ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Παράλληλη Επεξεργασία Χωρικών Ερωτημάτων

Διπλωματική Εργασία

Μπακίρης Εμμανουήλ

Επιβλέπων: Βασιλακόπουλος Μιχαήλ

Βόλος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Παράλληλη Επεξεργασία Χωρικών Ερωτημάτων

Διπλωματική Εργασία

Μπακίρης Εμμανουήλ

Επιβλέπων: Βασιλακόπουλος Μιχαήλ

Βόλος 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Parallel Processing of Spatial Queries

Diploma Thesis

Bakiris Emmanouil

Supervisor: Michael Vassilakopoulos

Volos 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Βασιλακόπουλος Μιχαήλ**
Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Αθανάσιος Φεύγας**
ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Γεώργιος Θάνος**
ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 15-2-2021

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον αναπληρωτή καθηγητή κ. Μιχαήλ Βασιλακόπουλο, καθώς επίσης, και το στέλεχος της Επιτροπής Ερευνών κ. Πολυχρόνη Βελέτζα. Η συμβολή τους στην επιτυχή εκπόνηση της διπλωματικής μου εργασίας ήταν ουσιαστική και καθοριστική, αφού οποτεδήποτε χρειάστηκα βοήθεια, η ανταπόκριση τους ήταν άμεση και πολύτιμη. Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου, η οποία βρισκόταν δίπλα μου και με στήριζε, τόσο κατά τη διάρκεια των σπουδών μου, όσο και κατά τη διάρκεια της μελέτης και εκπόνησης της παρούσας διπλωματικής εργασίας. Τέλος, δεν θα μπορούσα να μην ευχαριστήσω τους συμφοιτητές μου, καθώς καθόλη τη διάρκεια των σπουδών μου, υπήρχε πάντα πνεύμα συνεργασίας και ομαδικότητας, που ακόμα και σε δύσκολες στιγμές, βρίσκαμε τη λύση και προχωρούσαμε μαζί.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Μπακίρης Εμμανουήλ

15-2-2021

Περίληψη

Η παράλληλη επεξεργασία χωρικών ερωτημάτων αναφέρεται στην αξιοποίηση παράλληλων αρχιτεκτονικών υπολογισμού για την επεξεργασία και την απάντηση ερωτημάτων τα οποία περιλαμβάνουν χωρικά δεδομένα (δεδομένα με συντεταγμένες στο χώρο). Στην παρούσα διπλωματική εργασία, μελετήθηκαν διάφοροι αλγόριθμοι σε κεντρικά αλλά και σε παράλληλα, κατανεμημένα συστήματα οι οποίοι επεξεργάζονται και απαντούν χωρικά ερωτήματα αποτελεσματικά και με γνώμονα την ταχύτητα εκτέλεσης. Ο σκοπός είναι η υλοποίηση ενός αλγορίθμου που απαντάει στο χωρικό ερώτημα των "Κ πλησιέστερων ζευγών" σε αρχιτεκτονική μιας μονάδας επεξεργασίας γραφικών (GPU). Έγινε ανάπτυξη και υλοποίηση ενός Παράλληλου Αλγορίθμου και σύγκριση του με σειριακούς που να απαντούν στο ίδιο ερώτημα. Τέλος, παρουσιάζεται λεπτομερής πειραματική αξιολόγηση αναλύωντας τις βασικές παραμέτρους που επηρεάζουν τον χρόνο εκτέλεσης του ερωτήματος και μελλοντικές προοπτικές έρευνας για την περαιτέρω διερεύνηση αλγορίθμων σε μία μονάδα επεξεργασίας γραφικών.

Abstract

The ”Parallel processing of spatial queries” is referred to the use of parallel architectures, for processing and answering queries that consist of spatial data (data related to space). In this thesis, different algorithms based on central, paralllel and distributed systems that process efficiently spatial queries have been the main research point. The purpose of this project is the implementation of an algorithm that answers the “K closest pairs query” in a Graphics Processing Unit (GPU). A parallel algorithm has been developed and implemented and a comparison with serial ones that answers the same query has been made. Finally, analytical experimentation is presented that analyze the basic parameters which infect the execution time of the query and future prospects of research in order to give the opportunity to extend algorithms that proccess spatial queries in a graphics processing unit.

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κατάλογος πινάκων

Συντομογραφίες

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
κ.ά.	και άλλα
π.χ.	παραδείγματος χάρη
GPU	Graphics Processing Unit
CPU	Central Processing Unit
API	Application Programming Interface

Κεφάλαιο 1

Εισαγωγή

Τα χωρικά ερωτήματα απασχολούν όλο και περισσότερο την επιστημονική κοινότητα. Οι εφαρμογές τους είναι ευρείες, όπως για παράδειγμα στον αστικό σχεδιασμό, στον προγραμματισμό διαδρομών αστικών μεταφορών, στην διαχείριση των πόρων μιας επιχείρησης, στην προώθηση προϊόντων με βάση πρόσφατες τοποθεσίες, στην αρχαιολογία καθώς και στην περιβαλλοντική μοντελοποίηση [6]. Αυτές είναι μόνο μερικές από τις εφαρμογές τους, στις οποίες προστίθενται και αυτές που χρησιμοποιεί καθημερινά ο άνθρωπος για την ικανοποίηση των διαφόρων αναγκών του. Έτσι, η επεξεργασία των χωρικών ερωτημάτων αποτελεσματικά και γρήγορα κρίνεται απαραίτητη, βελτιώνοντας την απόδοση αλλά και την ποιότητα των αποτελεσμάτων των προαναφερθέντων εφαρμογών.

Ένα χωρικό ερώτημα είναι ένας ειδικός τύπος ερωτήματος σε μία βάση δεδομένων που ονομάζεται Χωρική Βάση Δεδομένων (Spatial Database). Μια τέτοια βάση δεδομένων έχει σχεδιαστεί να αποθηκεύει και να διαχειρίζεται γεωγραφικές πληροφορίες και χωρικά δεδομένα [7]. Τα χωρικά ερωτήματα διαφέρουν από τα γνωστά SQL ερωτήματα, με διάφορους τρόπους. Δύο από τους πιο σημαντικούς είναι ότι επιτρέπουν την χρήση γεωμετρικών τύπων δεδομένων όπως σημεία, γραμμές και πολύγωνα και ότι αυτά τα ερωτήματα μελετούν τη χωρική σχέση μεταξύ αυτών των γεωμετρικών στοιχείων [8].

Μέχρι σήμερα όμως, λίγες είναι οι αναφορές στη βιβλιογραφία, εκτέλεσης χωρικών ερωτημάτων σε παράλληλες αρχιτεκτονικές υπολογισμού. Οι περισσότερες υλοποιήσεις αλγορίθμων που να απαντούν σε χωρικά ερωτήματα έχουν εφαρμοστεί αποδοτικά σε Κεντρικές Μονάδες Επεξεργασίας (CPU). Μια παράλληλη αρχιτεκτονική αποτελεί την Μονάδα Επεξεργασίας Γραφικών (GPU), που είναι σχεδιασμένη με διαφορετική φιλοσοφία από αυτήν της CPU. Μια κεντρική μονάδα επεξεργασίας έχει γενικότερη εφαρμογή και η απόδοση στη-

ρίζεται στην μέγιστη αξιοποίηση και βελτιστοποίηση του κάθε πυρήνα. Αποτελείται από ένα μικρό αριθμό πυρήνων που εκτελούν ατομικές διεργασίες σε πολύ μικρό χρόνο. Από την άλλη, οι μονάδες επεξεργασίας γραφικών αρχικά σχεδιάστηκαν για να επιταχύνουν τις διεργασίες που αφορούν τρισδιάτατα δεδομένα και γραφικά. Με το πέρασμα των χρόνων, αυτές οι μονάδες έγιναν πιο ευέλικτες και ευκολότερες στον προγραμματισμό. Η βασική τους χρήση ακόμα προορίζεται για τα γραφικά παιχνιδιών που έχουν υψηλές απαιτήσεις, όμως η εξέλιξη τους αρχίζει να ευνοεί ένα πιο γενικό σκοπό καθώς μπορούν να αξιοποιηθούν για παράλληλο προγραμματισμό σε ένα όλο και αυξανόμενο εύρος εφαρμογών [9].

Επομένως, οι GPUs μπορούν να αξιοποιηθούν κατάλληλα, για να βελτιώσουν την ταχύτητα απάντησης των χωρικών ερωτημάτων, συνεισφέροντας σε μια συνολικά καλύτερη απόδοση. Η GPU (Μονάδα Επεξεργασίας Γραφικών) είναι μια παράλληλη αρχιτεκτονική που προσφέρει πολλές δυνατότητες. Οι πολλοί πυρήνες που περιέχει, καθιστούν εργασίες που περιλαμβάνουν εξαιρετικά πολλούς υπολογισμούς, κατάλληλες για εκτέλεση. Τα χωρικά ερωτημάτα όπως των "Κ πλησιέστερων γειτόνων", "Κ πλησιέστερων ζευγών" είναι μερικά που απαιτούν πλήθος υπολογισμών για να απαντηθούν και η ιδέα να υλοποιηθούν σε παράλληλες αρχιτεκτονικές και ειδικότερα μιας GPU, αποτελεί αντικείμενο μελέτης που αξίζει την προσοχή.

1.1 Αντικείμενο της διπλωματικής εργασίας

Στην παρούσα διπλωματική εργασία γίνεται ανάπτυξη και υλοποίηση ενός αλγορίθμου σε κάρτα γραφικών GPU, που απαντάει το ερώτημα των "Κ πλησιέστερων ζευγών". Στην βιβλιογραφία υπάρχει πλήθος υλοποιήσεων σε Μονάδες Κεντρικής Επεξεργασίας (CPU) και σε παράλληλα κατανεμημένα συστήματα (Spatial Hadoop, Apache Spark κ.ά.). Παρόλα αυτά, οι υλοποιήσεις σε κάρτες γραφικών είναι λιγοστές. Γι'αυτό και ο σχεδιασμός ενός αλγορίθμου σε κάρτα γραφικών, που να απαντάει σε αυτό το ερώτημα, θα μπορούσε να αποτελέσει σημαντική συνεισφορά σε αυτόν τον τομέα, δίνοντας ταυτόχρονα την δυνατότητα μιας βάσης για την περαιτέρω εξέλιξη διαφορετικών αλγορίθμων σε GPU, βελτιώνοντας όλο και περισσότερο την ταχύτητα απόκρισης του ερωτήματος.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν σειριακοί και παράλληλοι αλγόριθμοι σε κατανεμημένα συστήματα παρόμοιων χωρικών ερωτημάτων, με σκοπό την εύρεση αλγορίθμου που να χρησιμοποιεί ίδιες ή παρόμοιες τεχνικές που να μπορούν να εφαρμοστούν σε μια παράλληλη εκτέλεση σε κάρτα γραφικών.

2. Αναπτύχθηκε και υλοποιήθηκε ένας παράλληλος αλγόριθμος σε κάρτα γραφικών GPU που δίνει απάντηση στο ερώτημα των "Κ πλησιέστερων ζευγών".

3. Αξιολογήθηκε η επίδοση του αλγορίθμου, συγκρίνοντας την ταχύτητα εκτέλεσής του με σειριακούς που απαντούν στο ίδιο ερώτημα.

4. Παρουσιάζεται λεπτομερής πειραματική αξιολόγηση, με σκοπό τη μελλοντική χρήση των συμπερασμάτων που αναδεικνύει, στην υλοποίηση καλύτερων αλγορίθμων σε παρόμοιες παράλληλες αρχιτεκτονικές.

1.2 Οργάνωση του τόμου

Στην εργασία αυτή στο 2ο Κεφάλαιο δίνεται ένα θεωρητικό υπόβαθρο, στο 3ο Κεφάλαιο παρουσιάζονται εργασίες σχετικές με το αντικείμενο της διπλωματικής και στο 4ο Κεφάλαιο αναλύεται ο αλγόριθμος που υλοποιήθηκε και οι τεχνικές που χρησιμοποιήθηκαν. Τέλος, στο 5ο Κεφάλαιο προβάλλεται η πειραματική αξιολόγηση του αλγορίθμου και στο 6ο Κεφάλαιο συγκεντρώνονται τα συμπεράσματα της έρευνας με μελλοντικές προτάσεις.

Κεφάλαιο 2

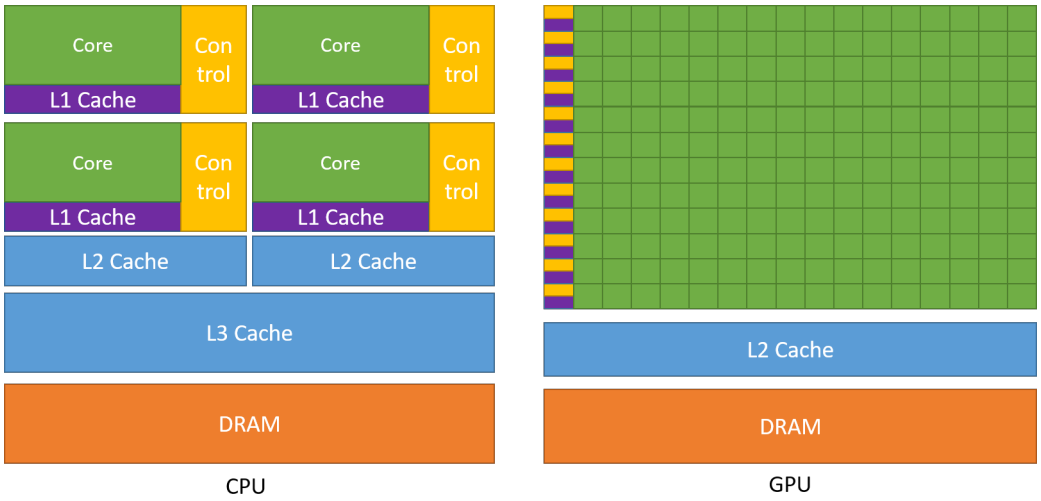
Θεωρητικό Υπόβαθρο

2.1 Εισαγωγή

Η μονάδα επεξεργασίας γραφικών (GPU) είναι μία αρχιτεκτονική που ευνοεί την παραλληλοποίηση.Παρέχει τη δυνατότητα εκτέλεσης πολλών εντολών ταυτόχρονα, σε συνδυασμό με έναν υψηλό ρυθμό ανάγνωσης και αποθήκευσης των δεδομένων στην μνήμη της. Γι' αυτό και πολλές εφαρμογές αξιοποιούν τις παραπάνω δυνατότητες για την ταχύτερη εκτέλεσή τους σε μια μονάδα επεξεργασίας γραφικών, από ότι σε μια μονάδα κεντρικής επεξεργασίας (επεξεργαστής ή CPU). Η διαφορά αυτή έγκειται στο γεγονός ότι, οι CPU έχουν σχεδιαστεί με γνώμονα την εκτέλεση μιας σειράς εντολών και λειτουργιών, το λεγόμενο νήμα (thread), καθώς και δεκάδες τέτοιων νημάτων ταυτόχρονα, όσον το δυνατόν ταχύτερα, σε αντίθεση με τις GPU οι οποίες έχουν σχεδιαστεί ώστε να εκτελούν χιλιάδες νημάτων παράλληλα και αποδοτικά [1]. Όπως φαίνεται και στο Σχήμα 2.1 μια CPU διαθέτει λιγότερους πυρήνες, μεγαλύτερη μνήμη προσωρινής αποθήκευσης και μεγαλύτερο έλεγχο ροής. Από την άλλη, η GPU αποτελείται από πολλούς περισσότερους πυρήνες, δίνοντας λιγότερη βάση στην μνήμη προσωρινής αποθήκευσης και στον έλεγχο ροής. Εργασίες με υψηλό υπολογιστικό κόστος εκτελούνται γρήγορα σε μια GPU, γι'αυτό και ένα κοστοβόρο ερώτημα όπως των "K πλησιέστερων ζευγών" αποτελεί ιδανική επιλογή για να υλοποιηθεί στην παραπάνω αρχιτεκτονική.

