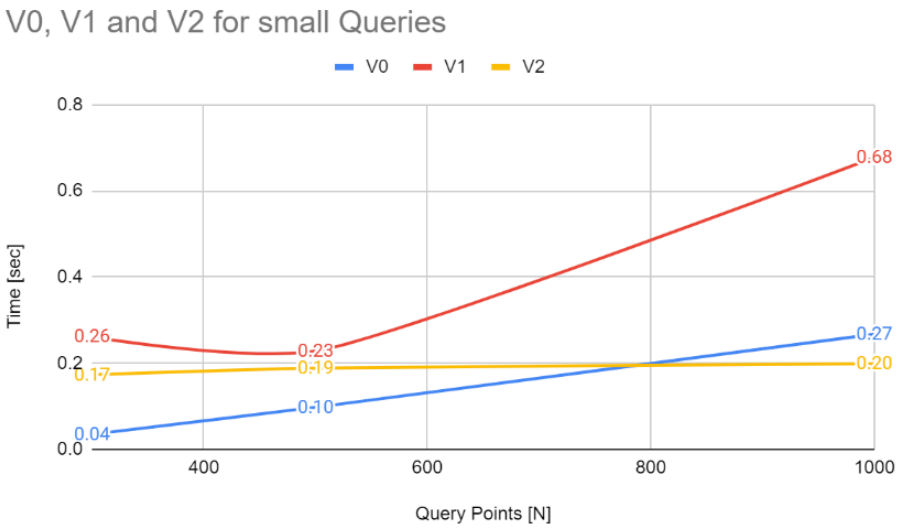
Κωνσταντίνος Χατζής
ΑΕΜ: 9256
github.com/kostascc/PDS-Project-2

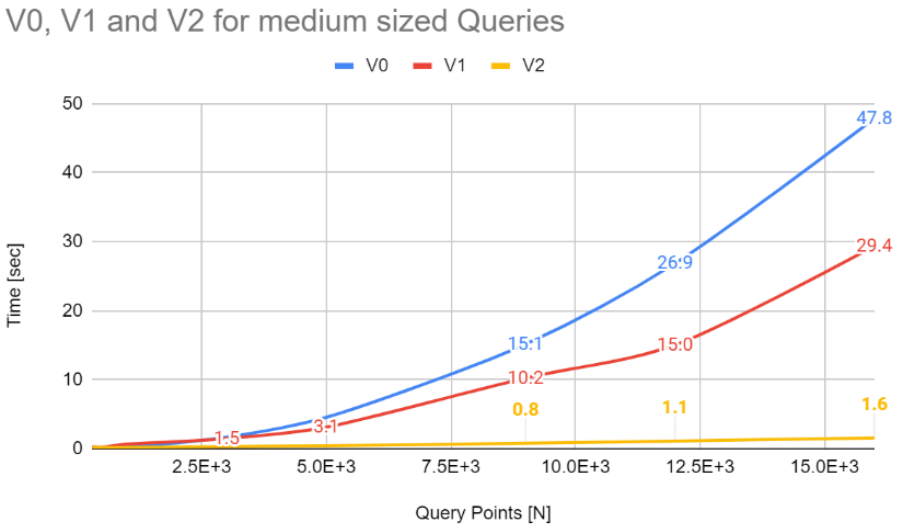
Κώδικας

Στη δεύτερη εργασία ασχολούμαστε με *k-NN* αναζήτηση σε μεγάλα σετ δεδομένων.

Στην έκδοση *V0* γίνεται χρήση πινάκων και υπολογισμός της απόστασης όλων των σημείων μεταξύ τους. Για την εύρεση των k γειτονικών σημείων εκτελείται *Merge Sort* σε κάθε σύνολο *m* αποστάσεων. Το *OpenBLAS* χρησιμοποιείται για ταχύτερες πράξεις μεταξύ πινάκων.

