατασκευής της δομής, για κάθε έναν από τους διαφορετικούς τύπους δεδομένων, υπο το πρίσμα της αύξησης των διαστάσεων.

- Σημειακά Αντικείμενα



Μετρήσεις των Index και Leaf I/Os για ερωτήματα 1NN, 10NN και 100NN, σε R* tree με σειριακή και μαζική εισαγωγή δεδομένων, όταν τα δεδομένα είναι σημειακά αντικείμενα του χώρου δεδομένων.

Εικόνα 49: Τιμές μετρικών για την περίπτωση σημειακών αντικειμένων.

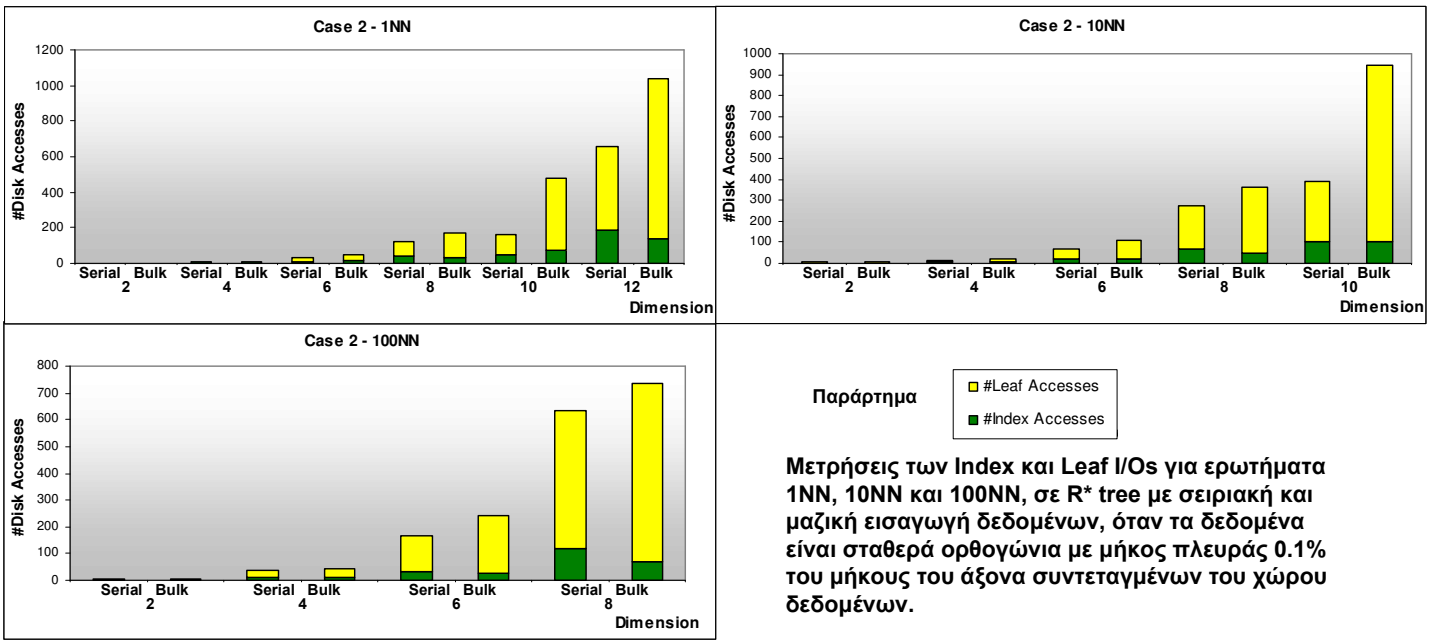
Πίνακας 13: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για σημειακά αντικείμενα.

Case 1	2-Dims		4-Dims		6-Dims		8-Dims		10-Dims		12-Dims	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN												
# Index Accesses	2.05	2.19	4.32	4.54	11.27	13.04	41.62	32.47	51.20	69.67	185.48	136.95
# Leaf Accesses	1.31	1.16	3.83	5.65	18.35	37.05	83.62	134.71	108.56	405.62	471.30	901.00
# Disk Accesses	3.37	**3.36**	**8.16**	10.20	**29.63**	50.10	**125.25**	167.19	**159.77**	475.30	**656.79**	1037.96
10NN												
# Index Accesses	2.08	2.23	5.52	5.73	18.64	18.85	68.37	49.40	100.02	103.01	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**	
# Leaf Accesses	1.88	1.83	9.57	13.50	49.14	91.56	205.23	312.14	291.75	843.11		
# Disk Accesses	**3.97**	4.07	**15.10**	19.24	**67.79**	110.42	**273.61**	361.55	**391.78**	946.13		
100NN												
# Index Accesses	2.26	2.41	8.09	8.19	31.85	28.82	115.76	70.43	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**			
# Leaf Accesses	4.91	4.15	28.53	37.01	136.94	213.60	517.54	664.89				
# Disk Accesses	7.18	**6.57**	**36.63**	45.21	**168.80**	242.43	**633.31**	735.33				

Για την περίπτωση σημειακών αντικειμένων, η μέθοδος σειριακής εισαγωγής υπερτερεί καθολικά ανεξαρτήτως διάστασης χώρου δεδομένων και πλήθους αναζητούμενων κοντινότερων γειτόνων. Η αποδοτικότητα των δύο μεθόδων είναι παραπλήσια στην περίπτωση των δυσδιάστατων δεδομένων, όμως το ποσοστό κέρδους σε μεγαλύτερες διαστάσεις κυμαίνεται μεταξύ 20% και 66% για 1 γείτονα, μεταξύ 21% και 58% για 10 γείτονες και μεταξύ 14% και 30% για 100 γείτονες.

<u>Συμπέρασμα:</u> *οι δύο μέθοδοι εισαγωγής εμφανίζουν παραπλήσια απόδοση στην περίπτωση του δυσδιάστατου χώρου δεδομένων, ενώ για μεγαλύτερες διαστάσεις, η μέθοδος σειριακής εισαγωγής εξασφαλίζει σημαντικά αποδοτικότερη λειτουργία για οποιοδήποτε πλήθος αναζητούμενων κοντινότερων γειτόνων.*

- Ορθογώνια σταθερού μήκους πλευράς ίσο με το 0.1% του άξονα συντεταγμένων στο χώρο δεδομένων.



Εικόνα 50: Τιμές μετρικών για την περίπτωση ορθογωνίων με σταθερό μήκος πλευράς (0.1% του άξονα συντεταγμένων).