2.2 Αρχιτεκτονική NVIDIA GPU

Στην παρούσα διπλωματική εργασία, γίνεται χρήση της διεπαφής προγραμματισμού εφαρμογών (API) CUDA της Nvidia, για την υλοποίηση του αλγορίθμου που θα εκτελεί το ερώ-



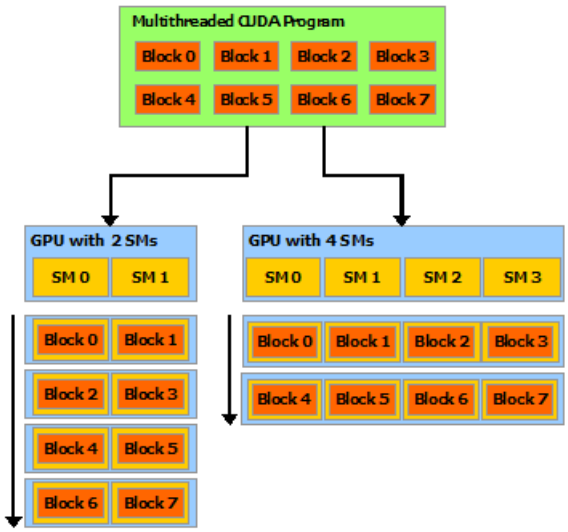
Σχήμα 2.1: Διαφορές στην αρχιτεκτονική μιας CPU και μιας GPU [1]

τημα. Για τον λόγο αυτό αναλύονται τα βασικά σημεία της συγκεκριμένης αρχιτεκτονικής, που θα βοηθήσουν στην κατανόηση της φιλοσοφίας με την οποία δημιουργήθηκε ο συγκεκριμένος αλγόριθμος. Η NVIDIA GPU αρχιτεκτονική είναι δομημένη γύρω από μια κλιμακούμενη σειρά πολυνημάτινης ροής πολυεπεξεργαστών- Multithreaded Streaming Multiprocessors (SMs) [2]. Κάθε πολυεπεξεργαστής μπορεί να εκτελέσει ταυτόχρονα πολλά μπλοκ νημάτων, τα οποία αποτελούν το λεγόμενο πλέγμα (grid), όπως φαίνεται στο Σχήμα 2.2. Ένα μπλοκ μπορεί να περιέχει πολλά νήματα στις περισσότερες κάρτες γραφικών, μέχρι 1024. Κάθε πλέγμα, έχει περιορισμένη χωρητικότητα μπλοκ που μπορεί να δεχτεί, όπως το ίδιο συμβαίνει και με τα μπλοκ που έχουν περιορισμένη χωρητικότητα νημάτων. Η χωρητικότητα σε κάθε στοιχείο ποικίλλει, ανάλογα με τα χαρακτηριστικά και τις δυνατότητες του μοντέλου της κάρτας γραφικών που χρησιμοποιείται.

2.3 Προγραμματιστικό μοντέλο CUDA

Όταν πρόκειται για τον προγραμματισμό σε CUDA και την εκτέλεση ενός αλγορίθμου ή μιας σειράς εντολών που επιδέχονται παραλληλοποίησης σε μια κάρτα γραφικών NVIDIA, συμβαίνουν με τη σειρά :

1. Γίνεται δέσμευση μνήμης στην κάρτα γραφικών (device memory) από τον εξυπηρετητή (host CPU) με συναρτήσεις που παρέχει η CUDA, των απαραίτητων δομών δεδομένων που θα χρησιμοποιηθούν για την εκτέλεση του αλγορίθμου.



Σχήμα 2.2: Κάθε πολυεπεξεργαστής (SM), μπορεί να εκτελέσει ταυτόχρονα πολλά μπλοκ νημάτων [1]

2. Ο εξυπηρετητής (host CPU) πρέπει να αντιγράψει τα απαραίτητα δεδομένα στην μνήμη της κάρτας γραφικών (device memory).

3. Κατά την κλήση ενός πυρήνα (kernel), αποφασίζεται από τον προγραμματιστή πόσα μπλοκ νημάτων καθώς και πόσα νήματα θα εκτελεστούν στην κάρτα γραφικών για την επεξεργασία του ερωτήματος.

4. Στη συνέχεια, τα μπλοκ και τα νήματα που έχουν ανατεθεί στην GPU εκτελούνται ταυτόχρονα για την εκτέλεση των εργασιών που είναι αναγκαίες και λύνουν το πρόβλημα.

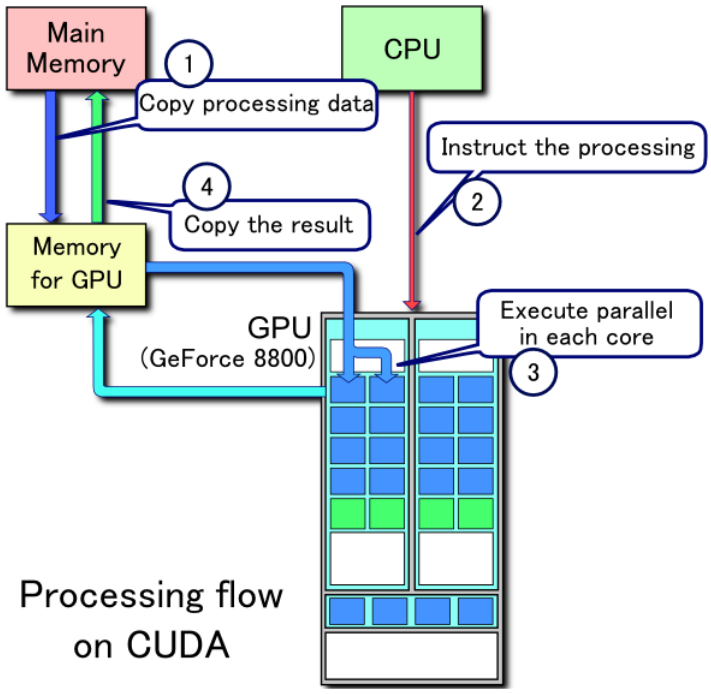
5. Όταν όλα τα νήματα από όλα τα μπλοκ, τελειώσουν με τις εργασίες που τους έχουν ανατεθεί, ο πυρήνας επιστρέφει στον εξυπηρετητή (host).

6. Για την ανάκτηση των αποτελεσμάτων από την device memory με τη βοήθεια συναρτήσεων του API της CUDA, αντιγράφονται ξανά τα δεδομένα από την μνήμη της κάρτας γραφικών στην μνήμη του εξυπηρετητή.

Στο Σχήμα 2.3, φαίνονται σχηματικά τα παραπάνω βήματα καθώς και πως λειτουργούν τα διάφορα στοιχεία στο μοντέλο CUDA μεταξύ τους.

2.3.1 Οργάνωση των νημάτων CUDA

Όλα τα νήματα σε ένα μπλοκ και κατά συνέπεια σε ένα πλέγμα, κατά την εκτέλεση ενός πυρήνα (kernel) διαχωρίζονται μεταξύ τους και το καθένα αναλαμβάνει ένα συγκεκριμένο

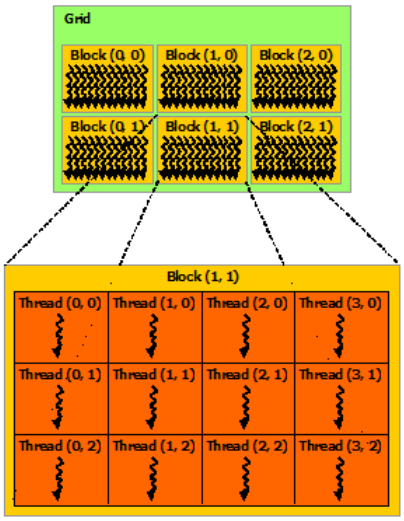


Σχήμα 2.3: Προγραμματιστικό μοντέλο CUDA [2]

μέρος της μνήμης προς επεξεργασία και αξιοποίησης του, για την εκτέλεση των εργασιών του. Ένας προγραμματιστής cuda θα πρέπει με κάποιο τρόπο να μπορεί να ξεχωρίσει τα νήματα και να του αναθέσει συγκεκριμένες λειτουργίες. Αυτό επιτυγχάνεται με τον τρόπο που είναι δομημένα τα μπλοκ και τα πλέγματα. Κάθε ένα από αυτά μπορεί να έχει μέχρι τρεις διαστάσεις και αφήνεται στην ευχέρεια του προγραμματιστή να αποφασίσει. Έτσι, η αναφορά σε ένα συγκεκριμένο νήμα γίνεται ανακτώντας τις συντεταγμένες του. Στο Σχήμα 2.4 φαίνεται ένα δισδιάστατο πλέγμα που περιέχει μπλοκ δύο διαστάσεων. Κάθε νήμα σε κάθε μπλοκ έχει δύο συντεταγμένες (για μπλοκ τριών διαστάσεων τρεις συντεταγμένες), μία που αντικατοπτρίζει την γραμμή στην οποία βρίσκεται και μία για την στήλη. Το ίδιο συμβαίνει και με τα μπλοκ μέσα στο πλέγμα. Συνεπώς, με ένα συνδυασμό συντεταγμένων του μπλοκ και του νήματος σε ένα μπλοκ, εντοπίζεται ένα συγκεκριμένο νήμα μέσα σε ολόκληρο το πλέγμα.

2.3.2 Εκτέλεση ενός πυρήνα και λεπτομέρειες υλοποίησης

Η CUDA είναι επέκταση της γλώσσας προγραμματισμού C++ και περιέχει επιπλέον συναρτήσεις για διάφορες λειτουργίες, απαραίτητες για την εκτέλεση των νημάτων σε μια κάρτα γραφικών NVIDIA.



Σχήμα 2.4: Πλέγμα (grid) δύο διαστάσεων, που περιέχει μπλοκ νημάτων, δύο διαστάσεων [1]

Στο Σχήμα 2.5 παρουσιάζεται ένα πρόγραμμα υλοποιημένο σε CUDA. Στην γραμμή 2 είναι ο ορισμός της συνάρτησης που εκτελεί τον πυρήνα. Ο πυρήνας εκτελείται στην κάρτα γραφικών και ορίζεται πάντα χρησιμοποιώντας τον προσδιοριστή δήλωσης `__global__`. Η κλήση του πυρήνα γίνεται από τον εξυπηρετητή (host) στην γραμμή 38 με μια νέα μορφή σύνταξης <<< ... >>> (`d_A`, `d_B` , `d_C`, N), όπου η παράμετρος "blocksPerGrid" προσδιορίζει τον αριθμό των blocks μέσα στο πλέγμα, ενώ η παράμετρος "threadsPerBlock" τον αριθμό των νημάτων που θα εκτελέσει τον πυρήνα. Οι μεταβλητές `d_A`, `d_B`, `d_C` και N, είναι παράμετροι της συνάρτησης πυρήνα. Στην προκειμένη περίπτωση είναι τρεις πίνακες τύπου float.

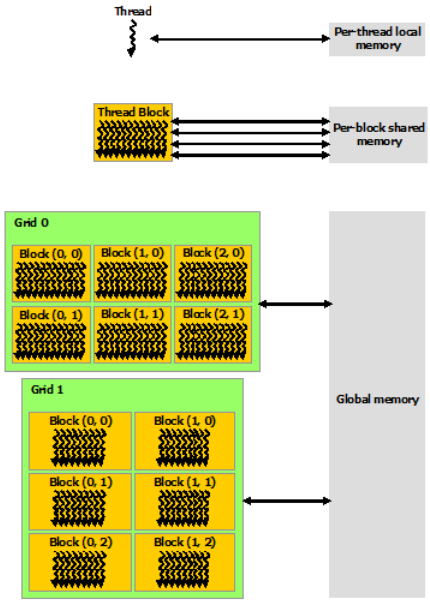
Σε κάθε πρόγραμμα CUDA οι παράμετροι της συνάρτησης πυρήνα θα πρέπει προηγούμενως να έχουν οριστεί μέσω του εξυπηρετητή και με κατάλληλες συναρτήσεις να δεσμευθεί μνήμη για τις δομές αυτές στην device memory. Ο ορισμός και η δέσμευση της μνήμης γίνεται στις γραμμές 23 - 28. Στις γραμμές 31,32 γίνεται αντιγραφή δεδομένων από την μνήμη του εξυπηρετητή στην μνήμη της κάρτας γραφικών. Πρέπει να τονιστεί το γεγονός ότι για την αντιγραφή των δεδομένων, χρησιμοποιούνται διαφορετικοί πίνακες από τον εξυπηρετητή, οι οποίοι έχουν οριστεί και αρχικοποιηθεί στην μνήμη του (γραμμές 16-20). Τέλος, μετά την επιστροφή από τον πυρήνα στην γραμμή 42, αντιγράφονται τα δεδομένα πίσω στον πίνακα του host.

```
1  // Device code
2  __global__ void VecAdd(float* A, float* B, float* C, int N)
3  {
4      int i = blockDim.x * blockIdx.x + threadIdx.x;
5      if (i < N)
6          C[i] = A[i] + B[i];
7  }
8
9  // Host code
10 int main()
11 {
12     int N = ...;
13     size_t size = N * sizeof(float);
14
15     // Allocate input vectors h_A and h_B in host memory
16     float* h_A = (float*)malloc(size);
17     float* h_B = (float*)malloc(size);
18
19     // Initialize input vectors
20     ...
21
22     // Allocate vectors in device memory
23     float* d_A;
24     cudaMalloc(&d_A, size);
25     float* d_B;
26     cudaMalloc(&d_B, size);
27     float* d_C;
28     cudaMalloc(&d_C, size);
29
30     // Copy vectors from host memory to device memory
31     cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
32     cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
33
34     // Invoke kernel
35     int threadsPerBlock = 256;
36     int blocksPerGrid =
37             (N + threadsPerBlock - 1) / threadsPerBlock;
38     VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
39
40     // Copy result from device memory to host memory
41     // h_C contains the result in host memory
42     cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
43
44     // Free device memory
45     cudaFree(d_A);
46     cudaFree(d_B);
47     cudaFree(d_C);
48
49     // Free host memory
50     ...
51 }
```