Στη συνέχεια, στη *V1* γίνεται χρήση του *OpenMPI* για διαμοιρασμό του *Corpus Set* σε περισσότερα *nodes*. Συγκεκριμένα, κάθε ένα από τα nodes κρατά ένα ξεχωριστό κομμάτι του *Corpus Set* και το χρησιμοποιεί αρχικά και ως *Query*. Αφού το χρησιμοποιήσει για την εύρεση των k γειτόνων, το προωθεί στο επόμενο node (ωρολογιακά), ενώ την ίδια στιγμή λαμβάνει ένα νέο Query Set από το προηγούμενο node (αντιωρολογιακά). Το *Query* που λήφθηκε χρησιμοποιείται για την εύρεση των κοντινότερων γειτόνων, και προωθείται ξανά. Η κίνηση αυτή συνεχίζεται εώς ότου κάθε *node* έχει λάβει και προωθήσει όλα τα *Query Sets* κυκλικά. Τέλος, τα *nodes* ωρολογιακά – ξεκινώντας από αυτό που έχει το μεγαλύτερο *rank* – στέλνουν τους k-γείτονες που αντιστοιχούν στο *Corpus Set* τους στο επόμενο node, όπου εκεί γίνεται επιλογή ξανά των κοντινότερων *k*-γειτόνων (μεταξύ των δύο *set*). Κάθε *node* με $rank = i$ γνωρίζει τα καλύτερα "$k * m$" αποτελέσματα των *node* με $rank \geq i$. Η κίνηση αυτή συνεχίζεται ώσπου το *node* με $rank = 0$ (στο εξής Master) λάβει τα καλύτερα k-αποτελέσματα όλων των υπολοίπων *node*, τα συγκρίνει με τα δικά του και δώσει ένα ολοκληρωμένο σύνολο k-NN. Σημειώνεται ότι τα υπόλοιπα nodes επιστρέφουν μόνο τα τοπικά τους αποτελέσματα.

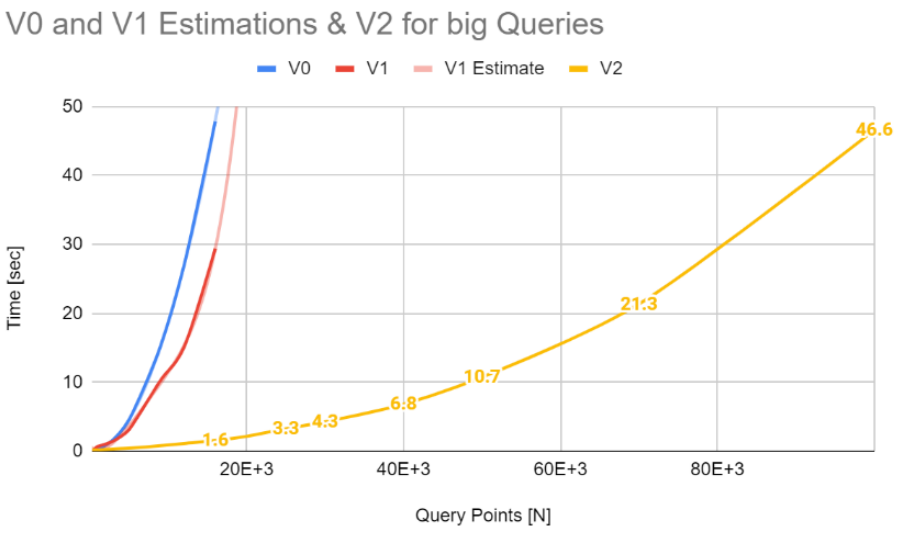




Τέλος, στη *V2* γίνεται χρήση των *Vantage Point Tree*. Εφόσον κάθε *node* ξεκινήσει από ένα *Corpus Set* (όπως στις *V0, V1*), χτίζει ένα ξεχωριστό *VP Tree*. Το δέντρο αυτό θα χρησιμοποιηθεί αντίστοιχα του V1 σε ότι αφορά τη διακίνηση του *Query Set* καθώς και των τελικών αποτελεσμάτων. Αντί για *Merge Sort* όπως στις *V0* & *V1*, στη *V2* γίνεται χρήση της *Quick Select*.