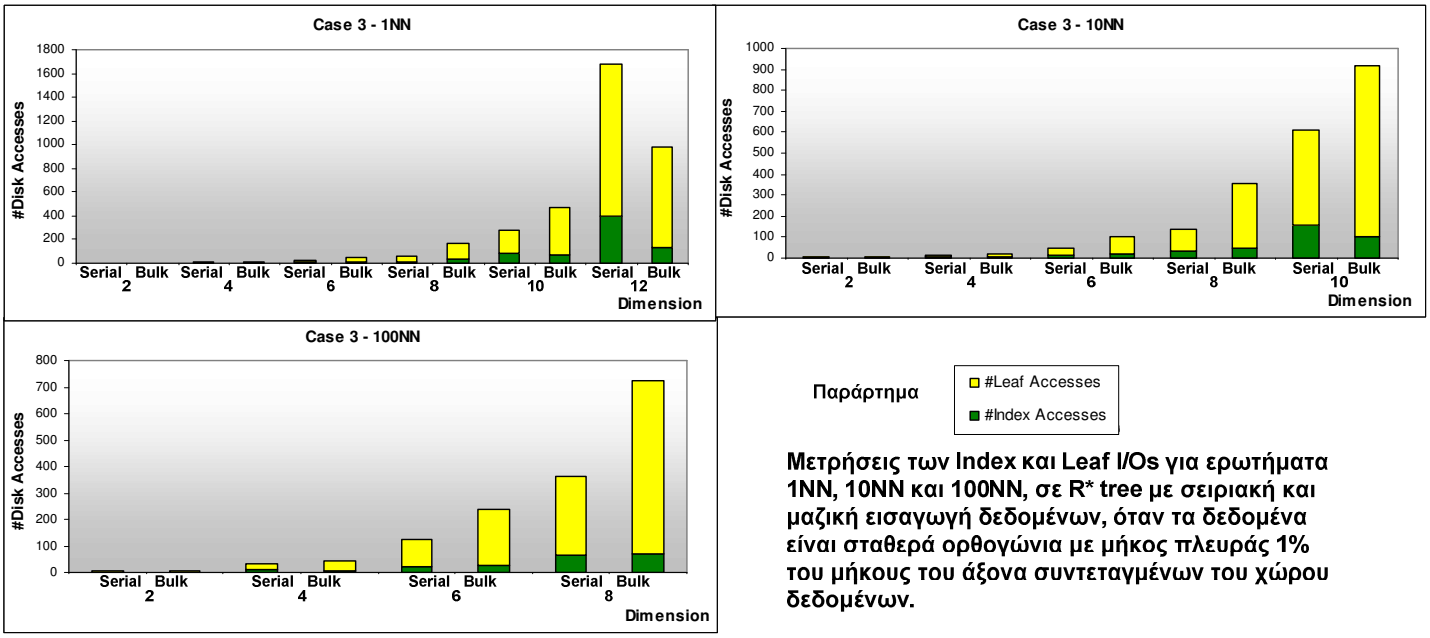
Πίνακας 14: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με σταθερό μήκος πλευράς (0.1% του άξονα συντεταγμένων).

Case 2	2-Dims		4-Dims		6-Dims		8-Dims		10-Dims		12-Dims	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN												
# Index Accesses	2.05	2.19	4.32	4.54	11.27	13.04	41.62	32.47	51.20	69.67	185.48	136.95
# Leaf Accesses	1.31	1.16	3.83	5.65	18.35	37.05	83.62	134.71	108.56	405.62	471.30	901.00
# Disk Accesses	**3.29**	3.38	11.66	**10.59**	**24.08**	51.85	**60.96**	173.91	**293.89**	495.23	**955.92**	981.43
10NN												
# Index Accesses	2.08	2.23	5.52	5.73	18.64	18.85	68.37	49.40	100.02	103.01	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**	
# Leaf Accesses	1.88	1.83	9.57	13.50	49.14	91.56	205.23	312.14	291.75	843.11		
# Disk Accesses	4.14	**4.09**	19.60	**19.47**	**50.16**	110.64	**146.96**	367.39	**637.81**	957.69		
100NN												
# Index Accesses	2.26	2.41	8.09	8.19	31.85	28.82	115.76	70.43	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**			
# Leaf Accesses	4.91	4.15	28.53	37.01	136.94	213.60	517.54	664.89				
# Disk Accesses	7.42	**6.54**	**43.35**	44.97	**133.13**	243.75	**390.01**	739.18				

Στην περίπτωση ορθογωνίων με σταθερό μικρό μήκος πλευράς, η απόδοση των δύο μεθόδων είναι εφάμιλλη έως τις 4 διαστάσεις. Ωστόσο, η μείωση των προσπελάσεων μνήμης που πετυχαίνει η σειριακή μέθοδος σε υψηλότερες διαστάσεις είναι σημαντική και κυμαίνεται από 33% έως 65%.

Συμπέρασμα: *η αποτελεσματικότητα των δύο μεθόδων είναι εφάμιλλη όταν ο χώρος δεδομένων περιλαμβάνει μέχρι και 4 διαστάσεις, αλλά για πιο πολυδιάστατους χώρους η σειριακή μέθοδος εξασφαλίζει σημαντικά μεγαλύτερη αποτελεσματικότητα (33%-65%).*

- Ορθογώνια σταθερού μήκους πλευράς ίσο με το 1% του άξονα συντεταγμένων στο χώρο δεδομένων.



Μετρήσεις των Index και Leaf I/Os για ερωτήματα 1NN, 10NN και 100NN, σε R* tree με σειριακή και μαζική εισαγωγή δεδομένων, όταν τα δεδομένα είναι σταθερά ορθογώνια με μήκος πλευράς 1% του μήκους του άξονα συντεταγμένων του χώρου δεδομένων.

Εικόνα 51: Τιμές μετρικών για την περίπτωση ορθογωνίων με σταθερό μήκος πλευράς (1% του άξονα συντεταγμένων).

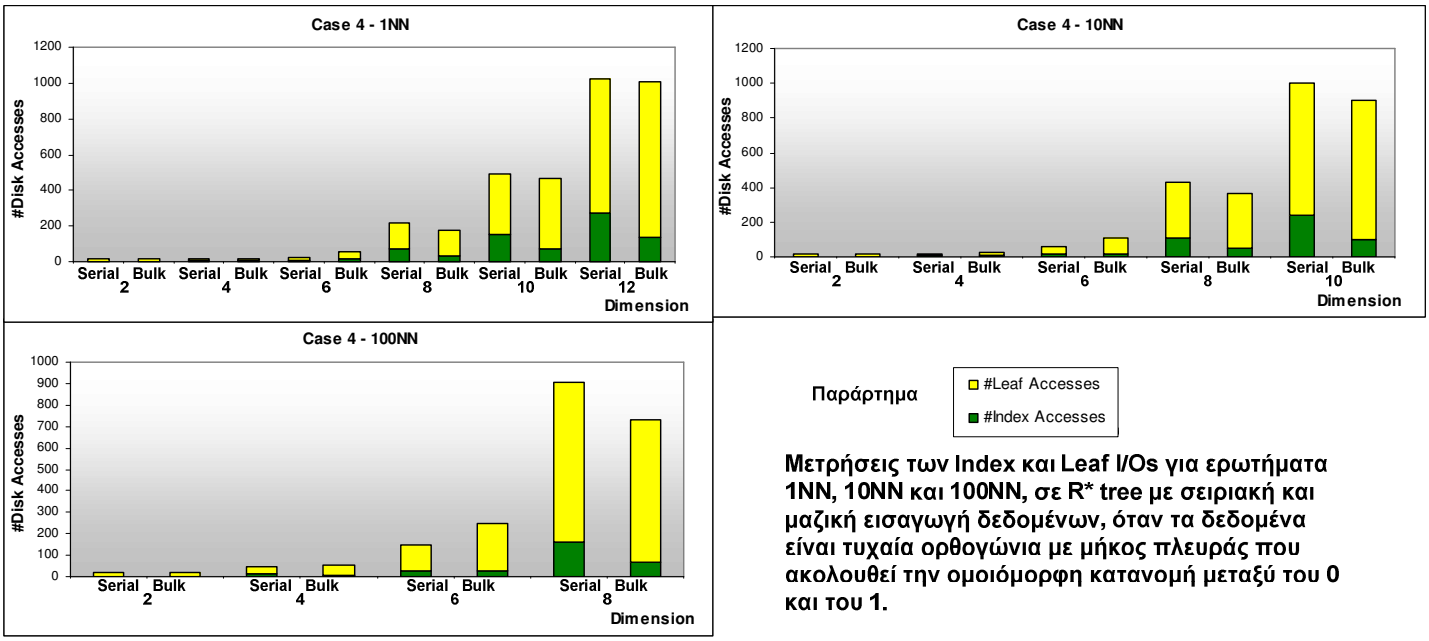
Πίνακας 15: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με σταθερό μήκος πλευράς (1% του άξονα συντεταγμένων).