Σχήμα 2.5: Ένα τυπικό πρόγραμμα CUDA [1]

Δεικτιοδότηση νημάτων και πρόσβαση στη μνήμη

Όπως αναφέρθηκε όλα τα νήματα σε ένα πλέγμα εκτελούν τον κώδικα του πυρήνα. Ο κώδικας που παρουσιάζεται στο Σχήμα 2.5 έχει παρόμοια φιλοσοφία με αυτήν που χρησιμοποιήθηκε στον αλγόριθμο που υλοποιείται στην παρούσα εργασία. Στην γραμμή 4 γίνεται χρήση των ενσωματωμένων μεταβλητών του περιβάλλοντος CUDA, “blockDim.x”, “blockIdx.x” και “threadIdx.x”, με σκοπό κάθε ένα ξεχωριστό νήμα σε κάθε μπλοκ του πλέγματος να εκτελεί διαφορετικό υπολογισμό. Οι μεταβλητές αυτές ορίζουν τις συντεταγμένες των μπλοκ και των νημάτων, αλλά και τις διαστάσεις τους. Στο παράδειγμα αυτό, κάθε νήμα προσθέτει κάθε στοιχείο του A πίνακα με το αντίστοιχο στοιχείο του B πίνακα και αποθηκεύει το αποτέλεσμα στο στοιχείο του C (γραμμή 6). Έτσι, προκύπτει συνολικά η πρόσθεση



Σχήμα 2.6: Είδη μνήμης και εμβέλεια πρόσβασης [1]

πινάκων A και B, η οποία όμως έχει γίνει παράλληλα εκτελούμενη στην device memory. Πρέπει να τονιστεί ότι ο δείκτης "i" είναι μοναδικός για κάθε νήμα και αναφέρεται σε αυτό. Ο λόγος που συμβαίνει είναι επειδή με τον συνδυασμό των συντεταγμένων των μπλοκ οι οποίες είναι μοναδικές και των νημάτων, προσδιορίζουν ένα και μοναδικό νήμα στο πλέγμα.

2.3.3 Είδη και ιεραρχία μνήμης

Υπάρχουν διαφόρων ειδών μνήμες στις οποίες παρέχεται περιορισμένη πρόσβαση αναλόγως το είδος τους, όπως φαίνεται στο Σχήμα 2.6. Επίσης αξίζει να αναφερθεί ότι διαφέρουν και στην ταχύτητα απόκρισης μιας ανάγνωσης ή μιας εγγραφής. Αυτές είναι οι παρακάτω:

1. Καθολική μνήμη ή αλλιώς global memory : Είναι η μνήμη, στην οποία έχουν πρόσβαση όλα τα νήματα από όλα τα μπλοκ. Μία αλλαγή σε θέση μνήμης που βρίσκεται σε αυτήν, είναι ορατή σε όλα τα νήματα. Η ταχύτητα απόκρισής της είναι η πιο αργή από όλες τις μνήμες που υπάρχουν στο μοντέλο CUDA.

2. Κοινόχρηστη μνήμη ή αλλιώς shared memory : Είναι ξεχωριστή για κάθε μπλοκ. Επόμενως, οποιαδήποτε αλλαγή σε θέση μνήμης μέσα σε αυτή, είναι ορατή μόνο στα νήματα του συγκεκριμένου μπλοκ. Η ταχύτητα απόκρισης της είναι γρήγορη, αλλά έχει και μικρότερο μέγεθος.

3. Τοπική μνήμη ή αλλιώς local memory : Είναι μνήμη ξεχωριστή για κάθε νήμα. Πρό-

σβαση σ'αυτήν έχει μόνο ένα νήμα, δεν μπορεί άλλο να αναφερθεί στην ίδια. Η ταχύτητα απόκρισής της είναι ταχύτερη από τις προηγούμενες δύο που αναφέρθηκαν.

4. Καταχωρητές ή αλλιώς registers : Είναι θέσεις μνήμης αποκλειστικά για χρήση από ένα νήμα. Δεν μπορεί ένα νήμα να χρησιμοποιήσει τον καταχωρητή ενός άλλου. Είναι περιορισμένοι σε ποσότητα, αλλά έχουν την ταχύτερη απόκριση απ'όλες τις μνήμες. Χρησιμοποιούνται για την αποθήκευση των μεταβλητών του νήματος.

5. Υπάρχουν άλλες δύο μνήμες, οι σταθερές και οι υφής (constant and texture). Είναι προσβάσιμες ακόμα κι από διαφορετικούς πυρήνες. Είναι μόνο για ανάγνωση και όλα τα νήματα έχουν πρόσβαση σε αυτές. Γίνεται χρήση τους μόνο για ειδικούς σκοπούς και εφαρμογές.

2.4 Χωρικό ερώτημα "Κορυφογραμμών"

Στο [10] ο συγγραφέας ορίζει το χωρικό ερώτημα "Κορυφογραμμών" (Spatial Skyline Query) όπως παρακάτω :

Ορισμός 2.1 (Skyline Query)**.** *Δεδομένου ενός συνόλου σημείων $p_1, p_2, ..., p_N$, το χωρικό ερώτημα "Κορυφογραμμών" βρίσκει ένα σύνολο σημείων P, τα οποία ονομάζονται "σημεία κορυφογραμμής", τέτοια ώστε οποιοδήποτε άλλο σημείο στο σύνολο δεν επικρατεί οποιοδήποτε σημείο $p_i \in P$. Ένα σημείο p_1 επικρατεί ενός άλλου σημείου p_2 εάν το p_1 δεν είναι χειρότερο του p_2 σε όλες τις διαστάσεις και το p_1 είναι καλύτερο από το p_2 σε τουλάχιστον μία διάσταση.*

Με άλλα λόγια, ένα σημείο p_1 επικρατεί ενός άλλου σημείου p_2 εάν και μόνο αν η συντεταγμένη του p_1 σε τουλάχιστον έναν άξονα είναι μεγαλύτερη από την αντίστοιχη συντεταγμένη του p_2.

2.5 Χωρικό ερώτημα "Κ πλησιέστερων γειτόνων"

2.5.1 Ορισμός

Σύμφωνα με τους ορισμούς που αναφέρονται στο [11], το ερώτημα των Κ πλησιέστερων γειτόνων μπορεί να οριστεί :

Ορισμός 2.2 (kNN). *Δεδομένου ενός σημείου p, ενός συνόλου S και ενός ακεραίου k, οι k κοντινότεροι γείτονες του p από το S, συμβολίζοντας ως kNN (p, S), είναι είναι σύνολο k σημείων του S, τέτοιο ώστε,* $\forall r \in kNN(p,S), \forall q \in S - kNN(p,S), dist(p,r) \leq dist(p,q)$.

Ορισμός 2.3 (AkNNQ). *Δεδομένου 2 συνόλων R και S και ενός ακεραίου k, το αποτέλεσμα του ερωτήματος "Όλοι οι κ κοντινότεροι γείτονες του R από το S", συμβολίζοντας ως AkNNQ(R, S), είναι το σύνολο των ζευγαριών* $\{(r,s) : r \in R, s \in kNN(r,S)\}$.

Στην παρούσα εργασία για την μέτρηση της απόστασης μεταξύ των σημείων, χρησιμοποιείται η Ευκλείδια απόσταση. Παρόλα αυτά θα μπορούσε να χρησιμοποιηθεί και άλλη μετρική της απόστασης (π.χ. απόσταση Μανχάταν κ.ά.).

2.6 Χωρικό ερώτημα "Κ πλησιέστερων γειτόνων για ομάδα σημείων"

Στο άρθρο [12] αυτό το χωρικό ερώτημα ορίζεται :

Ορισμός 2.4 (KGNNQ). *Δεδομένου ενός συνόλου* $P = \{p_1, .., p_N\}$ *στατικών σημείων στον πολυδιάστατο χώρο και μια ομάδα σημείων ερωτήματος* $Q = \{q_1, .., q_N\}$, *τα K* (≥ 1) *σημεία με το μικρότερο άθροισμα αποστάσεων από όλα τα σημεία που ανήκουν στο σύνολο Q αποτελούν τους K πλησιέστερους γείτονες της ομάδας σημείων Q. Η απόσταση μεταξύ ενός σημείου p και του Q ορίζεται ως* $dist(p,Q) = \sum_{i=1\ n} |pq_i|$, *όπου* $|pq_i|$ *είναι η Ευκλείδια απόσταση μεταξύ του p και του σημείου ερωτήματος* q_i.

2.7 Χωρικό ερώτημα "Κ πλησιέστερων ζευγών"

2.7.1 Ορισμός

Το χωρικό ερώτημα των Κ πλησιέστερων ζευγών ορίζεται στο άρθρο [13] :

Ορισμός 2.5 (kCP). *Έστω 2 σύνολα σημείων,* $P = \{p_1, p_2, ..., p_{NP}\}$ *και* $Q = \{q_1, q_2, ..., q_{NQ}\}$ *ως K πλησιέστερα ζεύγη των P και Q ορίζουμε το σύνολο των K διατεταγμένων ζευγών*

$(p_{z_1}, q_{l_1}), (p_{z_2}, q_{l_2}), ..., (p_{z_k}, q_{l_K}),$

$p_{z_1}, p_{z_2}, ..., p_{z_K} \in P \wedge q_{l_1}, q_{l_2}, ..., q_{l_K} \in Q$

τέτοια ώστε

$$dist(p_i, q_j) \geq dist(p_{z_K}, q_{l_K}) \geq dist(p_{z_{(K-1)}}, q_{l_{(K-1)}}) \geq ... \geq dist(p_{z_1}, q_{l_1}),$$
$$\forall (p_i, q_j) \in (P \times Q - \{(p_{z_1}, q_{l_1}), (p_{z_2}, q_{l_2}), ..., (p_{z_k}, q_{l_K})\})$$

Με άλλα λόγια, τα K πλησιέστερα ζεύγη των P και Q είναι αυτά που έχουν τις K μικρότερες αποστάσεις μεταξύ όλων των ζευγαριών σημειών που μπορούν να εξαχθούν, επιλέγοντας ένα σημείο του P και ένα σημείο του Q. Το K πρέπει να είναι μικρότερο από όλο το σύνολο των ζευγαριών σημείων που μπορούν να βρεθούν μεταξύ των συνόλων P και Q. Επίσης, λόγω ισοπαλιών στις αποστάσεις, το αποτέλεσμα μπορεί να μην είναι μοναδικό, καθώς ενδέχεται να υπάρχουν διαφορετικά σημεία με τις μικρότερες αποστάσεις. Όμως, οι K μικρότερες αποστάσεις θα είναι πάντα ίδιες. Τέλος, ο όρος "dist" αναφέρεται σε κάποια μετρική της απόστασης (Ευκλείδια, Minkowski κ.ά.).

Κεφάλαιο 3

Συγγενικές Εργασίες

3.1 Εισαγωγή

Τα πιο συχνά χωρικά ερωτήματα που αποτελούν αντικείμενο μελέτης της επιστημονικής κοινότητας τα τελευταία χρόνια είναι αυτά των "Κ πλησιέστερων γειτόνων", "Κ πλησιέστερων ζευγών", "Κ πλησιέστερων ομάδων" καθώς επίσης και χωρικά ερωτήματα "Κορυφογραμμών" (Spatial Skyline Queries). Για την μελέτη και την υλοποίηση αλγορίθμων που να απαντούν σε αυτά τα ερωτήματα αποτελεσματικά και γρήγορα έχουν χρησιμοποιηθεί διάφορες τεχνικές και αρχιτεκτονικές. Επίσης, το κάθε ερώτημα λόγω της φύσης του, υπόκειται σε διαφορετική προεπεξεργασία και σε διαφορετικούς τρόπους αποθήκευσης, με τέτοιο τρόπο που να βελτιώνεται η απόδοσή της εκτέλεσής του αλλά και να γίνεται πλήρη αξιοποίηση των πόρων και των δυνατοτήτων της αρχιτεκτονικής πάνω στην οποία εκτελείται. Ο σκοπός της διπλωματικής είναι η ανάπτυξη και υλοποίηση ενός αλγορίθμου, που θα εκτελείται σε μια μονάδα επεξεργασίας γραφικών. Επομένως καθίσταται αναγκαία η μελέτη συγγενών χωρικών ερωτημάτων και των μεθοδολογιών που χρησιμοποιούν για την υλοποίηση αποδοτικών αλγορίθμων, ουτώς ώστε να προκύψει μια δυνατή και αξιοποιήσιμη λύση για την εκτέλεση του ερωτήματος των "Κ πλησιέστερων ζευγών" σε μια αρχιτεκτονική GPU.

3.2 Επεξεργασία ερωτημάτων σε χωρικές βάσεις δεδομένων

Για την επεξεργασία χωρικών ερωτημάτων θα πρέπει να ληφθούν υπ'όψιν αρκετοί παράμετροι : τα χωρικά μοντέλα, οι μηχανισμοί ευρετηρίασης και η αποδοτική επεξεργασία των

ερωτημάτων [14]. Ένα Σύστημα Χωρικής Βάσης Δεδομένων (ΣΧΒΔ) δίνει την δυνατότητα χρήσης χωρικών τύπων δεδομένων παρέχοντας μηχανισμούς ευρετηρίασης για αποδοτική επεξεργασία των ερωτημάτων [15].

Οι Χωρικές Βάσεις Δεδομένων έχουν προσελκύσει το ενδιαφέρον των ερευνητών τα τελευταία χρόνια και οι λόγοι είναι πολλοί. Οι εφαρμογές τους περιλαμβάνουν τα Γεωγραφικά Πληροφοριακά Συστήματα , Πληροφοριακά Συστήματα Πολυμέσων, Αποθήκες Δεδομένων, Υπηρεσίες που προσφέρονται με την χρήση της τοποθεσίας του χρήστη [14] κ.ά.

Οι τύποι των δεδομένων που περιέχονται σε ένα ΣΧΒΔ διακρίνονται σε : σημεία, γραμμές, πολύγωνα αλλά και περιοχές [14]. Οι βασικές ενέργειες που μπορεί κανείς να εφαρμόσει σε τέτοια στοιχεία είναι η εύρεση σημείων τομής, η εύρεση επικάλυψης και η μέτρηση της απόστασης. Η απόδοση των παραπάνω λειτουργιών εξαρτάται από τις διάφορες μεθόδους πρόσβασης στα χωρικά δεδομένα αλλά και τους αλγορίθμους που εφαρμόζονται στους μηχανισμούς ευρετηρίασης.

