ΠΟΛΥΔΙΑΣΤΑΤΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Πειραματική ανάλυση M-Tree σε 2 και 3 διαστάσεις

Dimitris Samoladas - 6196
samoladas@ceid.upatras.gr

1. INSERT

Η insert περιλαμβάνει δύο μεγάλα μέρη:

1. Την εισαγωγή του στοιχείου στο κατάλληλο φύλλο ανάλογα με την απόσταση.
2. Διάσπαση αναδρομικά σε περίπτωση υπερχείλισης του κόμβου.

Η εισαγωγή του στοιχείου καθώς και η διάσπαση γίνονται ταυτόχρονα, δηλαδή το στοιχείο εισάγεται με αναδρομή και καθώς επιστρέφουμε από την αναδρομή από κόμβο σε κόμβο ελέγχουμε αν απαιτείται διάσπαση ή όχι. Σε κάθε επίπεδο αναδρομής έχουμε δύο περιπτώσεις:

1. Απαιτείται διάσπαση και συνεπώς επιστρέφουμε στο επόμενο επίπεδο αναδρομής τα δύο νέα Routing Objects που δημιουργήθηκαν από την συνάρτηση split.
2. Δεν χρειάστηκε διάσπαση και επομένως επιστρέφεται η ανανεωμένη ακτίνα λόγω εισαγωγής νέου στοιχείου.

Για την διάσπαση του κόμβου που υπερχείλισε ακολουθούμε την εξής στρατηγική:

1. Promotion: Επιλογή των σημείων που θα αποτελέσουν τα Routing Objects ή αλλιώς κέντρα των νέων κόμβων. Επιλέχθηκαν τα σημεία που απέχουν περισσότερο μεταξύ τους.
2. Partition: Ο διαμοιρασμός των στοιχείων στους δύο νέους κόμβους γίνεται σειριακά. Κάθε σημείο εισάγεται στον κόμβο που απέχει την μικρότερη απόσταση.

2. REMOVE

Η remove αποτελείται από δύο μέρη:

1. Εύρεση του στοιχείου προς αφαίρεση διατρέχοντας τους κόμβους με βάση την απόσταση του στοιχείου από το κέντρο τους.
2. Αφαίρεση κόμβου αναδρομικά σε περίπτωση υποχείλισης.

Οι κόμβοι που αφαιρούνται ωστόσο μπορεί να έχουν και άλλα στοιχεία μέσα που δεν τα έχουμε αφαιρέσει. Για αυτό το λόγο αποθηκεύουμε τους κόμβους που αφαιρούμε και στο τέλος του remove επανεισάγουμε τα στοιχεία τους στο δέντρο με χρήση της insert.

3. CIRCULAR RANGE SEARCH

Για την αναζήτηση των σημείων που βρίσκονται σε μια ακτίνα από το query σημείο χρησιμοποιούμε τις ήδη υπολογισμένες αποστάσεις που υπάρχουν για να απαλείψουμε υποδέντρα που αποκλείεται να έχουν σημεία που ανήκουν στην ακτίνα αναζήτησης. Συγκεκριμένα γνωρίζουμε τις εξής αποστάσεις:

1. Απόσταση των παιδιών από τον πατέρα e.DistanceFromParent.
2. Απόσταση του query σημείου από τον πατέρα (έχει υπολογιστεί στο προηγούμενο επίπεδο αναδρομής) DistanceQP.

Επομένως έχουμε ότι η απόσταση του query σημείου από κάθε παιδί του κόμβου είναι ίση με DistanceQP – e.DistanceFromParent. Για τα παιδιά των φύλλων που δεν είναι κόμβοι και συνεπώς δεν έχουν ακτίνα αρκεί η παραπάνω απόσταση να είναι μικρότερη της ακτίνας αναζήτησης. Για τους εσωτερικούς κόμβους που τα παιδιά τους είναι και αυτά κόμβοι, η παραπάνω απόσταση πρέπει να είναι μικρότερη της ακτίνας αναζήτησης συν την ακτίνα του παιδιού. Η παραπάνω διαδικασία μας γλιτώνει πολλούς υπολογισμούς αποστάσεων και είναι ιδιαίτερα σημαντική όταν η συνάρτηση απόστασης είναι αρκετά χρονοβόρα. Αν ο παραπάνω υπολογισμός βγαίνει εντός ορίων τότε συνεχίζουμε στο υποδέντρο υπολογίζοντας τις εκάστοτε αποστάσεις.