Case 3	2-Dims		4-Dims		6-Dims		8-Dims		10-Dims		12-Dims	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN												
# Index Accesses	2.06	2.20	4.90	4.60	7.44	12.52	16.40	31.96	89.75	68.99	404.34	134.33
# Leaf Accesses	1.79	1.81	4.29	5.61	12.43	35.31	37.98	131.98	186.73	398.29	1271.34	848.49
# Disk Accesses	**3.86**	4.02	**9.20**	10.22	**19.88**	47.84	**54.39**	163.95	**276.49**	467.29	1675.69	**982.83**
10NN												
# Index Accesses	2.06	2.20	6.03	5.78	11.62	18.25	31.94	49.50	155.63	100.68	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**	
# Leaf Accesses	1.87	1.87	9.79	12.93	33.46	86.61	104.52	306.98	456.66	820.21		
# Disk Accesses	**3.94**	4.08	**15.83**	18.72	**45.09**	104.87	**136.47**	356.49	**612.30**	920.90		
100NN												
# Index Accesses	2.25	2.33	8.13	8.04	21.17	28.38	66.87	69.66	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**			
# Leaf Accesses	4.64	4.03	26.96	35.83	102.46	208.28	296.65	655.03				
# Disk Accesses	6.90	**6.37**	**35.10**	43.88	**123.64**	236.67	**363.53**	724.70				

Στην περίπτωση ορθογωνίων με σταθερό μεγάλο μήκος πλευράς, συναντάται η ίδια "συμπεριφορά". Οι δύο μέθοδοι παρουσιάζουν παρόμοια απόδοση στις 2 και 4 διαστάσεις, με την σειριακή μέθοδο να πετυχαίνει μείωση *Disk I/Os* από 33% έως 68% στις υψηλότερες διαστάσεις. Η μόνη διαφορά εντοπίζεται στην περίπτωση των 12 διαστάσεων, όπου ως αποδοτικότερη εμφανίζεται η μέθοδος μαζικής εισαγωγής, οδηγώντας σε μείωση των προσπελάσεων μνήμης κατά 41%.

<u>Συμπέρασμα:</u> *για χώρο δεδομένων έως 4 διαστάσεων οι δύο μέθοδοι εξασφαλίζουν παρόμοια αποτελεσματικότητα, η αύξηση στις 10 διαστάσεις "εξυπηρετείται" καλύτερα από τη σειριακή εισαγωγή (33%-68%) και για τις 12 διαστάσεις, προτιμάται η μαζική εισαγωγή (41%).*

- Ορθογώνια με τυχαίο μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.



Μετρήσεις των Index και Leaf I/Os για ερωτήματα 1NN, 10NN και 100NN, σε R* tree με σειριακή και μαζική εισαγωγή δεδομένων, όταν τα δεδομένα είναι τυχαία ορθογώνια με μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.

Εικόνα 52: Τιμές μετρικών για την περίπτωση ορθογωνίων με τυχαίο μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.

Πίνακας 16: Πλήθος προσπελάσεων για τους κόμβους, τα φύλλα και συνολικά, για ορθογώνια με τυχαίο μήκος πλευράς που ακολουθεί την ομοιόμορφη κατανομή μεταξύ του 0 και του 1.

Case 4	2-Dims		4-Dims		6-Dims		8-Dims		10-Dims		12-Dims	
	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk	Serial	Bulk
1NN												
# Index Accesses	2.90	2.75	6.14	5.11	9.10	13.19	73.46	32.30	155.59	68.73	277.01	135.73
# Leaf Accesses	15.58	14.27	6.67	9.03	16.42	41.03	144.18	141.81	336.17	394.43	742.67	868.55
# Disk Accesses	18.49	**17.03**	**12.82**	14.15	**25.53**	54.23	217.65	**174.12**	491.77	**463.17**	1019.69	**1004.29**
10NN												
# Index Accesses	2.90	2.75	7.22	5.98	14.49	19.16	108.60	49.98	236.76	99.53	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**	
# Leaf Accesses	15.58	14.27	13.27	17.18	41.39	91.71	319.76	313.90	763.62	804.64		
# Disk Accesses	18.49	**17.03**	**20.50**	23.17	**55.89**	110.88	428.37	**363.89**	1000.39	**904.18**		
100NN												
# Index Accesses	2.90	2.75	10.08	8.34	27.31	28.92	161.80	70.07	**ΚΑΚΗ ΛΕΙΤΟΥΡΓΙΑ ΔΟΜΗΣ**			
# Leaf Accesses	15.61	14.30	36.71	42.23	121.32	221.83	745.71	663.17				
# Disk Accesses	18.52	**17.06**	**46.80**	50.58	**148.64**	250.76	907.52	**733.25**				

Στην περίπτωση ορθογωνίων με τυχαίο μήκος πλευράς, οι δύο μέθοδοι εισαγωγής εμφανίζουν κοινά αποτελέσματα για χώρους 2 και 4 διαστάσεων. Η μέθοδος σειριακής εισαγωγής υπερτερεί σημαντικά στην περίπτωση χώρου 6 διαστάσεων παρουσιάζοντας μειωμένο πλήθος προσπελάσεων από 40% έως 53% σε σχέση με τη μαζική εισαγωγή. Για μεγαλύτερες διαστάσεις, η μαζική μέθοδος εξασφαλίζει καλύτερη απόδοση κατά 20% στις 8 διαστάσεις, έως 10% στις 10 διαστάσεις και οριακά 1.5% στις 12 διαστάσεις.

Συμπέρασμα: *οι δύο μέθοδοι έχουν παρόμοια αποτελεσματικότητα για χώρο δεδομένων έως 4 διαστάσεις ενώ για μεγαλύτερες τιμές, η σειριακή εισαγωγή υπερτερεί στις 6 διαστάσεις (40%-53%), η μαζική είναι αποδοτικότερη στις 8 και 10 διαστάσεις (20% και 10% αντίστοιχα) και η απόδοσή των δύο μεθόδων είναι εφάμιλλη στην περίπτωση του χώρου 12 διαστάσεων.*

ΚΕΦΑΛΑΙΟ 4: Συμπεράσματα και Προτάσεις

Στα πλαίσια της εργασίας υλοποιήθηκε η συγκριτική μελέτη δύο μεθόδων εισαγωγής δεδομένων στη δομή του *R*-tree,* μέσω της αποδοτικότητας σε ερωτήματα Κοντινότερου Γείτονα για πολυδιάστατους χώρους δεδομένων. Πρώτο μέλημα ήταν ο καθορισμός ενός ορίου για τη διαστατικότητα των δεδομένων. Από τα πειράματα καταλήξαμε στο συμπέρασμα ότι, το *R*-tree* μπορεί να επεξεργαστεί με αποδοτικό τρόπο έως και 8 διαστάσεις, ακόμη και στις περιπτώσεις που η αναζήτηση περιλαμβάνει μεγάλο αριθμό γειτόνων. Η υψηλή αποτελεσματικότητα μπορεί να εξασφαλιστεί και για δεδομένα έως 12 διαστάσεων, σε περιπτώσεις που το εύρος της αναζήτησης είναι μικρότερο του (1‰) των διαθέσιμων δεδομένων. Συνεπώς, η δομή μπορεί να αποτελέσει ένα χρήσιμο εργαλείο εξόρυξης γνώσης μέσω ερωτημάτων Κοντινότερου Γείτονα, όταν ικανοποιούνται οι παραπάνω συνθήκες.