Στο άρθρο [14], οι συγγραφείς ορίζουν την επεξεργασία χωρικών ερωτημάτων και μελετούν την υπάρχουσα βιβλιογραφία για μεθόδους και λειτουργίες που εφαρμόζονται σε χωρικά δεδομένα. Παρουσιάζουν τις διαφορές μεταξύ μιας σχεσιακής βάσης δεδομένων από το ΣΧΒΔ. Επίσης, αναλύουν τις διάφορες λειτουργίες σε μία χωρική βάση δεδομένων, αφού πρώτα ορίσουν τι είναι ένα χωρικό αντικείμενο. Τονίζουν, ότι υπάρχουν διάφορες τεχνικές και στρατηγικές για την αντιμετώπιση ενός χωρικού ερωτήματος. Συνοψίζουν τις υπάρχουσες υλοποιήσεις και τους μηχανισμούς που χρησιμοποιούνται για το χωρικό ερώτημα "Κ πλησιέστερων γειτόνων". Ειδικά χωρικά ερωτήματα είναι υπαρκτά και σύμφωνα με τους συγγραφείς θα πρέπει να προσελκύσουν την προσοχή των ερευνητών. Ορίζουν το Web-based Γεωγραφικό Πληροφοριακό Σύστημα (WGIS) και την σημαντικότητά του στην επεξεργασία χωρικών ερωτημάτων. Τέλος, παρουσιάζουν μελλοντικές προοπτικές έρευνας περιλαμβανομένων όχι μόνο δισδιάστατων μοντέλων αλλά και μοντέλα με περισσότερες διαστάσεις, τη δημιουργία μοντέλων που να αναλύουν το κόστος εκτέλεσης των χωρικών ερωτημάτων, τη Δικτυακή Χωρική Βάση Δεδομένων και τη εξερεύνηση της επεξεργασίας των χωρικών ερωτημάτων χωρίς την χρήση μηχανισμών ευρετηρίασης.

3.3 Χωρικό ερώτημα "Κορυφογραμμών"

Ο συγγραφέας στο [10] παρουσιάζει αλγορίθμους που εφαρμόζονται σε τέτοια ερωτήματα, μαζί με τα πλεονεκτήματά και τα μεινονεκτήματα τους καθώς επίσης τις διάφορες εκδοχές του ερωτήματος "Κορυφογραμμών" (SSQ). Ο πρώτος αλγόριθμος ονομάζεται "Block Nested Loop (BNL) Algorithm" όπου κάθε σημείο p συγκρίνεται με κάθε άλλο σημείο στο σύνολο. Τα μειονεκτήματα του αλγορίθμου περιλαμβάνουν την εξάρτηση από την κύρια μνήμη, η αδυναμία για online επεξεργασία αφού πρέπει να διαβαστεί ολόκληρο το αρχείο για να βρει το πρώτο σημείο "Κορυφογραμμής" και εκτελεί αρκετούς περιττούς υπολογισμούς. Ο δεύτερος αλγόριθμος είναι ο "Divide-and-conquer algorithm" όπου γίνεται διαμέριση του συνόλου σε υποσύνολα που χωράνε στη μνήμη και για κάθε υποσύνολο υπολογίζεται το σημείο "Κορυφογραμμής". Είναι αποδοτικός μόνο για μικρά σύνολα και δεν είναι δυνατή η Online επεξεργασία του συνόλου καθώς το αποτελέσμα δεν μπορεί να σταλεί αν δεν έχει τελειώσει η φάση της διαμέρισης. Ο τρίτος αλγόριθμος είναι ο "Nearest Neighbor (NN) Algorithm" όπου τα αποτελέσματα αυτού χρησιμοποιούνται για αναδρομική διαμέριση του συνόλου και χρησιμοποιείται ένας μηχανισμός ευρετηρίασης των δεδομένων. Το μειονέκτημα του είναι ότι για πολυδιάστατα δεδομένα θα πρέπει ένα από τα βήματα να τροποποιηθεί για βελτίωση της απόδοσης. Τέταρτος και τελευταίος ο "Branch and Bound Skyline (BBS) Algorithm" βασίζεται στην αναζήτηση πλησιέστερων γειτόνων με τη χρήση του προτύπου "Branch and Bound". Τέλος, σύμφωνα με το άρθρο οι εφαρμογές των ερωτημάτων "Κορυφογραμμής" αλλά και των διαφόρων εκδοχών του, στις οποίες περιλαμβάνεται το Χωρικό Ερώτημα "Κορυφογραμμών", μπορούν να βρεθούν σε μηχανισμούς αποφάσεων, εφαρμογές κινητών, εφαρμογές ροής δεδομένων (streaming), εφαρμογές που λαμβάνουν υπ'όψιν την τοποθεσία του χρήστη κ.ά.

3.4 Χωρικό ερώτημα "Κ πλησιέστερων γειτόνων"

Στο άρθρο [16] υλοποιούνται δύο νέοι αλγόριθμοι που απαντούν το χωρικό ερώτημα "Κ πλησιέστερων γειτόνων" (KNNQ). Γίνεται χρήση της βιβλιοθήκης Thrust, της CUDA, που σημαίνει ότι η εκτέλεσή τους γίνεται στην κάρτα γραφικών. Αυτή η βιβλιοθήκη μοιράζει αποδοτικά τις εργασίες στα επιμέρους νήματα. Η πρώτη υλοποίηση T-BF είναι παραλληλοποίηση του αλγορίθμου Brute Force με την χρήση της Thrust, ενώ η δεύτερη T-DS βασίζεται σε μια τεχνική όπου εξετάζει πρώτα τα πλησιέστερα σημεία, μέσα σ'έναν κύκλο αναζήτησης

και αυξάνει ημι-εκθετικά την ακτίνα του κύκλου που γίνεται η αναζήτηση. Αποδεικνύεται ότι αυτές οι μεθόδοι που κάνουν χρήση της Thrust, λειτούργουν ταχύτερα από υπάρχουσες μεθόδους για μικρά σύνολα δεδομένων (query points), τα οποία είναι το σύνολο των σημείων που για το καθένα αναζητώνται οι K κοντινότεροι γείτονες (KNN).

Οι ίδιοι συγγραφείς σε μετέπειτα εργασία [17] βελτιώνουν την εκτέλεση του ερωτήματος, προτείνοντας μια νέα μέθοδο. Αυτή βασίζεται στην τεχνική της διαμέρισης (partitioning), δηλαδή στη διαμοίραση των query points σε χωρίσματα (partitions), ούτως ώστε να γίνεται περιορισμός των σημείων που εξετάζονται για K πλησιέστεροι γείτονες. Σύμφωνα με το άρθρο, στις προηγούμενες μεθόδους T-BF, T-DS, όλα τα νήματα CUDA αφορούσαν τον υπολογισμό KNN για ένα query point. Σε αυτήν την εργασία όμως, κάθε νήμα βρίσκει τους KNN για ένα query point παράλληλα με άλλο νήμα. Η όλη διαδικασία εκτελείται στην GPU, γεγονός που κάνει την υλοποίηση ταχύτερη. Τέλος, λόγω του ότι δεν εξετάζει όλα τα σημεία του reference dataset (σύνολο με σημεία τα οποία είναι υποψήφια για KNN), μπορεί να επεξεργαστεί μεγαλύτερα σύνολα. Η υλοποίηση αυτή λειτουργεί ταχύτερα από τις προηγούμενες και υπάρχουσες μεθόδους στη βιβλιογραφία.

3.5 Χωρικό ερώτημα "K πλησιέστερων γειτόνων για μια ομάδα σημείων"

Στο άρθρο [18], οι συγγραφείς εξετάζουν το ερώτημα "K πλησιέστερων γειτόνων για μια ομάδα σημείων" (KGNNQ). Αυτό το χωρικό ερώτημα είναι μία επέκταση του χωρικού ερωτήματος "K κοντινότερων γειτόνων". Ένα παράδειγμα που οι συγγραφείς αναφέρουν είναι όταν υπάρχει ένα σύνολο σημείων συνάντησης (το σύνολο σημείων) και ένα άλλο σύνολο από τοποθεσίες χρηστών (σύνολο ερωτήματος). Το πρόβλημα αναφέρεται στην εύρεση ενός (ή K) σημείου (-ων) συνάντησης τέτοιο ώστε να ελαχιστοποιείται το άθροισμα των αποστάσεων για όλες τις τοποθεσίες των χρηστών, αφού κάθε χρήστης θα μετακινηθεί από την τοποθεσία του σε καθέ ένα από τα σημεία συνάντησης. Για την επίλυση αυτού του προβλήματος σε αυτή την εργασία οι συγγραφείς υλοποιούν δύο αλγορίθμους που βασίζονται στην τεχνική Σάρωσης Επιπέδου για δεδομένα που δεν είναι αποθηκευμένα σε κάποιον μηχανισμό ευρετηρίασης. Το αποτελέσμα της πειραματικής αξιολόγησης αποδεικνύει ότι οι αλγόριθμοι επιτυγχάνουν ανταγωνιστική απόδοση σε σχέση με προηγούμενες υλοποιήσεις.

3.6 Χωρικό ερώτημα "Κ πλησιέστερων ζευγών"

3.6.1 Υλοποιήσεις σε μονάδα κεντρικής επεξεργασίας

Οι συγγραφείς στο [13] ήταν αυτοί που πρώτοι μελέτησαν διεξοδικά και υλοποίησαν αλγορίθμους που να απαντούν το ερώτημα των "Κ πλησιέστερων ζευγών" (KCP). Στην εργασία τους αυτή παρουσιάζουν τέσσερις διαφορετικούς αλγορίθμους, εκ των οποίων οι τρεις είναι αναδρομικοί ενώ ο ένας είναι επαναληπτικός και εφαρμόζονται σε σύνολα δύο διαστάσεων. Οι αλγόριθμοι αυτοί βασίζονται στο γεγονός ότι για την αποθήκευση των δύο συνόλων P και Q, γίνεται χρήση μιας δομής ευρετηρίασης που ονομάζεται R-δέντρο (R-trees). Οι δομές αυτές είναι ιεραρχικά δομημένες και βασίζονται στα B+δέντρα (B+trees). Παρουσιάζονται αναλυτικά αποτελέσματα των πειραμάτων λαμβάνοντας διαφορετικές παραμέτρους κάθε φορά, όπως το ποσοστό επικάλυψης των συνόλων δεδομένων, τον αριθμό τον Κ πλησιέστερων ζευγών που αναζητείται καθώς κι άλλων παραμέτρων, δομών δεδομένων, που δεν είναι όμως το αντικείμενο μελέτης της παρούσας εργασίας. Οι συγγραφείς συμπεραίνουν ότι για επικαλυπτόμενα σύνολα δεδομένων κάποιοι από τους αλγόριθμους συμπεριφέρονται καλύτερα κι ότι αριθμός των Κ πλησιέστερων ζευγών, δεν επηρεάζει δραματικά την απόδοση. Αν και παρατηρείται ότι ο μη-αναδρομικός αλγοριθμός συμπεριφέρεται καλύτερα στις περισσότερες ρυθμίσεις, γενικά δεν καταλήγουν σε έναν αλγόριθμο που να είναι ο καταλληλότερος από όλους, αλλά κυρίως συγκρίνουν τις υλοποιήσεις μεταξύ τους.

Σε επόμενο άρθρο τους [6] οι ίδιοι συγγραφείς, βελτιώνουν την εκτέλεση του ίδιου ερωτήματος παρουσιάζοντας τρεις νέους αλγόριθμους. Δύο εξ αυτών είναι αναδρομικοί και ένας επαναληπτικός. Η βελτίωση που προτείνουν βασίζεται στην ταξινόμηση της αναζήτησης στον έναν αλγόριθμο, ενώ στους άλλους δύο χρησιμοποιούν μια νέα τεχνική που ονομάζεται "Σάρωση Επιπέδου" ή "Plane Sweep". Η υλοποίηση του αλγορίθμου της παρούσας διπλωματικής εργασίας, βασίζεται σε αυτήν την τεχνική, οπότε θα αναλυθεί αργότερα. Και σε αυτό το άρθρο, τα σύνολα δεδομένων αποθηκεύονται σε R-trees. Αξίζει να αναφερθεί ότι η πειραματική αξιολόγηση έγινε λαμβάνοντας υπόψιν πέντε κριτήρια. Αυτά είναι ο αριθμός των προσβάσεων στο δίσκο, ο χρόνος απόκρισης, ο αριθμός των υπολογιζόμενων αποστάσεων, ο αριθμός των υποπροβλημάτων που δημιουργήθηκαν και θεωρήθηκε αναγκαίο να συμπεριληφθεί, ο αριθμός των εισαγωγών σε ένα σωρό μεγίστου για τον επαληπτικό αλγόριθμο μόνο. Το συμπέρασμα που προκύπτει είναι ότι ο επαναληπτικός αλγόριθμος (χρησιμοποιεί και την τεχνική Plane sweep) έχει τις λιγότερες προσβάσεις στο δίσκο, όσο ο αριθμός των Κ

πλησιέστερων ζευγών αυξάνεται, γεγονός που βελτιώνει την απόδοση. Επίσης, για μη επικαλυπτόμενα σύνολα δεδομένων ο ίδιος αλγόριθμος υπερτερεί και στα πέντε κριτήρια των υπολοίπων. Τέλος, για τρία από αυτά τα κριτήρια μέτρησης της απόδοσης, οι συγγραφείς καταλήγουν ότι η τεχνική "Σάρωσης Επιπέδου" αποδίδει πολύ καλύτερα.

3.6.2 Υλοποιήσεις σε παράλληλα και κατανεμημένα συστήματα

Όπως έχει αναφερθεί και στο [6], το ερώτημα των "Κ πλησιέστερων ζευγών" είναι ένα αποτέλεσμα δύο ερωτημάτων, των "Κ πλησιέστερων γειτόνων" και του ερωτήματος ένωσης αποστάσεων (Distance Join Query).