Επίσης, οποιαδήποτε αποστολή ή λήψη πληροφορίας μέσω του *OpenMPI* γίνεται ασύγχρονα, όπως και διάφορα σημεία των *V0, V1* & *V2* έχουν παραλληλιστεί με *OpenCilk* για καλύτερη χρήση πόρων.

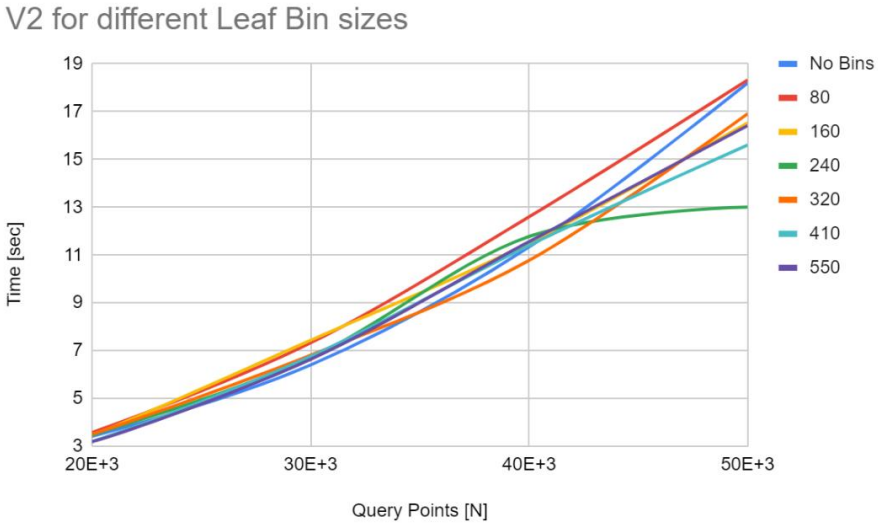
Εκτέλεση για διάφορα μεγέθη δεδομένων



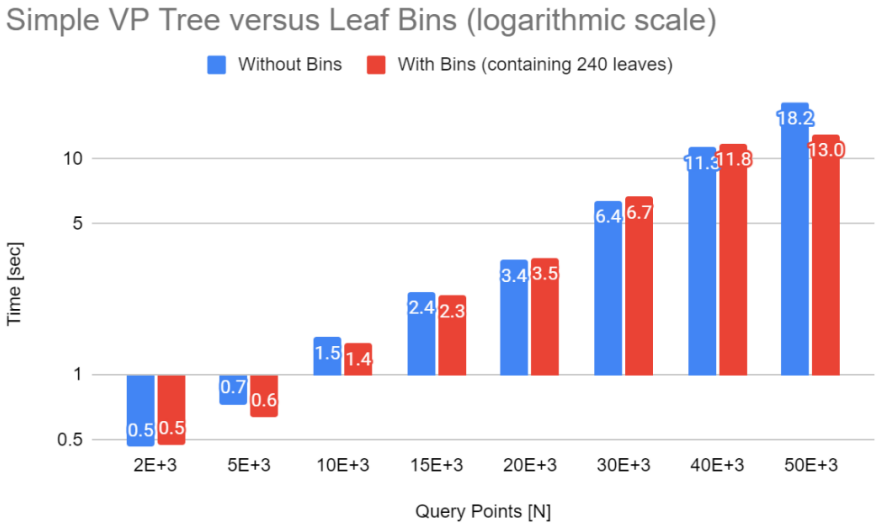
Για την παρουσίαση της εκτέλεσης των τριών αλγορίθμων, έγινε εκτέλεση των *V0, V1, V2* σε τοπικό υπολογιστή, με *8-Core i5 CPU,* και όριο μνήμης τα $6GB$. Τα μεγέθη πινάκων που επιλέχθηκαν για τις *V0* & *V1* είναι τέτοια ώστε να μην υπάρξει υπερκατανάλωση μνήμης (το μεγαλύτερο *Set* δεν ξεπερνούσε οριακά την ελεύθερη μνήμη). Για το διαμοιρασμό της διεργασίας στα *V1* & *V2* έγινε εκτέλεση τους μέσω *OpenMPI* με 2 θεωρητικά *nodes* (στον ίδιο υπολογιστή, με διαμοιρασμένους πόρους). Οι εκτελέσεις έγιναν για μικρό αριθμό γειτόνων *(*$K = 40$*)* και μέτριο αριθμό διαστάσεων *(*$D = 8$*)*.