Για τη σύγκριση των δύο μεθόδων, διερευνήθηκε η αποδοτικότητα του *R*-tree* ως προς τη διαδικασία κατασκευής και ως προς την εκτέλεση ερωτημάτων. Όσον αφορά το πρώτο, δείξαμε ότι η μέθοδος μαζικής εισαγωγής υπερτερεί σημαντικά της σειριακής, εξασφαλίζοντας σχεδόν βέλτιστη χρήση του διαθέσιμου αποθηκευτικού χώρου με ποσοστά 93%-98%, σε αντίθεση με τη σειριακή που κυμαίνεται στο 68%-75%. Η δομή που κατασκευάζεται είναι πολύ πιο συμπαγής και εμφανίζει μειωμένο μέγεθος 30% σε σχέση με την δομή που δημιουργείται από τη σειριακή εισαγωγή των δεδομένων. Η υπεροχή αυτή εντοπίζεται και σε όρους προσπέλασης μνήμης, αφού η μέθοδος μαζικής εισαγωγής απαιτεί μόλις το 0.5% του πλήθους των *Disk I/Os* που λαμβάνουν χώρα όταν τα δεδομένα εισάγονται σειριακά Έτσι, η μέθοδος μαζικής εισαγωγής αποτελεί ξεκάθαρα την καλύτερη επιλογή για την κατασκευή του *R*-tree*.

Όσον αφορά την απόδοση κατά την εκτέλεση ερωτημάτων, η ανάλυση των αποτελεσμάτων έγινε, (α) για κάθε μια από τις διαστάσεις του χώρου δεδομένων και (β) για καθένα από τους τύπους δεδομένων. Ο αναγνώστης μπορεί να ανατρέξει στην Ενότητα 3.3. (Μέρος 2) για αναλυτικά συμπεράσματα σε κάθε μια από τις περιπτώσεις. Μια γενικευμένη σύνοψη όλων των παραπάνω είναι η ακόλουθη:

- Οι δύο μέθοδοι εξασφαλίζουν παραπλήσια αποδοτικότητα σε χώρους δεδομένων δύο και τεσσάρων διαστάσεων, με τη σειριακή μέθοδο να υπερτερεί στη διαχείριση σημειακών αντικειμένων στις 4 διαστάσεις.

- Στην περίπτωση χώρου δεδομένων 6 διαστάσεων, η μέθοδος σειριακής εισαγωγής παρουσιάζει καθολική υπεροχή σε σχέση με τη μαζική για όλους τους τύπους δεδομένων.
- Σε χώρους δεδομένων έως 10 διαστάσεων, η σειριακή μέθοδος παρουσιάζεται καταλληλότερη στη διαχείριση "κανονικοποιημένων" χωρικών αντικειμένων (σημεία, ορθογώνια με σταθερά μήκη πλευρών) και η μαζική για αντικείμενα με πιο πολύπλοκη μορφή (ορθογώνια με τυχαία μήκη πλευρών).
- Τέλος, για χώρους 12 διαστάσεων, η υπεροχή της μεθόδου σειριακής εισαγωγής περιορίζεται στην περίπτωση που τα αντικείμενα αναπαριστώνται από σημεία στο χώρο. Η αύξηση του όγκου (ορθογώνια με σταθερά μήκη πλευρών) και της πολυπλοκότητας (ορθογώνια με τυχαία μήκη πλευρών) των χωρικών αντικειμένων, επωφελείται από την επιλογή της μαζικής εισαγωγής.

Μια παρατήρηση που απορρέει σχετικά με το εύρος της αναζήτησης είναι ότι, στις περισσότερες περιπτώσεις η αύξηση του αριθμού Κοντινότερων Γειτόνων οδήγησε σε μείωση της διαφοράς αποδοτικότητας, όταν επικρατέστερη ήταν η μέθοδος σειριακής εισαγωγής και σε αύξηση της διαφοράς, όταν ως καταλληλότερη παρουσιαζόταν η μέθοδος μαζικής εισαγωγής. Συνεπώς, ενδεχόμενη αναζήτηση που περιλαμβάνει ακόμη μεγαλύτερο πλήθος Κοντινότερων Γειτόνων, πιθανώς και να τυγχάνει παρόμοιας διαχείρισης από τις δύο μεθόδους εισαγωγής.

Από τα παραπάνω, είναι προφανές ότι θα αποτελούσε σφάλμα η προσπάθεια εξαγωγής ενός καθολικού συμπεράσματος από τη σύγκριση των δύο μεθόδων εισαγωγής. Σε κάθε περίπτωση, η επιλογή της κατάλληλης μεθόδου απορρέει από έναν συνδυασμό παραγόντων που εξετάστηκε αναλυτικά στην παρούσα εργασία. Η φύση των δεδομένων (τύπος, διάσταση, μεταβλητότητα) και του προβλήματος (απαιτήσεις σε χρόνο και μέγεθος κατασκευής ευρετηρίου), είναι αυτά που θα καθορίσουν την τελική απόφαση. Η παρούσα εργασία αποτελεί μια απόπειρα να εξαχθεί χρήσιμη πληροφορία σχετικά με την αποδοτικότητα των δύο μεθόδων κατασκευής του *R*-tree,* η οποία θα συμβάλει σε μια πιο συστημική και αιτιολογημένη επιλογή της καταλληλότερης μεθόδου εισαγωγής.

Αναφορές

Agarwal, P. and Arge, L. (2003), “CRB-Tree: An efficient indexing scheme for range aggregate queries”, Proc. of the 9th International Conference on Database Theory (ICDT’03).

Arge, L., de Berg, M., Haverkort, H., Yi, K. (2004), “The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree”, Proc. of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04), Paris, France, 347-358.

Arge, L., Hinrichs., K. H., Vahrenhold, J. and Vitter, J. S. (2002), "Efficient Bulk Operations on Dynamic R-Trees", Algorithmica 33: 104–128.

Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B. (1990), "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", Proc. ACM SIGMOD, 323-331.

Belussi, A., Faloutsos, C. (1995), “Estimating the selectivity of spatial queries using the ‘correlation’ fractal dimension”, In Proc. VLDB, 299-310.

Bentley, J. L. (1975), “Multidimensional binary search trees used for associative searching”, Communications of the ACM, 18(9):509-517.

Berchtold, S., Bohm, C. and Kriegel, H.-P. (1998), “Improving the query performance of high dimensional index structures by bulk load operations”, In Schek, H.-J., Saltor, F., Ramos, I. and Alonso, G., editors, Advances in Database Technology EDBT 1998, 6th International Conference on Extending Databases Technology, volume 1377 of Lecture Notes in Computer Science, 216-230. Springer.

Berchtold, S., Keim, D., Kriegel, H.-P. (1996), “The X-tree: An Index Structure for High-Dimensional Data”, 22nd Conf. on Very Large Databases, Bombay, India, 28-39.

Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U. (1999), "When is “Nearest Neighbor” Meaningful?", Proc. 7th International Conference on Database Theory - ICDT'99. LNCS 1540: 217–235. doi:10.1007/3-540-49257-7_15. ISBN 978-3-540-65452-0

Comer, D. E. (1979), “The ubiquitous B-tree”, ACM Computing Surveys, 11(2):121–137.