5. k-NN Search

Για τον υπολογισμό των k κοντινότερων γειτόνων ενός query σημείου χρησιμοποιούμε ένα priority queue, που αποθηκεύει ταξινομημένους με βάση την απόσταση από το query σημείο τους κόμβους που θα διατρέξουμε, και ένα array μεγέθους k στο οποίο αποθηκεύουμε τους γείτονες ως ζεύγη με την απόσταση τους από το query σημείο. Το array είναι ταξινομημένο ώστε το k-οστό σημείο να έχει την μέγιστη απόσταση από το query σημείο.

Ουσιαστικά ξεκινάμε από μια άπειρη μέγιστη απόσταση και όσο διατρέχουμε τους κόμβους αυτή η απόσταση μειώνεται. Για να το κάνουμε αυτό χρησιμοποιούμε δύο είδη αποστάσεων. Την dmin ελάχιστη απόσταση του query σημείου από τον κόμβο, που ορίζεται ως η απόσταση του query σημείου από το κέντρο του κόμβου μείον την ακτίνα του κόμβου, και την dmax μέγιστη απόσταση του query σημείου από τον κόμβο, που ορίζεται ως η απόσταση του query σημείου από το κέντρο του κόμβου συν την ακτίνα του κόμβου. Η διαδικασία είναι η εξής:

1. Για κάθε κόμβο που βγάζουμε από το priority queue υπολογίζουμε την dmin του για κάθε παιδί του. Αν η dmin του εκάστοτε παιδιού είναι μικρότερη από την μέγιστη απόσταση που έχουμε ως τώρα τότε τοποθετούμε το παιδί στο priority queue μαζί με την dmin του και υπολογίζουμε την dmax του.
2. Αν η dmax του είναι μικρότερη της μέγιστης απόστασης τότε ανανεώνουμε την μέγιστη απόσταση και αφαιρούμε από το priority queue όσους κόμβους έχουν dmin μεγαλύτερο από τη νέα μέγιστη απόσταση.

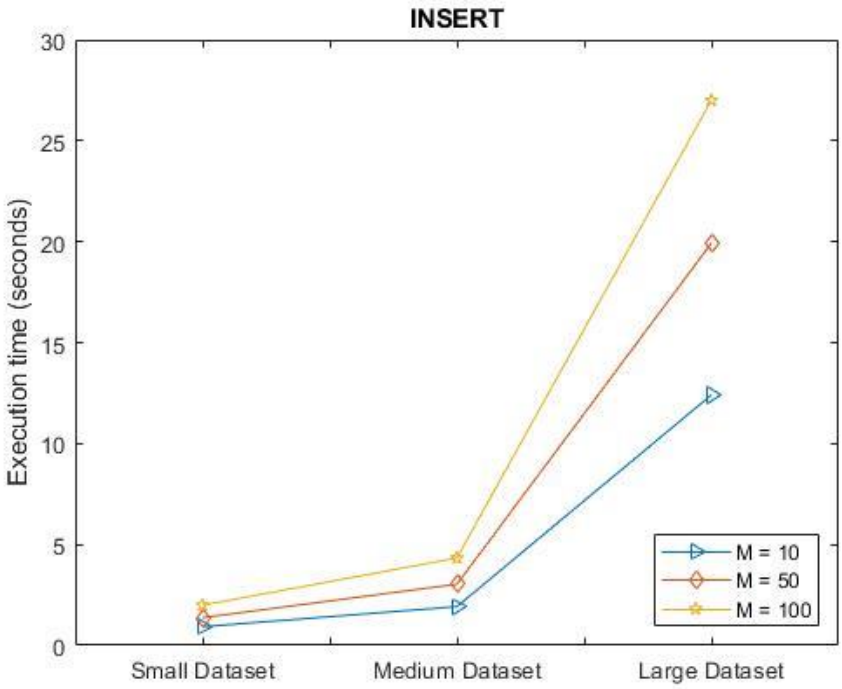
Πως διαχειριζόμαστε την μέγιστη απόσταση: Η μέγιστη απόσταση ξεκινάει από το άπειρο και είναι ουσιαστικά η απόσταση του k-οστού στοιχείου του array που αποθηκεύει τους γείτονες. Κάθε φορά που μειώνουμε την μέγιστη απόσταση ουσιαστικά αυτό που κάνουμε είναι να προσθέτουμε ένα καινούργιο στοιχείο στο array, να αφαιρούμε το k-οστό και να επιστρέφουμε την απόσταση του νέου k-οστού στοιχείου ως νέα μέγιστη απόσταση.