Η εκτέλεση των *V0* & *V1* (με μπλέ και κόκκινο αντίστοιχα) μας δείχνει ότι η *Merge Sort* ήταν ξεκάθαρα μια κακή επιλογή για την διαλογή γειτόνων, έναντι της *Quick Select* που χρησιμοποιήθηκε στη *V2*. Επίσης, η V2 παρά τον μεγάλο όγκο δεδομένων, κατά την εκτέλεση του *Set* με $100 * 10^3$ *Corpus Points* κατέλαβε μόλις $200MB$ έναντι των περίπου $6GB$ που κατέλαβαν οι άλλες εκδόσεις σε μόλις $15 * 10^3$ *Points*.

Επιλογή βάθους Leaf-Bin



Για την δημιουργία της V2, τέθηκε ως ερώτημα κατά πόσο η παύση της κατασκευής *VP Tree* όταν απομένει συγκεκριμένος αριθμός (B) φύλλων μπορεί να βελτιώσει το χρόνο εκτέλεσης. Στα σχήματα παρουσιάζονται οι εκτελέσεις του V2 για διαφορετικά μεγέθη *Corpus Set* και τιμές B. Από δοκιμές, καθώς και από τον πίνακα που παρατίθεται, επιλέχθηκε $B = 250$ για όλες τις υπόλοιπες δοκιμές. Στο τελευταίο σχήμα της σελίδας φαίνεται ξεκάθαρα η βελτίωση του χρόνου εκτέλεσης για την τιμή B σε σχέση με εκτέλεση χωρίς χρήση *Leaf-Bins*.