Faloutsos, C., Gaede, V. (1996), “Analysis of n-dimensional quadtrees using the Housdorff fractal dimension”, In Proc. ACM SIGMOD, Int. Conf. of the Management of Data.

Faloutsos, C., Kamel, I. (1994), “Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension”, In Proc. 13th AM SIGACT-SIGMOD-SIGART Symposium on PODS, 4-13.

Finkel, R., Bentley, J.L. (1974), "Quad Trees: A Data Structure for Retrieval on Composite Keys", Acta Informatica **4** (1): 1–9.

Fodor I. K. (2002), "A survey of dimension reduction techniques", LLNL technical report, UCRL-ID-148494.

Friedman, J. H., Bentley, J. L., and Finkel, R.A (1977), “An algorithm for finding the best matches in logarithmic expected time”, ACM Trans. Math. Software, 209-226.

Garcıa, Y. J., Lopez, M. A. and Leutenegger, S. T. (1998), “A greedy algorithm for bulk loading R-trees”, In Proc. 6th ACM Symposium on Advances in GIS, 163–164.

Greene, D. (1989), “An Implementation and Performance Analysis of Spatial Data Access Methods”, In Proc. of ICDE. 606-615.

Guttman, A. (1984), “R-trees: A dynamic index structure for spatial searching”, In Yormark, B., editor, SIGMOD '84, Proceedings of Annual Meeting, volume 14.2 of SIGMOD Record, 47-57. ACM Press.

Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E., Zimek, A. (2010), "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?", Scientific and Statistical Database Management. Lecture Notes in Computer Science. 6187. pp. 482. doi:10.1007/978-3-642-13818-8_34. ISBN 978-3-642-13817-1.

ICS 214A: Database Management Systems Winter 2004, Lecture 08:Multi-dimensional Indexing

Kamel, I. and Faloutsos, C. (1994), “Hilbert R-tree: An improved R-tree using fractals”, In Bocca, J. B., Jarke, M. and Zaniolo, C., editors, Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), 500-509. Morgan Kaufmann.

Kamel, I., Faloutsos, C. (1993), "On Packing R-trees", Proc. 2nd International Conference on Information and Knowledge Management (CKIM-93), 490-499, Arlington, VA.

Kamel, I., Khalil, M. and Kouramajian, V. (1996), “Bulk insertion in dynamic R-trees”, In Kraak, M. and Molenaar, M., editors, Proceedings of the 4th International Symposium on Spatial Data Handling (SDH '96), 3B.31-3B.42.

Katayama, N., Satoh, S. (1997), “The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries”, Proc. ACM SIGMOD Int. Conf. on Management of Data.

Lee, T., Moon, B., Lee, S. (2006), “Bulk insertion for R-trees by seeded clustering”, Data Knowl. Eng. 59(1): 86-106

Leutenegger, S. T., Lopez, M. A., Edgington, J. (1997), “STR: A simple and efficient algorithm for R-tree packing”, In A. Gray and P.-_A. Larson, editors, Proceedings of the Thirteenth International Conference on Data Engineering, 497-506. IEEE Computer Society Press.

Lin, B., Su, J. (2004), “On Bulk Loading TPR-Tree”, Mobile Data Management 2004: 114-124

Oh, K., Feng, Y., Kaneko, K., Makinouchi, A., Bae, S. (2001), "SOM-Based R*-tree for Similarity Retrieval", dasfaa, pp.0182, 7th International Conference on Database Systems for Advanced Applications (DASFAA '01).

Papadopoulos, A., Manolopoulos, Y. (2003), “Parallel bulk-loading of spatial data”, Parallel Computing 29(10): 1419-1444

Parsons et al. (2004), KDD Explorations 2004

Robinson, J. (1981), “The K-D-B tree: A search structure for large multidimensional dynamic indexes”, In Proc. SIGMOD International Conference on Management of Data, 10–18.

Roussopoulos, N, Leifker, D. (1985), "Direct Spatial Search on Pictorial Databases Using Packed R-trees", Proc. ACM SIGMOD.

Roussopoulos, N., Kelley, S., Vincent, F. (1995), “Nearest Neighbor Queries”, SIGMOD Conference 1995: 71-79

Samet, H. (1989), “The Design and Analysis Of Spatial Data Structures”, Addison-Wesley.

Samet, H. (1990), “Applications Of Spatial Data Structures, Computer Graphics, Image Processing and GIS”, Addison – Wesley.

Sellis, T., Roussopoulos, N., Faloutsos, C. (1987), "The R+ Tree: A Dynamic Index for Multidimensional Objects", Proc. 13th International Conference on Very Large Databases (VLDB-87), 507-518.

Sproull, R. F., (1991), “Refinements to Nearest Neighbor Searching in k-Dimensional Trees”, Algorithmica, 579-589.

van den Bercken, J., Seeger, B., Widmayer, P. (1996), “A Generic Approach to Bulk Loading Multidimensional Index Structures”, Technical Report 13, University of Marburg, Computer Science Department.

van den Bercken, J., Seeger, B., Widmayer, P. (1997), “A General Approach to Bulk Loading Multidimensional Index Structures”, 23rd Conf. on Very Large Databases, 1997, Athens, Greece.

White, D.A., Jain, R. (1996) “Similarity indexing with the SS-tree”, Proc. 12th Int. Conf on Data Engineering, New Orleans, LA.

Xia, T., Zhang, D. (2005), “Improving the R*-tree with Outlier Handling Techniques”, Proc. 13th annual ACM International Workshop on Geographic Information Systems, New York, 125-134.

Zhou, Q., Zhong, E.-S., Huang, Y.-H. (2009), “Parallel Computation Technique for Static R-tree Index Based on Partition Technology”, Computer Engineering 2009 35 (2): 68-69, 7, ISSN: 1000-3428: CN: 31-1289/TP

Ιστοσελίδες

TIGER/LineTM Files, 1995 Technical Documentation. Prepared by the U.S. Bureau of Census, Washington, DC, July 1996. http://www.census.gov/geo/www/tiger/tl95doc.html (διαθέσιμο 04 Οκτωβρίου 2011)

http://www.nationalarchives.gov.uk/documents/18nov08.pdf (διαθέσιμο 04 Οκτωβρίου 2011)

ICS 214A: Database Management Systems Winter 2004, Lecture 08:Multi-dimensional Indexing csis.bits-pilani.ac.in/faculty/goel/R-TREES/slides08.ppt (διαθέσιμο 04 Οκτωβρίου 2011)

http://www.statisticalengineering.com/curse_of_dimensionality.htm (διαθέσιμο 04 Οκτωβρίου 2011)

www.cs.sjsu.edu/faculty/lee/cs157/Data%20Indexing%20Presentation.ppt (διαθέσιμο 04 Οκτωβρίου 2011)

http://students.fim.uni-passau.de/~kaebisch/Proseminar/Proseminar.pdf (διαθέσιμο 04 Οκτωβρίου 2011)

ΠΑΡΑΡΤΗΜΑ Α: ΑΛΓΟΡΙΘΜΟΙ *R-TREE*

Στη συνέχεια το τμήμα που αναφέρεται στο *MBR* ενός κόμβου συμβολίζεται με *EI* και ο δείκτης στον κόμβο παιδί με *Ep* αντίστοιχα. Οι αλγόριθμοι ορίζονται στο (Guttman (1984)).