Στο άρθρο [19] τονίζεται η αναγκαιότητα να υλοποιηθεί το ερώτημα των "Κ πλησιέστερων ζευγών" σε ένα παράλληλο και κατανεμημένο περιβάλλον, αφού το συγκεκριμένο ερώτημα πέρα από τον συνδυασμό των παραπάνω χωρικών ερωτημάτων, περιλαμβάνει και ιδιαίτερα πολλούς και κοστοβόρους υπολογισμούς. Οι συγγραφείς θεωρούν ότι το Spatial-Hadoop, αποτελεί μια ελκυστική επιλογή. Τα δύο σύνολα δεδομένων χωρίζονται σε μέρη τα οποία διανέμονται στους κόμβους του κατανεμημένου συστήματος. Προτείνουν έναν νέο αλγόριθμο που εκτελείται στο παραπάνω σύστημα, χρησιμοποιώντας Plane sweep μεθόδους, όμως με κάποια βελτίωση, αφού στη φάση της προεπεξεργασίας υπολογίζουν ένα άνω όριο απόστασης (bound) από ένα δείγμα δεδομένων σημείων, το οποίο χρησιμοποιείται για την ελαχιστοποίηση των υπολογισμών που απαιτούνται για να απαντηθεί το ερώτημα. Τα αποτελέσματα των πειραμάτων αποδεικνύουν την αποτελεσματικότητα του αλγορίθμου όσον αφορά τον συνολικό χρόνο εκτέλεσης, τον αριθμό των συνολικών αποστάσεων που συμπεριλήφθηκαν στον υπολογισμό, καθώς και την ιδιότητα του αλγορίθμου να δέχεται μεγαλύτερο K (πλησιέστερα ζεύγη) και μεγαλύτερα σύνολα δεδομένων .

Μια άλλη υλοποίηση παράλληλη και κατανεμημένη παρουσιάζεται στο [20]. Το σύστημα που χρησιμοποιείται είναι το Apache Spark το οποίο θεωρείται από τους συγγραφείς ότι έχει περισσότερα πλεονεκτήματα από άλλα, όπως το SpatialHadoop. Η φιλοσοφία της υλοποίησης βασίζεται στο ότι τα σύνολα δεδομένων χωρίζονται σε χωρίσματα (partitions) κατά μήκος ενός άξονα (π.χ. άξονα των x). Τα χωρίσματα αυτά διανέμονται στους κόμβους του κατανεμημένου συστήματος και επεξεργάζονται ανεξάρτητα. Η διαμοίραση των σημείων, γίνεται λαμβάνοντας υπόψιν δύο κριτήρια. Είτε τα χωρίσματα θα έχουν ίδιο πλάτος, είτε θα περιέχουν το ίδιο πλήθος σημείων (ίδιο μέγεθος). Κάνουν σύγκριση των δύο υλοποιήσεων που βασίζονται στα παραπάνω κριτήρια με μία τρίτη, σχεδιασμένη από τους

ίδιους, στην οποία ο αριθμός των χωρισμάτων είναι προκαθορισμένος. Πρέπει να τονιστεί ότι σε κάθε χώρισμα ο υπολογισμός τον Κ πλησιέστερων ζευγών γίνεται με χρήση της μεθόδου Plane sweep. Το αποτέλεσμα της αξιολόγησης των μεθόδων που παρουσιάζονται σε αυτό το άρθρο, αποδεικνύει ότι παρέχει μια ευελιξία στην επιλογή του αριθμού των χωρισμάτων που θα προκύψουν, αφού ο διαμοιρασμός προσδιορίζεται από τον αριθμό των σημείων σε αυτά, με συνέπεια τη βελτίωση της ταχύτητας εκτέλεσης του ερωτήματος.

Μια αρκετά σχετική υλοποίηση με την προηγούμενη αναφέρεται στο [3]. Σύμφωνα με το άρθρο αυτό, στο χωρικό ερώτημα των "Κ πλησιέστερων ζευγών", τα ζευγάρια που αποτελούν λύση του προβλήματος συνήθως είναι διασκορπισμένα σε όλο το χώρο, οπότε μία μέθοδος όπως της ευρετηρίασης (π.χ. R-trees) δε θα μπορούσε να επιταχύνει την εκτέλεση. Οι συγγραφείς αρχικά χωρίζουν τα σύνολα δεδομένων σε χωρίσματα (partitions). Αυτά τα χωρίσματα μοιράζονται στους κόμβους (Workers) όπου γίνεται κατανεμημένος υπολογισμός των ζευγών. Η υλοποίηση πραγματοποιείται στο Apache Spark. Ο αλγόριθμος χωρίζεται σε τέσσερις φάσεις. Κατά την πρώτη φάση δεδομένου ενός δείγματος των συνόλων, υπολογίζεται γρήγορα ένα άνω όριο, bound. Το όριο αυτό χρησιμοποιείται στη συνέχεια για τον περιορισμό των χωρισμάτων που είναι πιθανό να περιέχουν κάποια από τα Κ πλησιέστερα ζεύγη και κατ'επέκταση την ελαχιστοποίηση των υπολογισμών. Στη δεύτερη φάση γίνεται ένας πρώτος υπολογισμός των Κ πλησιέστερων ζευγών που δεν είναι όμως τα τελικά ζευγάρια. Στην τρίτη φάση συμπεριλαμβάνονται επιπλέον χωρίσματα στον υπολογισμό κι έτσι θα προκύψουν τα τελικά αποτελέσματα όπου στην φάση 4 γίνεται η συλλογή των Κ πλησιέστερων ζευγών. Κάποια σημεία που αξίζει να εστιάσει κανείς, είναι ότι γίνεται χρήση δύο διαφορετικών τρόπων για την εύρεση των σημείων που αποτελούν τα διαχωριστικά σημεία (split points), από τα οποία προκύπτουν τα χωρίσματα και αποτελεί σημαντικό παράγοντα στην απόδοση του ερωτήματος. Λόγω των σημαντικών βελτιώσεων που προέκυψαν, η εκτέλεση του ερωτήματος είναι ταχύτερη όπως επίσης και η επεκτασιμότητα είναι μεγαλύτερη, σε σχέση με προηγούμενες υλοποιήσεις των ίδιων συγγραφέων. Τέλος, τονίζεται ότι αυτή η μέθοδος της διαμέρισης φαίνεται να είναι αποτελεσματική για το χωρικό ερώτημα των "Κ πλησιέστερων ζευγών" σε παράλληλα και κατανεμημένα συστήματα, καθιστώντας αναγκαία την περαιτέρω εξερεύνηση της μεθόδου.

Κεφάλαιο 4

Παράλληλος Αλγόριθμος Κ Πλησιέστερων Ζευγών

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει η παρουσίαση και η ανάλυση του αλγορίθμου που αναπτύχθηκε και υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας. Πρόκειται για έναν αλγόριθμο, που επιλύει το πρόβλημα των "Κ πλησιέστερων ζευγών" σε παράλληλη αρχιτεκτονική, αυτής μιας κάρτας γραφικών, με χρήση του προγραμματιστικού μοντέλου CUDA. Ο αλγόριθμος, βασίζεται σε διάφορες τεχνικές των εργασιών που παρουσιαστήκαν στο Κεφάλαιο 3. Το γεγονός ότι είναι ένα ερώτημα που απαιτεί πολλούς υπολογισμούς, το καθιστά κατάλληλο για να εκτελεστεί παράλληλα σε GPU. Στην συνέχεια θα γίνει παρουσίαση της γενικής ιδέας και αναλυτική περιγραφή των μεθόδων, των φάσεων και των δομών δεδομένων που χρησιμοποιήθηκαν.

4.2 Γενική ιδέα

Η ιδέα βασίζεται στην διαδικασία της διαμέρισης σε χωρίσματα (partitions). Δεδομένων δύο συνόλων δεδομένων P και Q, το ερώτημα των "Κ πλησιέστερων ζευγών" βρίσκει τα ζευγάρια σημείων με τις Κ κοντινότερες αποστάσεις, μεταξύ όλων των πιθανών ζευγαριών σημείων, εάν επιλεχθεί ένα σημείο του P και ένα σημείο του Q. Κάθε ένα από τα δύο σύνολα δεδομένων διαμερίζεται σε χωρίσματα. Στην πρώτη φάση του αλγορίθμου, επιλέγονται εκείνα τα ζευγάρια χωρισμάτων από το P και το Q τα οποία επικαλύπτονται μεταξύ

τους. Αυτή η διαδικασία θα δειχθεί στη συνέχεια ότι συμβάλλει σημαντικά στην απόδοση του αλγορίθμου. Πρέπει να σημειωθεί, ότι τα ζευγάρια που επιλέγονται στην πρώτη φάση δεν είναι όλα από τα επικαλυπτόμενα. Επιλέγονται μόνο το 10%, το δείγμα.

Στο επόμενο βήμα, σε κάθε νήμα της κάρτας γραφικών, ανατίθεται ένα ζευγάρι από τα παραπάνω χωρίσματα που αποτελούν το δείγμα και μεταξύ αυτών των υποσυνόλων ουσιαστικά των P και Q, το κάθε νήμα εκτελεί έναν Plane Sweep αλγόριθμο, για την επιλογή των K πλησιέστερων ζευγών του.

Μετέπειτα, αφού όλα τα νήματα ολοκληρώσουν τον υπολογισμό τους, εισάγουν τις αποστάσεις των K πλησιέστερων ζευγών τους σε έναν κοινό μονοδιάστατο πίνακα. Γίνεται αύξουσα ταξινόμηση αυτού του πίνακα και επιλέγεται το K στοιχείο, ως το άνω όριο απόστασης, bound.

Στην τελευταία φάση του αλγορίθμου, εξάγονται όλα εκείνα τα ζευγάρια χωρισμάτων των P και Q των οποίων οι αποστάσεις μεταξύ τους είναι μικρότερη ή ίση του ορίου, καθώς επίσης εξάγονται και τα επικαλυπτόμενα ζευγάρια. Έπειτα, κάθε νήμα αναλαμβάνει ένα ζευγάρι χωρισμάτων, εκτελεί Plane Sweep μεταξύ αυτών και εισάγει τα K πλησιέστερα ζεύγη που έχει βρει σε έναν κοινό μονοδιάστατο πίνακα, προσβάσιμο από όλα τα νήματα των μπλοκ. Τέλος, γίνεται αύξουσα ταξινόμηση αυτού του πίνακα στον εξυπηρετητή. Στις K πρώτες θέσεις του βρίσκονται τα K πλησιέστερα ζεύγη.

4.3 Μεθοδολογία - Φάσεις αλγορίθμου

4.3.1 Διαμέριση σε χωρίσματα

Η διαμέριση σε χωρίσματα (partitions) γίνεται με την ίδια τεχνική που παρουσιάζεται στο [3]. Αναλυτικότερα, για την συγκεκριμένη υλοποίηση είναι ανάγκη να τονιστούν τα εξής:

1. Για την διαδικασία της διαμέρισης, δεν λαμβάνονται υπ'όψιν ολόκληρα τα σύνολα δεδομένων, αλλά το 30% του καθενός.

2. Η διαμέριση γίνεται ξεχωριστά για καθένα από τα δύο σύνολα P και Q, με την έννοια ότι τα χωρίσματα είναι ανεξάρτητα μεταξύ των συνόλων.

3. Τα δύο σύνολα περιέχουν εγγραφές που αποτελούνται από έναν αναγνωριστικό αριθμό (id), τη συντεταγμένη x και τη συντεταγμένη y των σημείων. Αρχικά, χρησιμοποιούνται σύνολα δύο διαστάσεων.

4. Τα δύο σύνολα αποθηκεύονται σε μονοδιάστατους πίνακες, οι οποίοι πριν την διαμέριση ταξινομούνται ως προς τη συντεταγμένη x των σημείων.

5. Το πρώτο χώρισμα κάθε συνόλου ξεκινά από το σημείο με τη μικρότερη συντεταγμένη x όλων των σημείων και τελειώνει στο πρώτο σημείο διαχωρισμού. Το τελευταίο χώρισμα, ξεκινά από το σημείο διαχωρισμού του προτελευταίου χωρίσματος και τελειώνει στο σημείο με τη μεγαλύτερη συντεταγμένη x, σημείο διαχωρισμού του τελευταίου χωρίσματος.

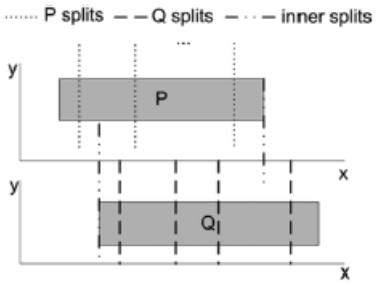
6. Ο αριθμός των συνολικών χωρισμάτων που απαιτούνται για κάθε σύνολο, δίνεται σαν είσοδος στο πρόγραμμα.

Εύρεση σημείων διαχωρισμού

Για την εύρεση των σημείων διαχωρισμού (split points), χρησιμοποιούνται οι αύξοντες ταξινομημένοι ως προς x πίνακες, P και Q των συνόλων. Ο σκοπός είναι να γίνει ο διαχωρισμός με τέτοιο τρόπο, ούτως ώστε κάθε χώρισμα να περιέχει προσεγγιστικά ίσο αριθμό σημείων. Αυτό σημαίνει ότι πιθανόν τα χωρίσματα θα έχουν διαφορετικό πλάτος, όπως φαίνεται και στο Σχήμα 4.1. Για κάθε σύνολο εξάγεται το δείγμα τους (M) και αποθηκεύεται σ'έναν πίνακα ταξινομημένο κατά τη συντεταγμένη x. Στη συνέχεια υπολογίζεται ο λόγος M/N όπου N ο συνολικός αριθμός των ζητούμενων χωρισμάτων. Τα N σημεία διαχωρισμού εξάγονται ως συγκεκριμένες τιμές της x-συντεταγμένης του συνόλου του δείγματος, όπως φαίνεται και στον ακόλουθο ψευδοκώδικα 1. Αυτά τα σημεία είναι και τα σημεία διαχωρισμού ολόκληρου του συνόλου. Σύμφωνα με το [3], αποδεικνύεται ότι αυτή η τεχνική δουλεύει αρκετά καλά, διαμορφώνοντας χωρίσματα των οποίων ο αριθμός σημείων που περιέχει το καθένα, διαφέρει περίπου 10% μεταξύ τους. Τέλος, αυτή η τεχνική εύρεσης των σημειών διαχωρισμού εφαρμόζεται για κάθε ένα από τα δύο δεδομένα σύνολα ξεχωριστά.

Algorithm 1 createSplitPoints(dataSet, N): find split points for a single dataset

```
1: sampledData = dataSet.takeSample(noReplace, M)
2: xCoord = sampledData.map(point=>point.x)
3: quicksort(xCoord)
4: step = M / N
5: splits = Array(xCoord(step), xCoord(step*2),…, xCoord(step*(N-1))
```



Σχήμα 4.1: Τα σημεία διαχωρισμού και τα χωρίσματα που προκύπτουν για κάθε σύνολο ξεχωριστά [3]

4.3.2 Εκτέλεση δείγματος και εύρεση άνω ορίου

Η επόμενη φάση του αλγορίθμου είναι αυτή της εύρεσης ενός άνω ορίου. Η διαδικασία πραγματοποιείται σε ένα δείγμα του συνόλου δεδομένων. Το ποσοστό αυτού του δείγματος σε αυτή την υλοποίηση, έχει επιλεχθεί να είναι 10% των δύο συνόλων δεδομένων. Να τονιστεί ότι το 30% που αναφέρθηκε παραπάνω, αφορά το δείγμα το οποίο χρησιμοποιείται για την εύρεση των σημείων διαχωρισμού και όχι για την εύρεση του άνω ορίου.