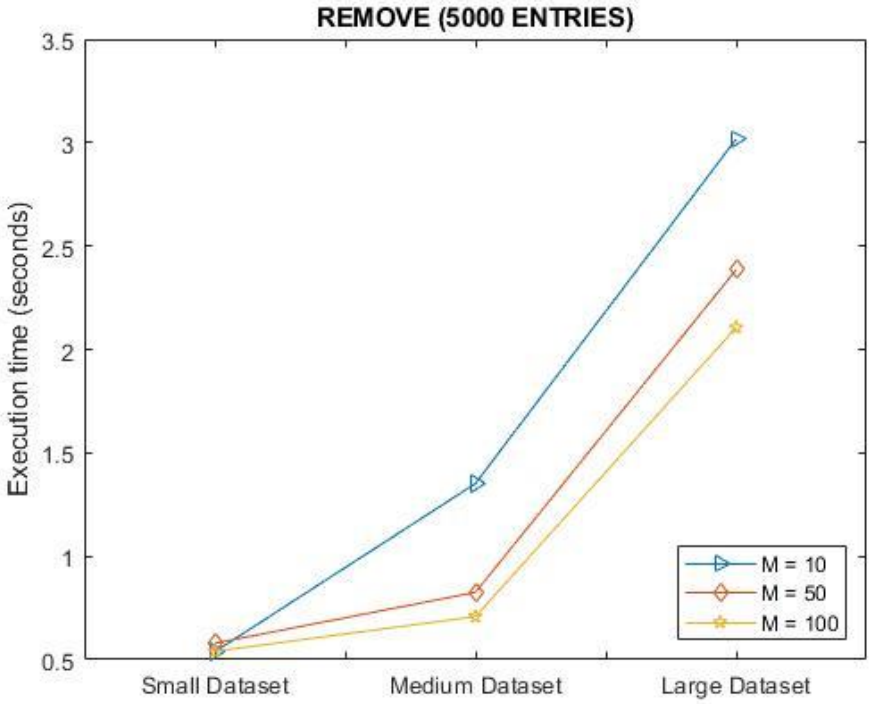
Η διαδικασία ολοκληρώνεται όταν αδειάσει το priority queue.