Αλγόριθμος Search

Δεδομένου ενός R tree με ρίζα T, εντοπισμός όλων των εγγραφών με ορθογώνια τα οποία επικαλύπτουν ένα ορθογώνιο αναζήτησης S.

S1 *[Αναζήτηση υπο-δένδρων]*

Εάν το T δεν είναι φύλλο

Για κάθε εγγραφή E, εάν το EI επικαλύπτει το S, κάλεσε την Search για το δένδρο με ρίζα τον κόμβο στον οποίο δείχνει το Ep.

Αλλιώς

S2 *[Αναζήτηση φύλλων]*

Για κάθε εγγραφή E, εάν το EI επικαλύπτει το S, επέστρεψε το E.

Τέλος

Αλγόριθμος Insert

Εισαγωγή μιας νέας εγγραφής E στη δομή του R tree

I1 *[Εύρεση της θέσης που θα αποθηκευτεί η νέα εγγραφή]*

Κάλεσε την ChooseLeaf για την επιλογή ενός κόμβου-φύλλου L στο οποίο θα αποθηκευτεί η εγγραφή E.

I2 *[Προσθήκη της εγγραφής στον κόμβο-φύλλο]*

Εάν το L έχει χώρο για μια ακόμη εγγραφή

Αποθήκευσε το E.

Αλλιώς

Κάλεσε την SplitNode για τη δημιουργία των L και LL που θα περιλαμβάνουν το E και όλες τις εγγραφές που υπήρχαν στο L.

Τέλος

I3 *[Διάδοση των αλλαγών προς τα πάνω]*

Εάν δεν έγινε διάσπαση

Κάλεσε την AdjustTree με όρισμα το L.

Αλλιώς

Κάλεσε την AdjustTree με όρισμα το LL.

Τέλος

I4 *[Αύξηση του ύψους του δένδρου]*

Εάν το I3 οδήγησε σε διάσπαση της ρίζας

Δημιούργησε μια νέα ρίζα με παιδιά του δύο νέους κόμβους που προέκυψαν μετά τη διάσπαση.

Τέλος

Αλγόριθμος ChooseLeaf

Επιλογή του κόμβου-φύλλου που θα αποθηκευτεί η νέα εγγραφή E

CL1 *[Αρχικοποίηση]*

Θέσε το N να είναι η ρίζα.

CL2 *[Έλεγχος για φύλλο]*

Εάν το N είναι φύλλο

Επέστρεψε το N.

Αλλιώς

CL3 *[Επιλογή υπο-δένδρου]*

Επέλεξε την εγγραφή στο N της οποίας το ορθογώνιο απαιτεί τη μικρότερη αύξηση επιφάνειας για να συμπεριλάβει το νέο δεδομένο. Σε περίπτωση ισοπαλίας (δύο κατάλληλα ορθογώνια) επέλεξε το ορθογώνιο με τη μικρότερη επιφάνεια

Τέλος

CL4 *[Καθοδική διάσχιση του δένδρου έως ότου οδηγηθούμε σε κόμβο-φύλλο]*

Θέσε το N να είναι ο κόμβος παιδί στον οποίον "δείχνει" ο δείκτης της επιλεγμένης εγγραφής και επανέλαβε από το CL2

Αλγόριθμος AdjustTree

Ανοδική διάσχιση του δένδρου από ένα κόμβο-φύλλο L προς τη ρίζα, ρυθμίζοντας τα Ορθογώνια Κάλυψης των εσωτερικών κόμβων και μεταδίδοντας τις διασπάσεις κόμβων όποτε είναι απαραίτητο.

AT1 *[Αρχικοποίηση]*

Θέσε το N να είναι το φύλλο L.

Εάν το L διασπάστηκε προηγουμένως

Θέσε το NN να είναι το δεύτερο φύλλο που προέκυψε από τη διάσπαση.

Τέλος

AT2 *[Έλεγχος τερματισμού]*

Εάν το N είναι η ρίζα

Σταμάτησε.

Αλλιώς

AT3 [Προσαρμογή του Ορθογωνίου Κάλυψης του κόμβου-γονέα]

Θεώρησε το P να είναι ο κόμβος-γονέας του N και το E_N η εγγραφή του N στο P. Ρύθμισε το $E_N I$ έτσι ώστε να περιλαμβάνει ακριβώς όλα τα ορθογώνια των εγγραφών του N.

AT4 [Διάδοση της διάσπασης κόμβων προς τα πάνω]

Εάν το N διασπάστηκε προηγουμένως

Δημιούργησε μια νέα εγγραφή E_{NN} με το $E_{NN} p$ να δείχνει στο NN και το $E_{NN} I$ να περιλαμβάνει όλα τα ορθογώνια στο NN.

Εάν υπάρχει διαθέσιμος χώρος

Πρόσθεσε την εγγραφή E_{NN} στο P.

Αλλιώς

Κάλεσε την SplitNode για τη δημιουργία των P και PP που θα περιλαμβάνουν την εγγραφή E_{NN} και όλες τις εγγραφές που υπήρχαν στο P.

Τέλος

Τέλος

AT5 [Άνοδος στο επόμενο επίπεδο του δένδρου]

Εάν δεν προκλήθηκε διάσπαση

Θέσε το N να είναι ο κόμβος P και επανέλαβε από το AT2.

Αλλιώς

Θέσε το N να είναι ο κόμβος P και το NN να είναι ο κόμβος PP και επανέλαβε από το AT2.

Τέλος

Τέλος

<u>Αλγόριθμος QuadraticSplit</u>

Διαίρεση ενός συνόλου M+1 εγγραφών σε δύο ομάδες

QS1 [Επιλογή της πρώτης εγγραφής για κάθε ομάδα]

Κάλεσε την PickSeeds για την επιλογή δύο εγγραφών που θα είναι τα πρώτα στοιχεία των ομάδων. Αντιστοίχισε κάθε μια από αυτές σε μια ομάδα.

QS2 [Έλεγχος ολοκλήρωσης]

Εάν όλες οι εγγραφές έχουν αντιστοιχιστεί

Σταμάτησε.

[Αντιστοίχιση εγγραφών στην ομάδα που χρειάζεται όλες τις εγγραφές ώστε να περιλαμβάνει τον ελάχιστο αριθμό m εγγραφών]

Αλλιώς, εάν μια ομάδα περιλαμβάνει τόσες λίγες εγγραφές ούτως ώστε θα πρέπει να αντιστοιχιστούν σε αυτή όλες οι υπόλοιπες για να φτάσει τον ελάχιστο αριθμό εγγραφών m

Αντιστοίχισε τες σε αυτή την ομάδα και σταμάτησε.

Τέλος

QS3 *[Επιλογή της εγγραφής που θα αντιστοιχιστεί σε ομάδα]*

Κάλεσε την PickNext για την επιλογή της επόμενης εγγραφής που θα αντιστοιχιστεί σε μια ομάδα. Πρόσθεσε την εγγραφή στην ομάδα που απαιτεί την ελάχιστη αύξηση της επιφάνειας του Ορθογωνίου Κάλυψης της, ώστε να συμπεριλάβει την εγγραφή. Στην περίπτωση της ισοπαλίας, επέλεξε την ομάδα με τη μικρότερη επιφάνεια και κατόπιν αυτή με τις λιγότερες εγγραφές. Επανέλαβε από το QS2.