Αρχικά, εφόσον γίνει η επιλογή των σημείων διαχωρισμού και δημιουργηθούν τα χωρίσματα για κάθε σύνολο δεδομένων, ο σκοπός είναι να βρεθούν τα καταληλλότερα ζευγάρια χωρισμάτων από το P και το Q, ούτως ώστε το άνω όριο που θα βρεθεί να είναι όσο το δυνατόν μικρότερο. Με την έννοια καταλληλότερα, εννοείται ο αριθμός επικαλυπτόμενων χωρισμάτων αυτής της φάσης να είναι όσο γίνεται μεγαλύτερος. Με τον τρόπο αυτό υπάρχουν περισσότερες πιθανότητες για ένα μικρότερο άνω όριο. Όσο πιο μικρό είναι, τόσο λιγότεροι υπολογισμοί συμπεριλαμβάνονται στην τελευταία φάση του αλγορίθμου.

Επιλογή καταλληλότερων χωρισμάτων

Η επιλογή των καταλληλότερων χωρισμάτων γίνεται στη φάση της προεπεξεργασίας. Έστω ότι τα συνολικά χωρίσματα που δίνονται σαν είσοδος στο πρόγραμμα είναι 100. Τα χωρίσματα που είναι αναγκαία για την εκτέλεση της πρώτης φάσης και την εύρεση του άνω ορίου είναι 10 από κάθε σύνολο.

Για την αποθήκευση των χωρισμάτων, φτιάχνεται ένας πίνακας από δομές struct, στο συγκεκριμένο παράδειγμα 10 θέσεων. Αυτή η δομή struct περιέχει πέντε πεδία. Αυτά είναι τα εξής:

1. Το πρώτο πεδίο είναι ο δείκτης του αρχικού σημείου του χωρίσματος από το σύνολο P δηλαδή αυτό με τη μικρότερη συντεταγμένη x. Ο δείκτης αυτός αναφέρεται στον πίνακα του συνόλου P που περιέχει όλα τα σημεία του.

2. Το δεύτερο πεδίο είναι ο δείκτης του σημείου διαχωρισμού του χωρίσματος από το σύνολο P, αυτό με τη μεγαλύτερη συντεταγμένη x στο χώρισμα. Ο δείκτης αυτός αναφέρεται στον πίνακα του συνόλου P που περιέχει όλα τα σημεία του.

3. Το τρίτο πεδίο είναι ο δείκτης του πρώτου σημείου του χωρίσματος από το σύνολο Q, που έχει τη μικρότερη συντεταγμένη x. Ο δείκτης αυτός αναφέρεται στον πίνακα του συνόλου Q που περιέχει όλα τα σημεία του.

4. Το τέταρτο πεδίο είναι ο δείκτης του σημείου διαχωρισμού του χωρίσματος από το σύνολο Q, αυτό με τη μεγαλύτερη συντεταγμένη x στο χώρισμα. Ο δείκτης αυτός αναφέρεται στον πίνακα του συνόλου Q που περιέχει όλα τα σημεία του.

5. Το πέμπτο πεδίο είναι ένας αριθμός προτεραιότητας που φανερώνει πόσο κατάλληλο είναι το ζευγάρι χωρισμάτων P και Q, αν είναι μερικώς επικαλυπτόμενα, πλήρως επικαλυπτόμενα το ένα από το άλλο ή καθόλου. Η τιμή 2 φανερώνει ότι ένα από τα δύο χωρίσματα καλύπτεται πλήρως από το άλλο. Η τιμή 1 φανερώνει ότι ένα από τα δύο χωρίσματα καλύπτεται μερικώς από το άλλο. Τέλος, η τιμή 0 φανερώνει ότι τα χωρίσματα δεν επικαλυπτόνται μεταξύ τους. Αυτό το πεδίο είναι ουσιαστικά αυτό που προσδιορίζει την καταλληλότητα των ζευγαριών χωρισμάτων.

Φάση της επιλογής

Για τη φάση της επιλογής των κατάλληλων χωρισμάτων, λαμβάνονται υπ'όψιν τα σημεία διαχωρισμού των χωρισμάτων. Για την ακρίβεια ελέγχονται τρεις κανόνες που υποδεικνύουν τον αριθμό προτεραιότητας του κάθε ζευγαριού χωρισμάτων. Οι κανόνες είναι :

1. Ο πρώτος κανόνας ελέγχει αν το σημείο διαχωρισμού που ανήκει στο χώρισμα P είναι μικρότερο ή ίσο του αντίστοιχου σημείου διαχωρισμού του χωρίσματος που ανήκει στο Q και μεγαλύτερο ή ίσο του πρώτου σημείου του χωρίσματος που ανήκει στο Q. Με άλλα λόγια αν το σημείο διαχωρισμού του χωρίσματος που ανήκει στο P βρίσκεται εσωτερικά του χωρίσματος Q. Αν ισχύει μόνον αυτός ο κανόνας, το χώρισμα P και το χώρισμα Q είναι μερικώς επικαλυπτόμενα και έχουν προτεραιότητα 1.

2. Ο δεύτερος κανόνας ελέγχει αν το πρώτο σημείο που ανήκει στο χώρισμα P (ή αλλιώς το σημείο διαχωρισμού του προηγούμενου χωρίσματος του P από αυτό που ελέγχεται) είναι μικρότερο ή ίσο του σημείου διαχωρισμού του χωρίσματος που ανήκει στο Q και μεγαλύτερο ή ίσο του πρώτου σημείου του χωρίσματος που ανήκει στο Q. Με άλλα λόγια, αν το πρώτο σημείο του χωρίσματος που ανήκει στο P βρίσκεται εσωτερικά του χωρίσματος Q. Αν ισχύει μόνον αυτός ο κανόνας, το χώρισμα P και το χώρισμα Q είναι μερικώς επικαλυπτόμενα και έχουν προτεραιότητα 1. Αν επιπλέον ισχύει και ο πρώτος κανόνας, τότε το χώρισμα Q βρίσκεται ολόκληρο ανάμεσα στο χώρισμα P και επικαλύπτεται πλήρως από αυτό, οπότε και παίρνει προτεραιότητα 2.

3. Ο τρίτος κανόνας ελέγχει αν το σημείο διαχωρισμού που ανήκει στο χώρισμα P είναι μεγαλύτερο του αντίστοιχου σημείου διαχωρισμού του χωρίσματος που ανήκει στο Q κι αν το πρώτο σημείο που ανήκει στο χώρισμα P είναι μικρότερο από το πρώτο σημείο που ανήκει στο χώρισμα Q. Δηλαδή, σ'αυτήν την περίπτωση το χώρισμα P επικαλύπτει πλήρως το χώρισμα Q. Με την ισχύ αυτού του κανόνα, το ζευγάρι χωρισμάτων των P και Q παίρνει αμέσως προτεραιότητα 2.

Κάθε ζευγάρι χωρισμάτων που εξετάζεται, εισάγεται στον πίνακα καταλληλότερων χωρισμάτων με βάση την προτεραιότητα. Αν ο πίνακας των καταλληλότερων χωρισμάτων αποτελείται μόνον από ζευγάρια χωρισμάτων που έχουν προτεραιότητα 2, τότε η φάση της επιλογής τερματίζεται, καθώς προκύπτει η βέλτιση λύση. Διαφορετικά, θα περιέχει όσον το δυνατόν περισσότερα ζευγάρια χωρισμάτων με μεγαλύτερη προτεραιότητα.

Εκτέλεση πυρήνα με το δείγμα για την εύρεση του άνω ορίου

Πλέον κι αφού έχουν επιλεχθεί τα καταλληλότερα ζευγάρια χωρισμάτων, αυτά πρέπει να ανατεθούν σε νήματα. Ο πίνακας που περιέχει τους δείκτες γι'αυτά τα χωρίσματα, μεταφέρεται στη μνήμη της κάρτας γραφικών, όπως και τα δύο σύνολα που περιέχουν το σύνολο των σημείων για την επίλυση του ερωτήματος.

Τα νήματα που θα προκύψουν εξαρτώνται από τον αριθμό των χωρισμάτων του δείγματος. Στο συγκεκριμένο παράδειγμα 10 νήματα θα εκτελεστούν παράλληλα. Κάθε νήμα για την αποθήκευση των K πλησιέστερων ζευγών των χωρισμάτων των συνόλων P και Q που έχει αναλάβει, χρησιμοποιεί έναν πίνακα K θέσεων που αποθηκεύεται στην τοπική μνήμη νήματος, τον localMaxKHeap. Η ανάθεση των χωρισμάτων στα νήματα γίνεται με βάση του παραπάνω πίνακα των καταλληλότερων χωρισμάτων.

Μεταξύ αυτών των χωρισμάτων εκτελείται ένας αλγόριθμος Σάρωσης Επιπέδου (Plane Sweep), που θα αναλυθεί καλύτερα στη δεύτερη φάση, για την εύρεση των K πλησιέστερων ζευγών. Αφού το κάθε νήμα τελειώσει τον υπολογισμό των K πλησιέστερων ζευγών, εισάγει τις αποστάσεις τους σε έναν άλλο πίνακα που αποθηκεύεται στην καθολική μνήμη της κάρτας γραφικών, τον globalMaxKHeap. Ο πίνακας αυτός περιέχει K επί τον αριθμό των νημάτων, αποστάσεις. Με την εισαγωγή όλων αυτών των αποστάσεων ολοκληρώνεται η εκτέλεση του πυρήνα με το δείγμα.

Ο globalMaxKHeap αντιγράφεται στη μνήμη του εξυπηρετητή. Εφαρμόζεται ο αλγόριθμος quicksort για την ταξινόμηση των αποστάσεων. Το άνω όριο είναι το στοιχείο που βρίσκεται στην K θέση αυτού του πίνακα.

4.3.3 Εκτέλεση κύριας φάσης

Πριν την εκτέλεση της κύριας φάσης, είναι γνωστό το άνω όριο. Με βάση αυτό το όριο, προηγείται η φάση της προεπεξεργασίας στον εξυπηρετητή, όπου γίνεται η επιλογή όλων εκείνων των πιθανών ζευγαριών χωρισμάτων από το P και το Q που είναι υποψήφια να περιέχουν κάποιο από τα K πλησιέστερα ζεύγη. Θα δειχθεί στην συνέχεια, πως αυτό το όριο επηρεάζει την απόδοση.

Επιλογή όλων των πιθανών ζευγαριών χωρισμάτων

Για την επιλογή όλων των πιθανών ζευγαριών χωρισμάτων όπως αναφέρθηκε και παραπάνω, λαμβάνεται υπ'όψιν το άνω όριο. Με την έννοια πιθανά ζευγάρια χωρισμάτων εννοούνται όλα εκείνα τα ζευγάρια χωρισμάτων από το P και το Q τα οποία, πρέπει και είναι αναγκαίο να συμπεριληφθούν στον υπολογισμό των K πλησιέστερων ζευγών. Σ'αυτά περιλαμβάνονται τα ζευγάρια χωρισμάτων από το P και το Q που είναι πλήρως ή μερικώς επικαλυπτόμενα καθώς κι εκείνα που απέχουν μεταξύ τους απόσταση μικρότερη ή ίση του άνω ορίου. Με αυτόν τον τρόπο και με την εκτέλεση προηγουμένως του δείγματος, έχει γίνει περιορισμός άπαντων των συνόλων δεδομένων που θα συμπεριληφθούν στον υπολογισμό, με σκοπό τη βελτίωση της ταχύτητας απάντησης του ερωτήματος.

Όλα τα πιθανά χωρίσματα βρίσκονται με παρόμοιο τρόπο, μ'αυτόν που ανακαλύπτονται τα καταλληλότερα χωρίσματα στη φάση εκτέλεσης του δείγματος και συγκεκριμένα στη φάση επιλογής των κατάλληλων χωρισμάτων. Οι τρεις κανόνες που θα πρέπει να ισχύουν εκεί, είναι ίδιοι με αυτούς που θα πρέπει να ισχύουν και στην προκειμένη περίπτωση, για

την εύρεση όλων των πιθανών ζευγαριών χωρισμάτων. Παρόλα αυτά παρουσιάζονται κι οι τέσσερις:

1. Ο πρώτος κανόνας είναι ίδιος με τον πρώτο της φάσης επιλογής των καταλληλότερων χωρισμάτων. Πλην όμως τα χωρίσματα δεν παίρνουν πλέον αριθμό προτεραιότητας. Επίσης, για τα υποψήφια χωρίσματα από το P και το Q που ελέγχονται δε χρειάζεται να ελεγχθούν και οι υπόλοιποι κανόνες, εφόσον ισχύει ο πρώτος κανόνας. Δηλαδή, αν είναι μερικώς επικαλυπτόμενα, τότε θα πρέπει σίγουρα να συμπεριληφθούν στον υπολογισμό.

2. Το ίδιο ισχύει και για τον δεύτερο κανόνα αντίστοιχα.

3. Παρομοίως και για τον τρίτο.

4. Ο τέταρτος κανόνας αφορά τα ζευγάρια χωρισμάτων από το P και το Q τα οποία δεν επικαλύπτονται. Θα πρέπει να συμπεριληφθούν λοιπόν στον υπολογισμό, όλα εκείνα για τα οποία είτε το σημείο διαχωρισμού του χωρίσματος P απέχει απόσταση μικρότερη ή ίση από το πρώτο σημείο του χωρίσματος του Q, είτε το πρώτο σημείο του χωρίσματος P απέχει απόσταση μικρότερη ή ίση από το σημείο διαχωρισμού του χωρίσματος Q, αναλόγως αν το χώρισμα P βρίσκεται αριστερά ή δεξιά από το χώρισμα Q αντίστοιχα. Η απόσταση, εννοείται η απόσταση των x συντεταγμένων τους.

Όλα αυτά τα πιθανά χωρίσματα που ανακαλύπτονται, αποθηκεύονται σε έναν πίνακα από δομές struct. Αυτός ο πίνακας δεν είναι γνωστός εκ των προτέρων και γι'αυτόν τον λόγο αυξάνεται δυναμικά, με κάθε νέο ζευγάρι χωρισμάτων που βρίσκεται και πρέπει να συμπεριληφθεί στον υπολογισμό. Η δομή struct είναι η ίδια με αυτήν της φάσης επιλογής των καταλληλότερων χωρισμάτων πριν την εκτέλεση του δείγματος, χωρίς όμως το πεδίο με τον αριθμό προτεραιότητας.