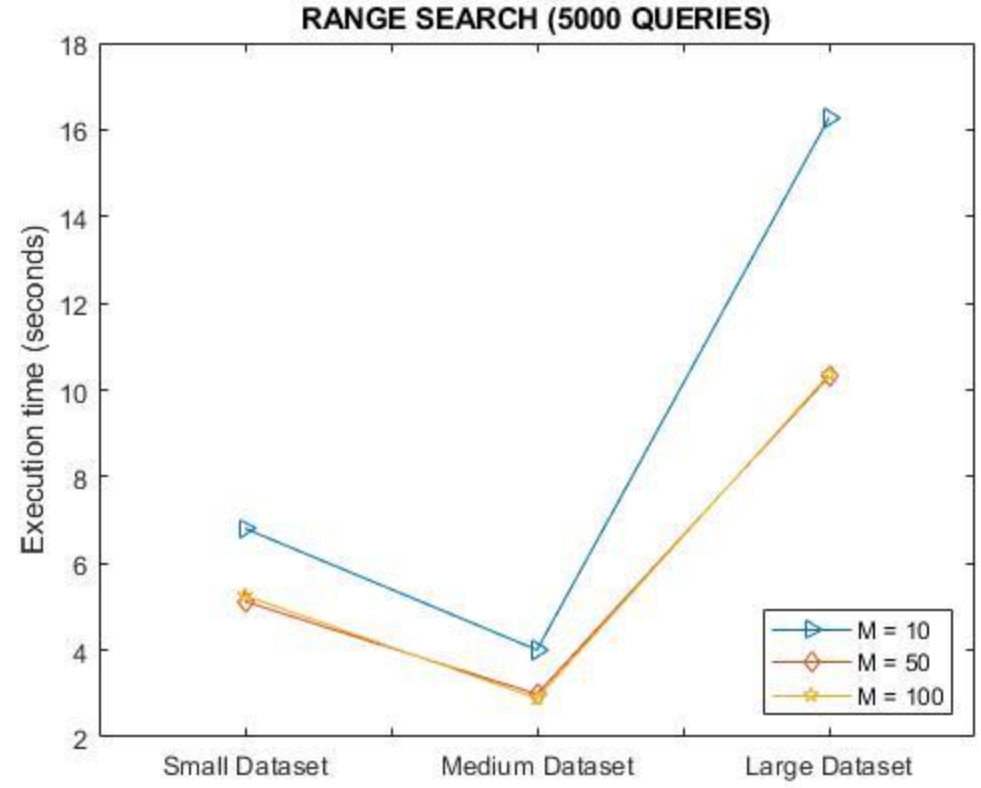
6. Benchmarks

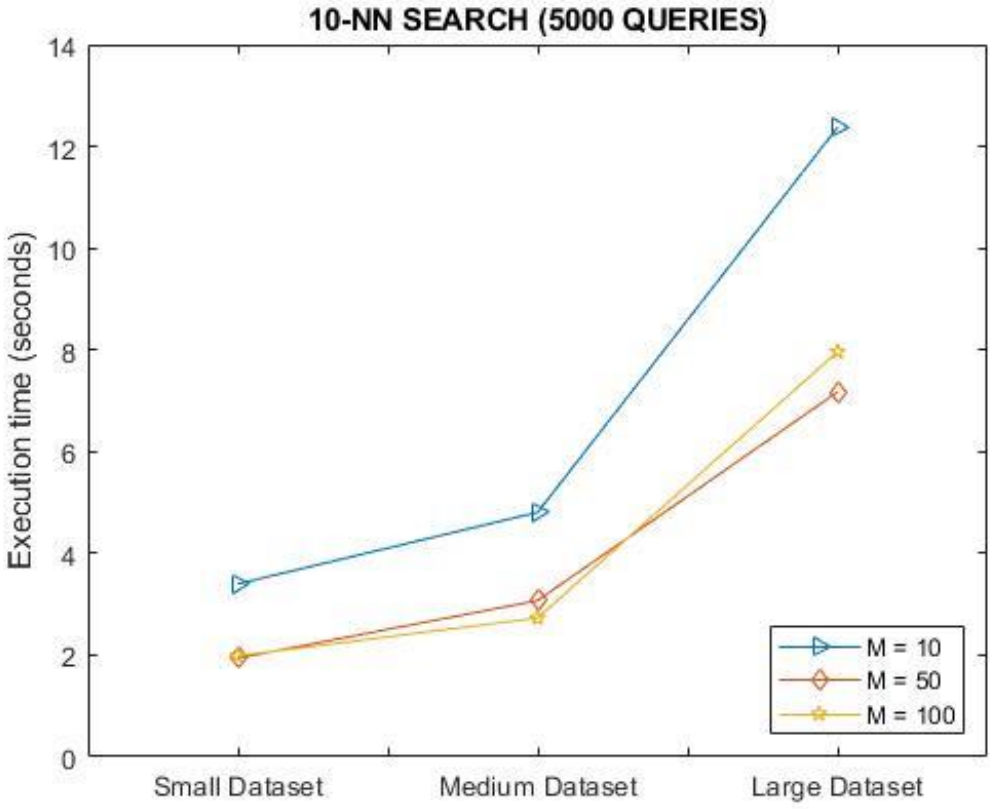
Για τα benchmarks χρησιμοποιήθηκαν συνθετικά datasets που δημιουργήθηκαν με χρήση python και της συνάρτησης uniform για την παραγωγή των σημείων. Για τα dataset έχουμε:

Small Dataset: 500.000 σημεία, Medium Dataset: 1.000.000 σημεία, Large Dataset: 5.000.000 σημεία