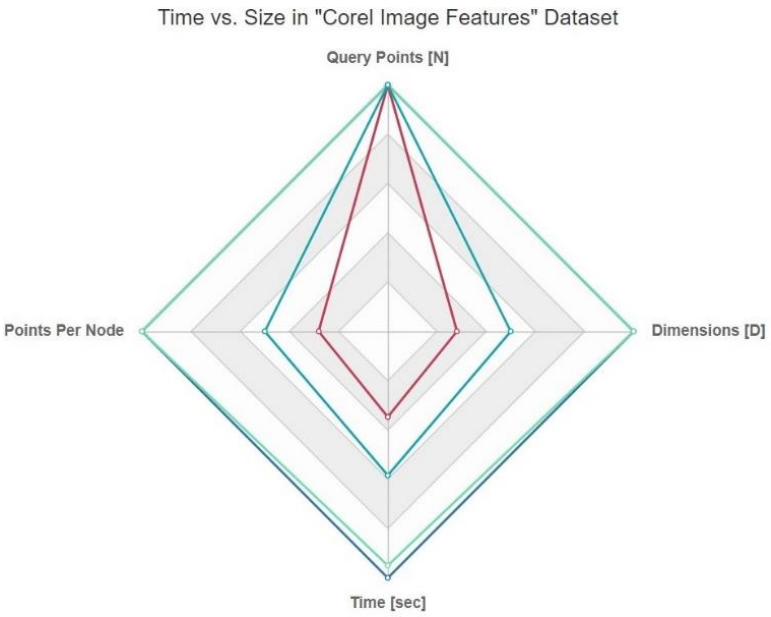


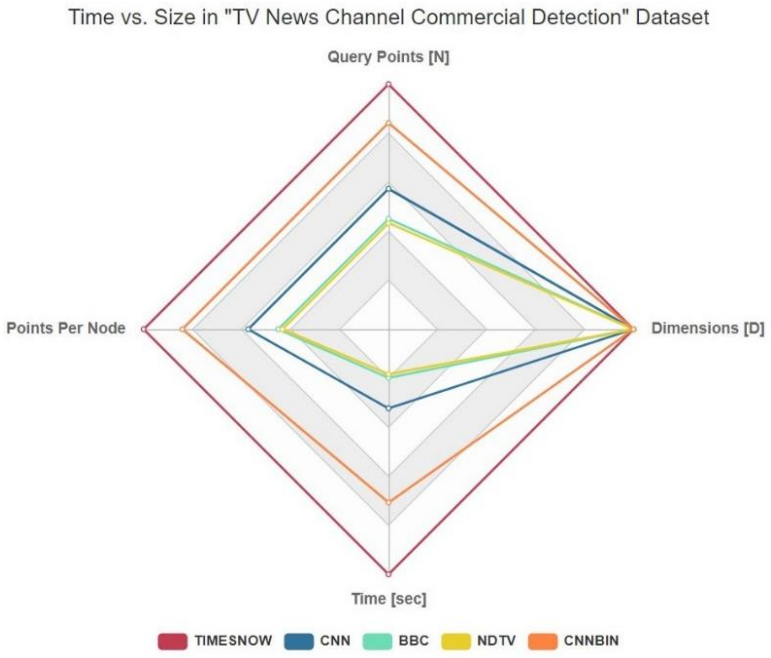
Εκτέλεση μεγάλων Set

Στα τελευταία δυο σχήματα παρουσιάζονται οι χρόνοι εκτέλεσης των πινάκων που δόθηκαν ως βάση σύγκρισης. Η εισαγωγή των δεδομένων έγινε με μετατροπή των αρχείων σε *Matrix-Market* μορφή εξωτερικά, και οι χρόνοι προέρχονται από τη *V2*, σε *2 Tasks* της συστοιχίας με *10 CPU Cores* το καθένα.

Παρατηρείται εδώ, από τα 2 διαγράμματα *Radar*, ότι ενώ η σχέση διαστάσεων – διάρκειας εκτέλεσης είναι γραμμική (με σταθερό πλήθος σημείων) στο πάνω σχήμα της σελίδας, αυτή του πλήθους των σημείων με τη διάρκεια δεν είναι (στο δεύτερο σχήμα). Αυτό συμβαίνει καθώς στη δεύτερη περίπτωση μεταβάλλεται το μέγεθος του Corpus και του Query ταυτόχρονα, έναντι της πρώτης περίπτωσης όπου οι διαφορετικές διαστάσεις χρησιμοποιούνται για κάθε *Query Point* που «σκανάρει» το δέντρο.

Τέλος, παρατίθεται ένας πίνακας με τη διάρκεια εκτέλεσης των παραπάνω. Η εκτέλεση έγινε σε ένα φυσικό node, με διαχωρισμό σε *2 Tasks* μέσω του *OpenMPI*, και συνολικά *20 Cores*.





Set	***Subset***	***Query Points [N]***	***Dimensions [D]***	***Time [sec]***
TV Ads	TIMESNOW	39325	4125	809.94
	CNN	22545	4125	261.46
	BBC	17720	4125	160.48
	NDTV	17051	4125	148.38
	CNNBIN	33117	4125	572.62
MiniBooNE	MiniBooNE	130066	50	73.83
FMA Music	Features	106560	518	738.37
Corel	ColorMoments	68040	9	5.45
	ColorHistogram	68040	32	15.7
	LayoutHistogram	68040	32	14.91
	CoocTexture	68040	16	9.17
(Random)	Million-by-12	1000000	12	1441.17