Ανάθεση ζευγαριών χωρισμάτων σε νήματα

Αφού βρεθούν όλα τα ζευγάρια χωρισμάτων που θα πρέπει να συμπεριληφθούν στον υπολογισμό, αποφασίζεται ο αριθμός των νημάτων που θα υπολογίσουν τα Κ πλησιέστερα ζεύγη. Η διαδικασία αυτή γίνεται στη φάση της προεπεξεργασίας στον εξυπηρετητή. Γενικά, τα νήματα που θα εκτελεστούν θα είναι όσα και τα ζευγάρια χωρισμάτων που υπολογίστηκαν προηγουμένως. Θα πρέπει να τονιστεί όμως, ότι εάν τα ζευγάρια χωρισμάτων είναι λιγότερα

από 256, τότε ο πυρήνας στην κύρια φάση θα εκτελεστεί με ένα μπλοκ νημάτων. Αν τα ζευγάρια χωρισμάτων υπερβαίνουν τα 256 τότε προστίθεται ένα επιπλέον μπλοκ νημάτων. Αν τα ζευγάρια χωρισμάτων υπερβαίνουν τα 512 τότε θα προκύψουν τρία μπλοκ νημάτων κ.ο.κ. Ο σκοπός είναι κάθε μπλοκ νημάτων να περιέχει 256 νήματα. Ο αριθμός 256, επιλέχτηκε με γνώμονα την βελτιστοποιημένη απόδοση των μπλοκ νημάτων και κατά συνέπεια της κάρτας γραφικών. Τα μπλοκ συνίσταται να μην είναι γεμάτα, συνήθως με 1024 νήματα.

Εκτέλεση πυρήνα και αλγόριθμος "Σάρωσης Επιπέδου"

Το επόμενο βήμα κι αφού έχουν βρεθεί όλα τα πιθανά ζευγάρια χωρισμάτων αλλά και έχει αποφασιστεί ο αριθμός των μπλοκ και των νημάτων, είναι η εκτέλεση του πυρήνα. Στη μνήμη της κάρτας γραφικών, μεταφέρεται ο πίνακας των πιθανών ζευγαριών χωρισμάτων που περιέχει τους δείκτες στον πίνακα συνόλων, αρχής και τέλους κάθε χωρίσματος καθώς και ένας μονοδιάστατος πίνακας phase2_globalMaxKHeap, μεγέθους K (τα K πλησιέστερα ζεύγη που αναζητούνται) επί τον αριθμό των νημάτων που θα εκτελεστούν στον πυρήνα. Αυτός ο πίνακας αποθηκεύεται στην καθολική μνήμη της CUDA, που σημαίνει ότι όλα τα νήματα απ'όλα τα μπλοκ έχουν πρόσβαση σ'αυτήν. Το κάθε νήμα, μετά την εκτέλεση ενός plane sweep αλγορίθμου ανάμεσα στα χωρίσματα από το P και το Q που έχει αναλάβει, εισάγει τα K πλησιέστερα ζεύγη που έχει βρει στον πίνακα που βρίσκεται στην καθολική μνήμη.

Όπως αναφέρθηκε προηγουμένως, το κάθε νήμα εκτελεί έναν αλγόριθμο "Σάρωσης Επιπέδου" για την επιλογή των K πλησιέστερων ζευγών του, για το υποσύνολο των σημείων που έχει αναλάβει. Η επιλογή αυτού του αλγορίθμου έγινε από τους plane sweep αλγορίθμους που παρουσιάζονται στο [5]. Σύμφωνα με το άρθρο, το όνομα της τεχνικής αυτής προήλθε από την ιδέα της σάρωσης της επιφάνειας από τα αριστερά προς τα δεξιά, με μια κάθετη γραμμή, η οποία σταματάει σε κάθε σημείο που πρόκειται να εξεταστεί. Η όλη διαδικασία στηρίζεται σ'αυτήν την κάθετη γραμμή, χωρίς καμία οπισθοδρόμηση και προχωρόντας μόνο προς τα μπρος εξετάζοντας ένα σημείο κάθε φορά. Αυτή η τεχνική έχει εφαρμοστεί επιτυχώς στην επεξεργασία χωρικών ερωτημάτων.

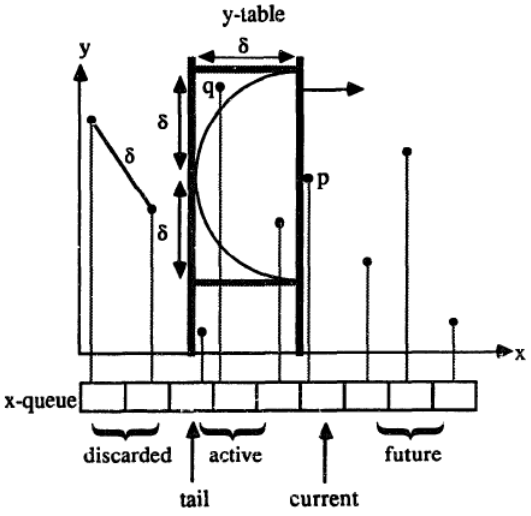
Στο [4] παρουσιάζεται ο κλασικός Plane Sweep για την επεξεργασία του χωρικού ερωτήματος των "K πλησιέστερων ζευγών". Σ'αυτήν την εργασία αναζητούνται τα K πλησιέστερα ζεύγη για ένα σύνολο δεδομένων, δύο διαστάσεων.Το κάθε εξεταζόμενο σημείο διατηρείται σε μια ουρά, x-queue. Τα σημεία που σαρώνονται διατηρούνται σ' έναν πίνακα y-table. Τα

σημεία που είναι υποψήφια για πλησιέστερα ζεύγη βρίσκονται στα αριστερά του σημείου που έχει επιλεχθεί και βρίσκεται στη νοητή κάθετη γραμμή προς εξέταση (βλπ. Σχήμα 4.2). Ειδικότερα εξετάζονται τα σημεία που βρίσκονται σε μια απόσταση δ, η οποία αρχικά είναι ένας πολύ μεγάλος αριθμός, όσο όμως προχωράει η επεξεργασία αυτός ο αριθμός μειώνεται δραστικά, περιορίζοντας έτσι και τα σημεία που ελέγχονται. Στο Σχήμα 4.2 φαίνεται το προς εξέταση σημείο p και σε απόσταση δ τα σημεία από τα οποία υπολογίζονται οι αποστάσεις και είναι υποψήφια για K πλησιέστερα ζεύγη. Τα σημεία προς εξέταση μπορεί να βρίσκονται είτε στο ημικύκλιο με ακτίνα δ, είτε για περισσότερη ευκολία στο ορθογώνιο με πλάτος δ. Είναι λογικό, ότι στην δεύτερη περίπτωση θα περιέχονται και κάποια περιττά σημεία. Τα ενεργά σημεία είναι αυτά που βρίσκονται είτε μέσα στο ορθογώνιο, είτε μέσα στο ημικύκλιο. Αυτά που είναι εκτός εξαιρούνται από τον υπολογισμό. Στο y-table περιέχονται μόνον τα σημεία που είναι εντός απόστασης δ και είναι ταξινομημένα κατά τη συντεταγμένη y. Εάν στο y-table υπάρχουν σημεία, σημαίνει ότι η απόσταση τους από το p είναι μικρότερη από το δ, επομένως γίνεται ανανέωση του δ με την απόσταση αυτή, για την αναζήτηση ενός πλησιέστερου ζεύγους. Ανακεφαλαιώνοντας, το σύνολο σημείων S χωρίζεται σε τέσσερα μη επικαλυπτόμενα υποσύνολα, όπως φαίνεται και στο Σχήμα 4.2 :

- Τα σημεία που παραλείπονται από τον υπολογισμό και βρίσκονται αριστερά από τον δείκτη “tail”(ο οποίος δείχνει τον πίνακα x-table) και πλέον είναι μη προσβάσιμα.
- Τα ενεργά σημεία ανάμεσα στο δείκτη “tail” και “current” που είναι αυτά που εξετάζονται και εισάγονται στο y-table.
- Το σημείο το οποίο εξετάζεται τη δεδομένη στιγμή, p.
- Τα μελλοντικά σημεία προς εξέταση και τα οποία δεν έχουν ακόμα ελεγχθεί.

Συγκρίνοντας την τεχνική αυτή με εκείνην του “διαιρεί και βασίλευε”, ο plane sweep αλγόριθμος είναι πολύ πιο αποδοτικός για τους ακόλοθους λόγους:

1. Ως επαναληπτικός αλγόριθμος δεν παρουσιάζει επιπλέον κόστος χρόνου και χώρου στο να διατηρεί στοίβα η οποία προκύπτει στους αναδρομικούς αλγόριθμους.
2. Με την ιδιότητα που έχει να λειτουργεί σε στάδια, έχει το πλεονέκτημα ότι ενημερώνει την τωρινή κατάσταση προσθέτωντας ένα νέο σημείο προς υπολογισμό ευκολότερα, από το να συνδυάσει δύο σύνολα $\frac{n}{2}$ σημείων, όπως θα γινόταν σ’έναν αναδρομικό αλγόριθμο.



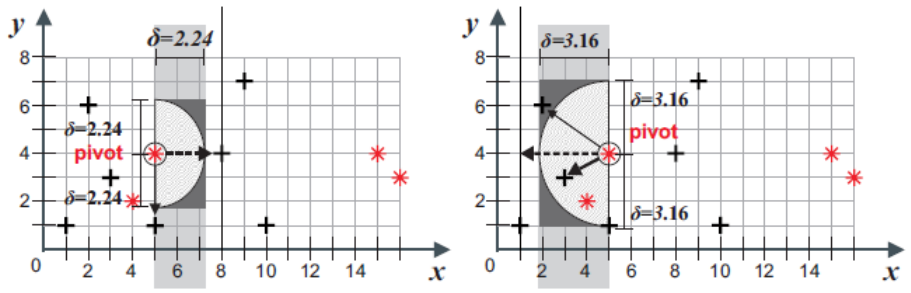
Σχήμα 4.2: Η μέθοδος σάρωσης επιπέδου, όπως παρουσιάστηκε πρώτα στο [4]

Η υλοποίηση που χρησιμοποιήθηκε στην παρούσα εργασία, είναι μια βελτιωμένη έκδοση της προαναφερθείσας και παρουσιάζεται στο [5]. Οι συγγραφείς εφαρμόζουν την τεχνική για το ερώτημα "Κ πλησιέστερων ζευγών" μεταξύ δυο συνόλων δεδομένων. Τονίζεται ότι ο κλασικός plane sweep αρχικά προοριζόταν για το χωρικό ερώτημα για ένα σύνολο σημείων. Η βελτίωση που προτείνεται στο άρθρο αυτό ονομάζεται "Reverse Run Plane-Sweep" ή "RRPS" και ελαχιστοποιεί της ευκλίδειες αλλά και τις αποστάσεις που υπολογίζονται κατά τη διάρκεια της σάρωσης. Οι βελτιώσεις όπως φαίνεται και στο Σχήμα 4.3 συνοψίζονται στις παρακάτω :

1. Γίνεται περιορισμός των σημείων που εξετάζονται όχι μόνον ως προς τον άξονα σάρωσης των σημείων, αλλά και ως προς τον άλλον άξονα με βάση το δ. Η βελτιώση αναφέρεται σαν "συρόμενο παράθυρο" ή καλύτερα στα αγγλικά "sliding window" κατά τον άξονα σάρωσης, με πλάτος δ και ύψος ίσο με 2 επί δ. Έτσι επιλέγονται μόνον τα σημεία του άλλου συνόλου που βρίσκονται σ'αυτό το "sliding window".

2. Η δεύτερη βελτίωση για την περαιτέρω μείωση του χώρου αναζήτησης, αφορά την επιλογή των σημείων που βρίσκονται σ'ένα ημικύκλιο ακτίνας δ. Η τεχνική αυτή ονομάζεται "sliding semi-circle" με ακτίνα δ.

Το σημείο που εξετάζεται ανήκει στο σύνολο P ενώ τα σημεία που περιέχονται στο "sliding window" ή στο "sliding semi-circle" (ημικύκλιο) ανήκουν στο σύνολο Q. Στην παρούσα διπλωματική εργασία επιλέχτηκε η βελτίωση με το "sliding window".



Σχήμα 4.3: Στα αριστερά ο κλασικός Plane Sweep και στα δεξιά ο Reverse Run Plane Sweep, χρησιμοποιώντας τις εκδοχές "sliding window" και "sliding semi-circle" [5]

Ο αλγόριθμος αυτός διαφέρει σε κάποια μέρη με αυτόν που παρουσιάστηκε στο [4]. Αρχικά, κάθε σημείο που εξετάζεται ονομάζεται σημείο αναφοράς (pivot) και με μεταγενέστερα σημεία του ίδιου συνόλου αποτελούν ενεργά σημεία κι εφόσον μεταξύ τους δεν περιέχονται σημεία του άλλου συνόλου. Για κάθε σύνολο, διατηρείται ένα αριστερό όριο, το οποίο ανανεώνεται προς τα δεξιά, κάθε φορά που ο αλγόριθμος περιλαμβάνει σημεία που είναι απαραίτητα για σύγκριση με τα σημεία αυτού του συνόλου που βρίσκονται στα δεξιά του ίδιου ορίου. Κάθε σημείο των ενεργών σημείων συγκρίνεται με κάθε σημείο του άλλου συνόλου που είναι στα αριστερά από το πρώτο σημείο των ενεργών σημείων, μέχρι το αριστερό όριο του άλλου συνόλου να βρεθεί. Δεύτερον, τα σημεία αναφοράς (και τα ενεργά σημεία τους) επεξεργάζονται σε αύξουσα σειρά (τα σύνολα πρέπει να είναι ταξινομημένα ως προς την x-συντεταγμένη τους πριν την εφαρμογή του RRPS). Κάθε σημείο από τα ενεργά σημεία συγκρίνεται με τα σημεία του άλλου συνόλου με την αντίθετη φορά (κατά την φθίνουσα σειρά των x-συντεταγμένων τους). Η πειραματική αξιολόγηση του αλγορίθμου RRPS αποδεικνύει ότι είναι περισσότερο αποδοτικός σε σχέση με τον κλασικό Plane Sweep και η βελτίωση με το ημικύκλιο είναι αυτή που χαρακτηρίζεται ως η πιο αποτελεσματική.

Με βάση την παραπάνω έρευνα, τα νήματα στην κύρια φάση του παρόντος αλγορίθμου εφαρμόζουν την τεχνική "RRPS" με την εκδοχή του "sliding window", μεταξύ των χωρισμάτων από τα σύνολα P και Q για την εύρεση των K πλησιέστερων ζευγών τους.