<u>Αλγόριθμος PickSeeds</u>

Επιλογή δύο εγγραφών που θα αποτελέσουν τα πρώτα στοιχεία των ομάδων

PS1 *[Υπολογισμός της "αναποτελεσματικότητας - κακής απόδοσης" (inefficiency) εάν ομαδοποιηθούν δύο εγγραφές]*

Για κάθε ζεύγος εγγραφών E_1 *και* E_2

Δημιούργησε το ορθογώνιο J που περικλείει τα ορθογώνια E_1I , E_2I *και υπολόγισε το* $d = area(J) - area(E_1I) - area(E_2I)$.

PS2 *[Επιλογή του χειρότερου ζεύγους]*

Επέλεξε το ζεύγος ορθογωνίων με τη μεγαλύτερη τιμή του d.

<u>Αλγόριθμος PickNext</u>

Επιλογή μιας εκ των υπολοίπων εγγραφών για αντιστοίχιση σε κάποια ομάδα

PN1 *[Προσδιορισμός του κόστους αντιστοίχισης κάθε εγγραφής σε κάθε μια από τις ομάδες]*

Για κάθε εγγραφή E που δεν έχει αντιστοιχιστεί σε κάποια ομάδα

Υπολόγισε το d_1 *ως την αύξηση της επιφάνειας του Ορθογωνίου Κάλυψης της Ομάδας 1, που απαιτείται για να συμπεριλάβει το ορθογώνιο EI . Υπολόγισε αντίστοιχα το* d_2 *για την Ομάδα 2.*

PN2 *[Εύρεση της καταλληλότερης εγγραφής για αντιστοίχιση με κάποια ομάδα]*

Επέλεξε την εγγραφή με τη μεγαλύτερη διαφορά μεταξύ των d_1 *και* d_2.

Αλγόριθμος Delete

Διαγραφή της εγγραφής E από ένα R tree.

D1 *[Εύρεση του κόμβου που περιλαμβάνει την εγγραφή]*

Κάλεσε τη FindLeaf για τον εντοπισμό του κόμβου-φύλλου L που περιλαμβάνει την εγγραφή E.

[Έλεγχος τερματισμού]

Εάν η εγγραφή δε βρεθεί.

Σταμάτησε.

Αλλιώς

D2 *[Διαγραφή εγγραφής]*

Αφαίρεσε την εγγραφή E από τον κόμβο-φύλλο L.

D3 *[Διάδοση των αλλαγών]*

Κάλεσε την CondenseTree, με όρισμα το L.

D4 *[Ελάττωση του ύψους του δένδρου]*

Εάν η ρίζα έχει μονάχα ένα κόμβο-παιδί

Κάνε τον κόμβο-παιδί να είναι η νέα ρίζα του δένδρου.

Τέλος

Τέλος

Αλγόριθμος FindLeaf

Δεδομένου ενός R tree με ρίζα το T, εύρεση του κόμβου-φύλλου που περιλαμβάνει την εγγραφή E.

FL1 *[Αναζήτηση υπο-δένδρων]*

Εάν το T δεν είναι φύλλο

Για κάθε εγγραφή F στο T

Εξέτασε εάν το ορθογώνιο FI επικαλύπτει το ορθογώνιο EI. Για κάθε μια τέτοια εγγραφή κάλεσε τη FindLeaf με όρισμα το δένδρο που έχει ρίζα τον κόμβο στον οποίο δείχνει το Fp έως ότου να βρεθεί η εγγραφή E, ή να έχουν εξεταστεί όλες οι εγγραφές.

Τέλος

FL2 *[Αναζήτηση της εγγραφής σε ένα κόμβο-φύλλο]*

Εάν το T είναι φύλλο

Για κάθε εγγραφή του

Εξέτασε εάν είναι η E. Εάν βρεθεί, επέστρεψε το T.

Τέλος

Αλγόριθμος CondenseTree

Δεδομένου ενός κόμβου-φύλλου L από το οποίο διαγράφηκε κάποια από τις εγγραφές του, κατάργηση του κόμβου εάν έχει μη επαρκές πλήθος εγγραφών και επανατοποθέτηση των εγγραφών του. Διάδοση της κατάργησης του κόμβου προς τα πάνω εάν είναι απαραίτητο. Προσαρμογή όλων των Ορθογωνίων Κάλυψης στη διαδρομή προς τη ρίζα, κάνοντάς τα μικρότερα εάν είναι δυνατό.

CT1 [Αρχικοποίηση]

Θέσε το N να είναι ο κόμβος-φύλλο L. Όρισε ως Q το σύνολο των κόμβων που έχουν καταργηθεί και θέσε το να είναι κενό.

CT2 [Εύρεση της εγγραφής γονέα]

Εάν το N είναι η ρίζα

Πήγαινε στο CT6.

Αλλιώς

Θεώρησε το P να είναι ο κόμβος-γονέας του N και το E_N να είναι η εγγραφή του κόμβου N στον κόμβο P.

CT3 [Κατάργησε τον κόμβο με μη επαρκές πλήθος εγγραφών]

Εάν ο κόμβος N περιλαμβάνει λιγότερες από m εγγραφές

Διέγραψε την εγγραφή E_N από τον κόμβο P και πρόσθεσε τον κόμβο N στο σύνολο Q.

Τέλος

CT4 [Προσαρμογή του Ορθογωνίου Κάλυψης]

Εάν ο κόμβος N δεν καταργήθηκε

Προσάρμοσε το ορθογώνιο $E_N I$ ώστε να περιλαμβάνει ακριβώς όλες τις εγγραφές του κόμβου N.

Τέλος

CT5 [Άνοδος στο επόμενο επίπεδο του δένδρου]

Θέσε το N να είναι ο κόμβος P και επανέλαβε από το CT2.

Τέλος

CT6 [Επαν-εισαγωγή των ορφανών εγγραφών]

Επαν-εισήγαγε όλες τις εγγραφές των κόμβων που του συνόλου Q. Οι εγγραφές από κόμβους-φύλλα που έχουν καταργηθεί, επαν-εισάγονται στα φύλλα του δένδρου όπως περιγράφεται από τον αλγόριθμο Insert, αλλά οι εγγραφές από κόμβους ανώτερων επιπέδων πρέπει να τοποθετηθούν σε κόμβους που βρίσκονται υψηλότερα στο δένδρο, έτσι ώστε τα φύλλα των υπο-δένδρων των κόμβων αυτών, να βρίσκονται στο ίδιο επίπεδο με τα φύλλα του κυρίως δένδρου.

ΠΑΡΑΡΤΗΜΑ Β: ΑΛΓΟΡΙΘΜΟΙ *R*-TREE*

Οι αλγόριθμοι ορίζονται στο (Beckmann et. al. (1990)).