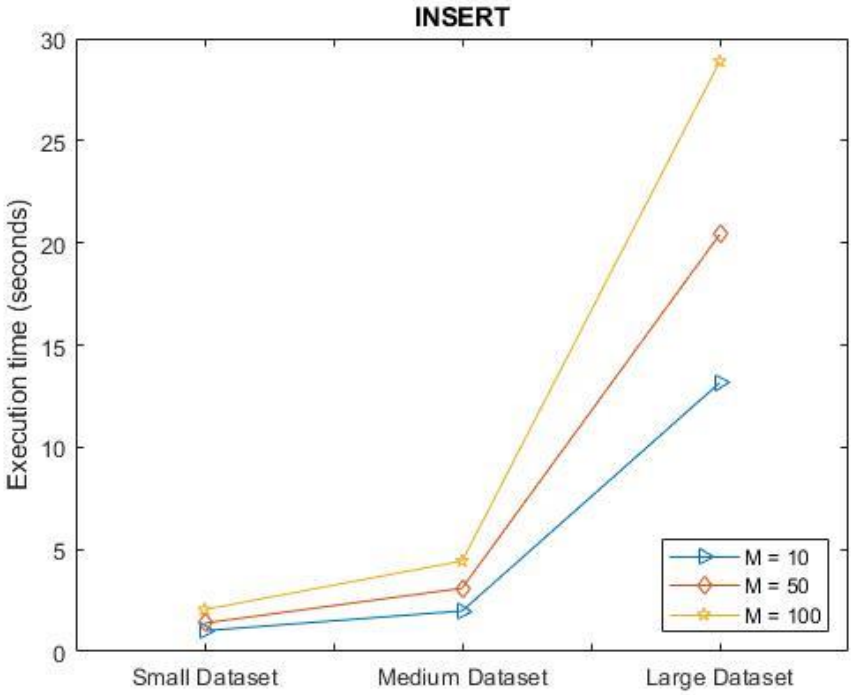

RANGE SEARCH (5000 QUERIES)
Execution time (seconds)
18
16
14
12
10
8
6
4
2
Small Dataset
Medium Dataset
Large Dataset
M = 10
M = 50
M = 100

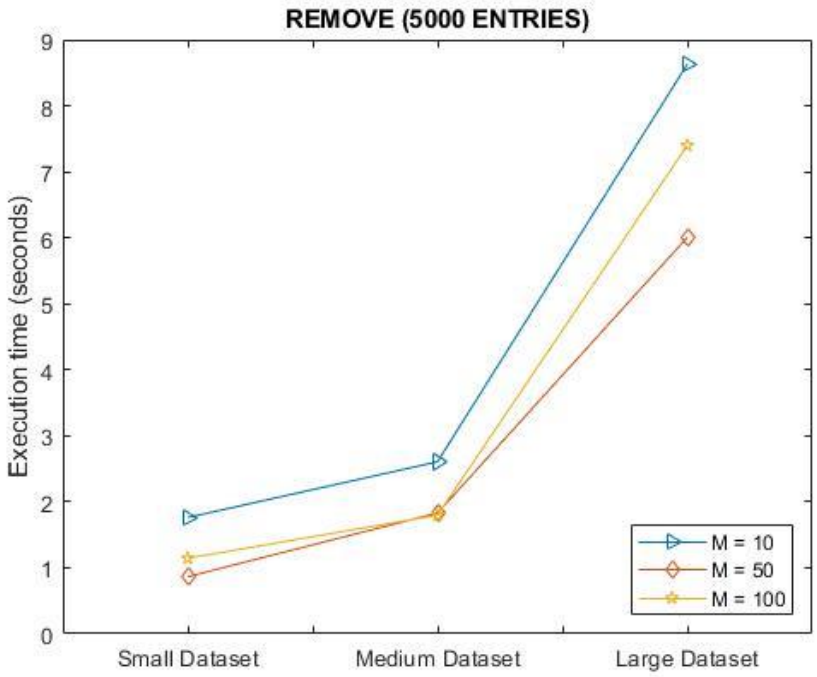

10-NN SEARCH (5000 QUERIES)
Execution time (seconds)
14
12
10
8
6
4
2
0
Small Dataset
Medium Dataset
Large Dataset
M = 10
M = 50
M = 100