Αλγόριθμος ChooseSubtree

Επιλογή του κόμβου-φύλλου που θα αποθηκευτεί η νέα εγγραφή E

CS1 [Αρχικοποίηση]

Θέσε το N να είναι η ρίζα

CS2 [Έλεγχος για φύλλο]

Εάν το N είναι φύλλο

Επέστρεψε το N

Αλλιώς

[Επιλογή υπο-δένδρου]

Εάν οι δείκτες στα παιδία του κόμβου N "δείχνουν" σε φύλλα

[προσδιόρισε το ελάχιστο κόστος επικάλυψης]

Επέλεξε την εγγραφή του N της οποίας το ορθογώνιο απαιτεί την ελάχιστη αύξηση επικάλυψης για να συμπεριλάβει το νέο ορθογώνιο δεδομένων. Σε περίπτωση ισοπαλίας επέλεξε την εγγραφή με το ορθογώνιο που απαιτεί την ελάχιστη αύξηση της επιφάνειας και κατόπιν την εγγραφή με το ορθογώνιο που έχει τη μικρότερη επιφάνεια

Εάν οι δείκτες στα παιδιά του κόμβου N δεν "δείχνουν" σε φύλλα

[προσδιόρισε το ελάχιστο κόστος επιφάνειας]

Επέλεξε την εγγραφή του N με το ορθογώνιο που απαιτεί την ελάχιστη αύξηση επιφάνειας για να συμπεριλάβει το νέο ορθογώνιο δεδομένων. Σε περίπτωση ισοπαλίας επέλεξε την εγγραφή της οποίας το ορθογώνιο έχει τη μικρότερη επιφάνεια

Τέλος

CS3 [Καθοδική διάσχιση του δένδρου έως ότου οδηγηθούμε σε κόμβο-φύλλο]

Θέσε το N να είναι ο κόμβος παιδί στον οποίο "δείχνει" ο δείκτης της επιλεγμένης εγγραφής και επανέλαβε από το CS2

[προσδιόρισε το σχεδόν ελάχιστο κόστος επικάλυψης]

Ταξινόμησε τα ορθογώνια στο N κατά αύξουσα σειρά με βάση την απαιτούμενη αύξηση επιφάνειας ώστε να συμπεριλάβουν το νέο ορθογώνιο δεδομένων

Έστω A είναι η ομάδα των πρώτων p εγγραφών

Από τις εγγραφές στην ομάδα A, θεωρώντας όλες τις εγγραφές στο N, επέλεξε την εγγραφή της οποίας το ορθογώνιο απαιτεί τη μικρότερη αύξηση επικάλυψης. Σε περίπτωση ισοπαλίας, λειτούργησε όπως προηγούμενα.

<u>Αλγόριθμος Split</u>

Διαίρεση ενός συνόλου M+1 εγγραφών σε δύο ομάδες

S1 [Επιλογή άξονα διάσπασης]

Κάλεσε την ChooseSplitAxis

S2 [Επιλογή της κατανομής των δύο ομάδων κατά μήκος του άξονα διάσπασης]

Κάλεσε την ChooseSplitIndex

S3 [Κατανομή των ομάδων]

Κατένειμε τις εγγραφές σε δύο ομάδες

<u>Αλγόριθμος ChooseSplitAxis</u>

Επιλογή του άξονα διάσπασης, κάθετα στον οποίο θα γίνει η κατανομή των εγγραφών

CSA1 [Υπολογισμός των κατανομών]

Για κάθε άξονα

Ταξινόμησε τις εγγραφές κατά την κατώτατη και κατόπιν κατά την ανώτατη τιμή των ορθογωνίων τους και προσδιόρισε όλες τις κατανομές όπως περιγράφεται παραπάνω. Υπολόγισε το S, ως το άθροισμα όλων των τιμών περιθωρίου για όλες τις διαφορετικές κατανομές.

CSA2 [Επιλογή άξονα διάσπασης]

Επέλεξε τον άξονα με το ελάχιστο S ως άξονα διάσπασης

<u>Αλγόριθμος ChooseSplitIndex</u>

Κατανομή των M+1 εγγραφών στις δύο ομάδες, βάσει του άξονα διάσπασης

CSI1 [Επιλογή της κατανομής]

Κατά μήκος του επιλεγμένου άξονα διάσπασης, επέλεξε την κατανομή με την ελάχιστη τιμή επικάλυψης. Σε περίπτωση ισοπαλίας, επέλεξε την κατανομή με την ελάχιστη τιμή επιφάνειας.

Αλγόριθμος InsertData

Εισαγωγή δεδομένων στη δομή του R tree*

ID1 *[Αρχικοποίηση της διαδικασίας εισαγωγής]*

Κάλεσε την Insert ξεκινώντας με το επίπεδο των φύλλων ως παράμετρο, για την εισαγωγή ενός νέου ορθογωνίου δεδομένων.

Αλγόριθμος Insert

Εισαγωγή μιας νέας εγγραφής E στη δομή του R tree*

I1 *[Επιλογή του υπο-δένδρου που θα γίνει η εισαγωγή]*

Κάλεσε την ChooseSubtree, με το επίπεδο ως παράμετρο, για να βρεθεί ο κατάλληλος κόμβος N, στο οποίο θα τοποθετηθεί η νέα εγγραφή E.

I2 *[Έλεγχος πληρότητας του κόμβου]*

Εάν ο κόμβος N έχει λιγότερες από M εγγραφές

Τοποθέτησε την εγγραφή E στον κόμβο N.

Εάν ο κόμβος N έχει M εγγραφές

Κάλεσε την OverflowTreatment με το επίπεδο του κόμβου N ως παράμετρο. [για επανατοποθέτηση ή διάσπαση]

I3 *[Διάδοση της διάσπασης κόμβων προς τα πάνω]*

Εάν η κλήση της OverflowTreatment πραγματοποίησε μια διάσπαση

Επέκτεινε την OverflowTreatment προς τα πάνω εάν είναι απαραίτητο.

Εάν η κλήση της OverflowTreatment προκάλεσε διάσπαση της ρίζας

Δημιούργησε μια νέα ρίζα.

I4 *[Προσαρμογή των ορθογωνίων κάλυψης της διαδρομής που ακολουθήθηκε]*

Προσάρμοσε όλα τα ορθογώνια κάλυψης (covering rectangles) της διαδρομής εισαγωγής, έτσι ώστε να είναι πλαίσια ελάχιστης οριοθέτησης που θα περιλαμβάνουν τα ορθογώνια παιδία τους.

Αλγόριθμος OverflowTreatment

Διαχείριση ενός υπερχειλισμένου κόμβου

OT1 *[Επαν-εισαγωγή των εγγραφών ή διάσπαση του κόμβου]*

Εάν το επίπεδο δεν είναι το επίπεδο ρίζας και αυτή είναι η πρώτη κλήση του αλγορίθμου στο δεδομένο επίπεδο κατά την εισαγωγή ενός ορθογωνίου δεδομένων, τότε

Κάλεσε την ReInsert.

Αλλιώς

Κάλεσε την Split.

Τέλος

<u>Αλγόριθμος ReInsert</u>

Επαν-εισαγωγή των εγγραφών μετά την περίπτωση υπερχείλισης ενός κόμβου της δομής

RI1 *[Υπολογισμός της απόστασης των ορθογωνίων των εγγραφών από το κέντρο του Περιβάλλοντος Ορθογωνίου του κόμβου]*

Για όλες τις M+1 εγγραφές ενός κόμβου N, υπολόγισε την απόσταση μεταξύ των κέντρων των ορθογωνίων τους και του κέντρου του Περιβάλλοντος Ορθογωνίου του κόμβου N.

RI2 *[Ταξινόμηση των εγγραφών]*

Ταξινόμησε τις εγγραφές σε φθίνουσα σειρά των αποστάσεων που υπολογίστηκαν στο RI1.

RI3 *[Αφαίρεση εγγραφών]*

Αφαίρεσε τις πρώτες p εγγραφές του κόμβου N και προσάρμοσε το Περιβάλλον Ορθογώνιό του.

RI4 *[Επαν-εισαγωγή των ταξινομημένων εγγραφών]*

Στην ταξινόμηση, που πραγματοποιήθηκε στο RI2, ξεκινώντας με τη μέγιστη απόσταση (μακρινή επανατοποθέτηση) ή την ελάχιστη απόσταση (κοντινή επανατοποθέτηση), κάλεσε την Insert για να επανεισάγεις τις εγγραφές.