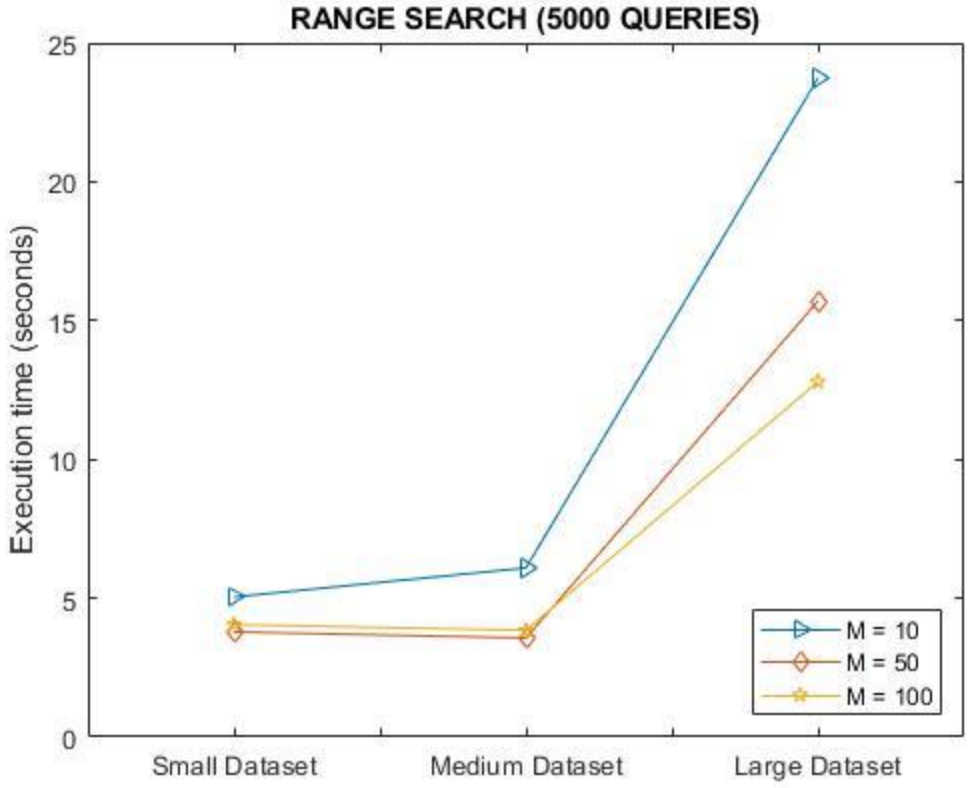
7. M-Tree σε 3 διαστάσεις

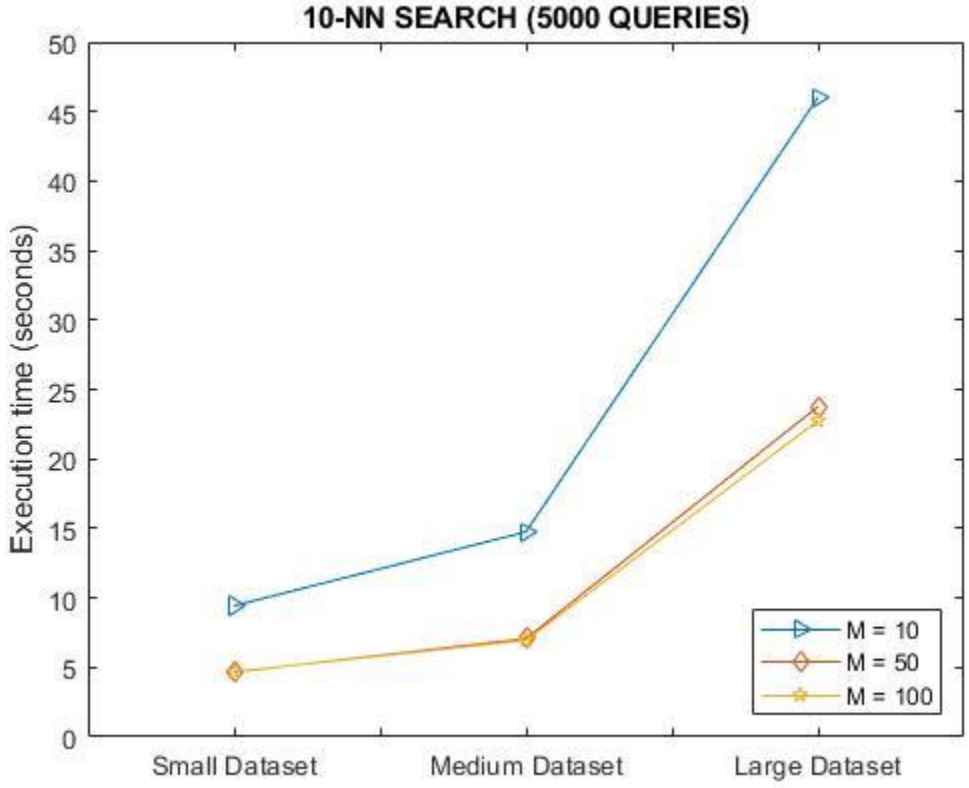
Για περισσότερες διαστάσεις το μόνο που αλλάζει είναι η γεωμετρία των σημείων και η συνάρτηση απόστασης. Όλες οι βασικές πράξεις υλοποιούνται το ίδιο.

Τα benchmarks για το M-Tree σε 3 διαστάσεις.






RANGE SEARCH (5000 QUERIES)
Execution time (seconds)
0
5
10
15
20
25
Small Dataset
Medium Dataset
Large Dataset
M = 10
M = 50
M = 100


10-NN SEARCH (5000 QUERIES)
Execution time (seconds)
0
5
10
15
20
25
30
35
40
45
50
Small Dataset
Medium Dataset
Large Dataset
M = 10
M = 50
M = 100