Επιλογή K πλησιέστερων ζευγών

Το κάθε ένα νήμα κατά τη διαδικασία εκτέλεσης του αλγορίθμου "Σάρωσης Επιπέδου", διατηρεί έναν πίνακα στην τοπική του μνήμη (localMaxKHeap) όπου αποθηκεύει τα ζεύγη σημείων που βρίσκει. Αφού ολοκληρωθεί ο "Plane Sweep", αυτός ο τοπικός πίνακας έχει

πλέον τα K πλησιέστερα ζεύγη του ζευγαριού χωρισμάτων από τα σύνολα P και Q. Στη συνέχεια, αυτά εισάγονται σε πίνακα της καθολικής μνήμης του μοντέλου CUDA απ'όλα τα νήματα και αφού ολοκληρώσουν την εισαγωγή, τερματίζεται ο πυρήνας και ο πίνακας αντιγράφεται στην μνήμη του εξυπηρετητή, όπου και ταξινομείται κατά αύξουσα σειρά. Μετά απο διαδοχική εκτέλεση πειραμάτων με τους αλγορίθμους quick-sort, insertion-sort, bubble-sort, selection-sort και merge-sort, διαπιστώθηκε ότι η καλύτερη επιλογή αλγορίθμου για ταξινόμηση είναι ο quick-sort, καθώς έχει τους μικρότερους χρόνους εκτέλεσης. Στις K πρώτες θέσεις του πίνακα βρίσκονται τα K πλησιέστερα ζεύγη.

Κεφάλαιο 5

Πειραματική Αξιολόγηση

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της πειραματικής αξιολόγησης του αλγορίθμου. Η σύγκριση έγινε μεταξύ των ακόλουθων τεσσάρων υλοποιήσεων :

- Παράλληλος αλγόριθμος "Κ πλησιέστερων ζευγών" με μεταφορά ολόκληρων των συνόλων σημείων στη μνήμη της κάρτας γραφικών (Parallel Whole Dataset In Device Memory (PWD)).
- Παράλληλος αλγόριθμος "Κ πλησιέστερων ζευγών" με μερική μεταφορά των συνόλων σημείων στη μνήμη της κάρτας γραφικών (Parallel Partial Dataset In Device Memory (PPD)).
- Σειριακός αλγόριθμος "Reverse Run Plane Sweep" (RRPS).
- Σειριακός αλγόριθμος "Brute Force" (BF).

Τα πειράματα περιλαμβάνουν συνθετικά αλλά και πραγματικά σύνολα δεδομένων. Διερευνάται η διαφορά στην ταχύτητα εκτέλεσης των παραπάνων αλγορίθμων καθώς επίσης πως επηρεάζουν ο αριθμός "Κ" των πλησιέστερων ζευγών, ο αριθμός των χωρισμάτων αλλά και το μέγεθος των συνόλων σημείων εισόδου, την απόδοση τους.

5.1 Υπολογιστικό σύστημα αξιολόγησης

Η πειραματική αξιολόγηση του αλγορίθμου πραγματοποιήθηκε σε σύστημα με επεξεργαστή Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz, λειτουργικό σύστημα Linux και κάρτα

γραφικών NVIDIA Quadro P400 (πυρήνες CUDA 256), ρολόϊ γραφικών 1252MHz, καθολική μνήμη 1992MB, κοινόχρηστη μνήμη σε κάθε μπλοκ 48MB και διαθέσιμος αριθμός καταχωρητών για κάθε νήμα 65536.

5.2 Προγραμματιστικό περιβάλλον και εκτέλεση πειραμάτων

Για τον προγραμματισμό των αλγορίθμων και την εκτέλεση των πειραμάτων απαραίτητη είναι η βιβλιοθήκη CUDA. Επίσης, προϋπόθεση αποτελεί η μονάδα επεξεργασίας γραφικών να είναι του κατασκευαστή Nvidia.

Για την εκτέλεση των πειραμάτων αρκεί το τερματικό ενός συστήματος με το λειτουργικό Linux. Για τον παράλληλο αλγόριθμο των "Κ πλησιέστερων ζευγών" οι παράμετροι εισόδου είναι:

1. Ο αριθμός "Κ" των πλησιέστερων ζευγών που αναζητείται.
2. Το αρχείο εισόδου που αποτελεί το πρώτο σύνολο P των σημείων.
3. Το αρχείο εισόδου που αποτελεί το δεύτερο σύνολο Q των σημείων.
4. Ο αριθμός των χωρισμάτων για την διαμέριση των δύο συνόλων P και Q.
5. Το σύνολο των σημείων του συνόλου P.
6. Το σύνολο των σημείων του συνόλου Q.

Μία ενδεικτική εντολή εκτέλεσης είναι: "./main 10 P_gaussian_500K.txt Q_gaussian_500K.txt 2850 500000 500000"

5.3 Δομή αρχείων δεδομένων εισόδου

Τα αρχεία που χρησιμοποιήθηκαν έχουν την μορφή αρχείων κειμένου (.txt) ή αρχείων με τιμές διαχωρισμένες με κόμμα (.csv). Η δομή των πρώτων αφορά τα συνθετικά σύνολα δεδομένων και κάθε γραμμή αποτελείται από την τιμή συντεταγμένης x και την τιμή τη συντεταγμένης y του σημείου, χωρισμένες με κόμμα. Η δομή των αρχείων με κατάληξη ".csv" που αφορά τα πραγματικά σύνολα δεδομένων, αποτελείται από τον αναγνωριστικό αριθμό

(id), τη συντεταγμένη x και τη συντεταγμένη y του σημείου σε κάθε γραμμή και χωρίζονται με χαρακτήρα “tab”.

5.4 Ανάλυση πειραμάτων

Αρχικά, γίνεται ανάλυση των πειραμάτων για τις δύο εκδοχές του Παράλληλου Αλγορίθμου “Κ πλησιέστερων ζευγών” και πως επηρεάζει ο αριθμός των χωρισμάτων την απόδοση τους. Επίσης, παρουσιάζεται ο βέλτιστος αριθμός χωρισμάτων για κάθε εκδοχή που έχει σαν αποτέλεσμα την ταχύτερη εκτέλεση του αλγορίθμου.

Στο δεύτερο σκέλος της αξιολόγησης, γίνεται σύγκριση των καλύτερων χρόνων των τεσσάρων αλγορίθμων “RRPS”, “BF”, “PPD” και “PWD” μεταβάλλοντας διάφορες παραμέτρους. Συγκεκριμένα, μελετάται πως επηρεάζει ο αριθμός “K” των πλησιέστερων ζευγών την απόδοση αλλά και πως το σύνολο των σημείων εισόδου.

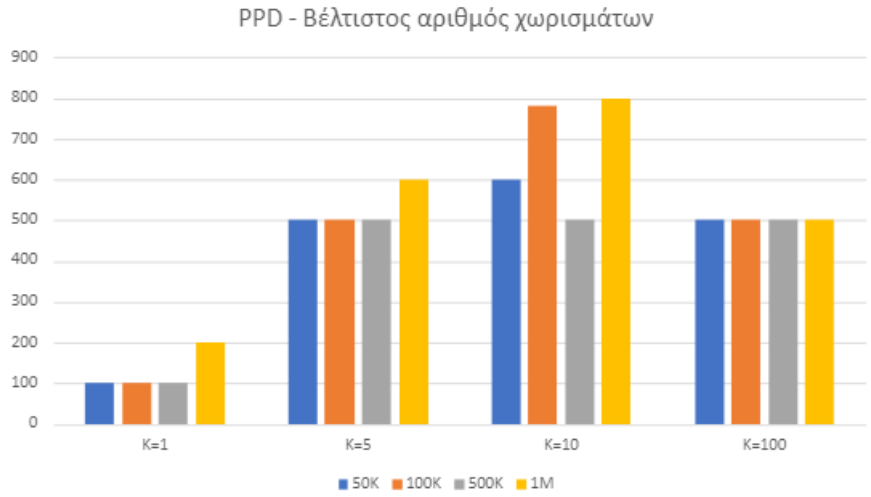
5.5 Αξιολόγηση με συνθετικά σύνολα δεδομένων

Τα αρχεία των συνόλων με συνθετικά δεδομένα αποτελούνται από 50.000 (50K), 100.000 (100K), 500.000 (500K) και 1.000.000 (1M) σημεία. Αυτά των 50K, 100K ακολουθούν διαγώνια κατανομή ενώ αυτά των 500K και 1M ακολουθούν γκαουσιανή κατανομή. Οι τιμές των συντεταγμένων x και y είναι κανονικοποιημένες στο διάστημα [0,1]. Τα αρχεία αυτά με τα σύνολα σημείων δημιουργήθηκαν με το λογισμικό “SpiderWeb” που βρίσκεται στην ιστοσελίδα `https://spider.cs.ucr.edu/#`. Τα πειράματα εκτελέστηκαν για την αναζήτηση 1, 5, 10 και 100 πλησιέστερων ζευγών.

5.5.1 Βέλτιστος αριθμός χωρισμάτων για τον Παράλληλο Αλγόριθμο “Κ πλησιέστερων ζευγών”

Τα σύνολα P και Q έχουν το ίδιο πλήθος σημείων και αναζητούνται οι 1, 5, 10 και 100 πλησιέστεροι γείτονες. Για την σύγκριση με τους αλγορίθμους RRPS και BF, πρέπει προηγουμένως να βρεθεί ο αριθμός χωρισμάτων για την διαμέριση των συνόλων που αποφέρει την βέλτιστη απόδοση δηλαδή τον μικρότερο χρόνο εκτέλεσης.

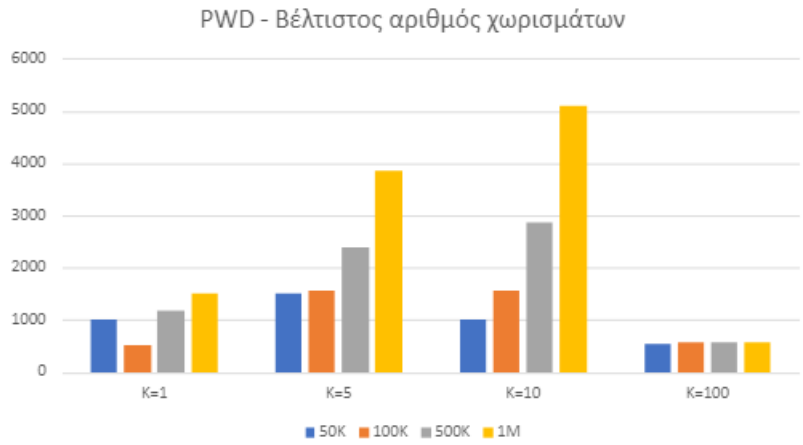
Στο Σχήμα 5.1 φαίνεται πως συμπεριφέρεται ο Παράλληλος Αλγόριθμος με μερική μεταφορά των συνόλων σημείων στην κάρτα γραφικών, για διάφορες τιμές του αριθμού χωρισμά-



Σχήμα 5.1: Ο βέλτιστος αριθμός χωρισμάτων για διάφορα σύνολα δεδομένων P,Q για τον παράλληλο αλγόριθμο με μερική μεταφορά των συνόλων σημείων στην μνήμη της κάρτας γραφικών.

των για την διαμέριση των συνόλων P και Q. Για μικρά "K" και μικρά σύνολα δεδομένων ο αριθμός των χωρισμάτων είναι μικρότερος, ενώ καθώς αυξάνει το "K" αυξάνει και ο αριθμός των χωρισμάτων που βελτιστοποιεί την απόδοση. Παρατηρείται επίσης, πως για 100 πλησιέστερα ζεύγη η αύξηση του αριθμού χωρισμάτων δεν ελαχιστοποιεί τον χρόνο εκτέλεσης. Αυτό συμβαίνει λόγω του πίνακα phase2_globalMaxKHeap που είναι όλο και μεγαλύτερος αφού κάθε νήμα χρειάζεται 100 θέσεις σε αυτόν τον πίνακα. Αυτός ο πίνακας, μεταφέρεται ολόκληρος στην μνήμη της κάρτας γραφικών γεγονός που καθυστερεί την εκτέλεση λόγω της μεταφοράς. Όταν αυξάνει ο αριθμός των χωρισμάτων, αυξάνουν και τα πιθανά ζεύγη χωρισμάτων που σημαίνει ότι αυξάνονται και τα νήματα που θα εκτελεστούν.

Στο Σχήμα 5.2, φαίνεται η συμπεριφορά του Παράλληλου Αλγορίθμου με μεταφορά ολόκληρων των συνόλων σημείων στην μνήμη της μονάδας επεξεργασίας γραφικών. Και εδώ παρατηρείται ότι για μικρά "K" ο αριθμός των χωρισμάτων είναι μικρός και καθώς μεγαλώνουν τα σύνολα P,Q αλλά και το "K", αυξάνουν και τα βέλτιστα χωρίσματα. Επίσης, αξίζει να σημειωθεί ότι τα χωρίσματα σε αυτήν την εκδοχή του αλγορίθμου είναι πολύ περισσότερα, σχεδόν πάνω από 1000. Αυτό συμβαίνει επειδή δεν απαιτείται προεπεξεργασία για την ανακάλυψη των υποσυνόλων που θα μεταφερθούν στην μνήμη της κάρτας γραφικών όπως γίνεται στην εκδοχή "PPD", παρέχοντας μια ευελιξία στην εύρεση του αριθμού χωρισμάτων που αποφέρουν ταχύτερους χρόνους. Τέλος, για 100 πλησιέστερα ζεύγη υπάρχει καθυστέρηση λόγω μεταφορών όπως αναφέρθηκε και στην παραπάνω εκδοχή, που επηρεά-



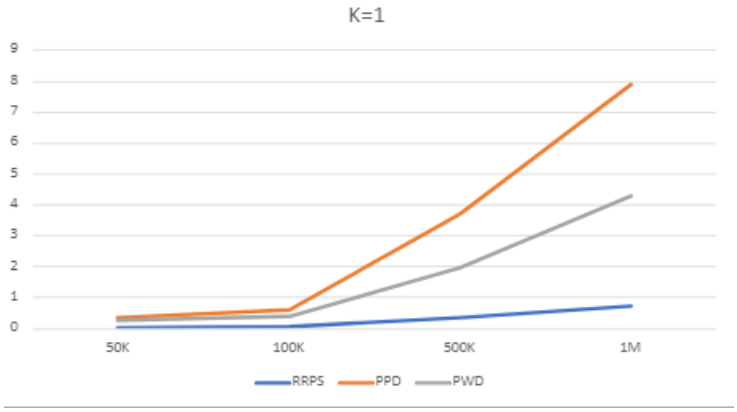
Σχήμα 5.2: Ο βέλτιστος αριθμός χωρισμάτων για διάφορα σύνολα δεδομένων P,Q για τον παράλληλο αλγόριθμο με μεταφορά ολόκληρων των συνόλων σημείων στην μνήμη της κάρτας γραφικών.

ζει σημαντικά την απόδοση.

5.5.2 Σύγκριση σειριακών με τον Παράλληλο Αλγόριθμο

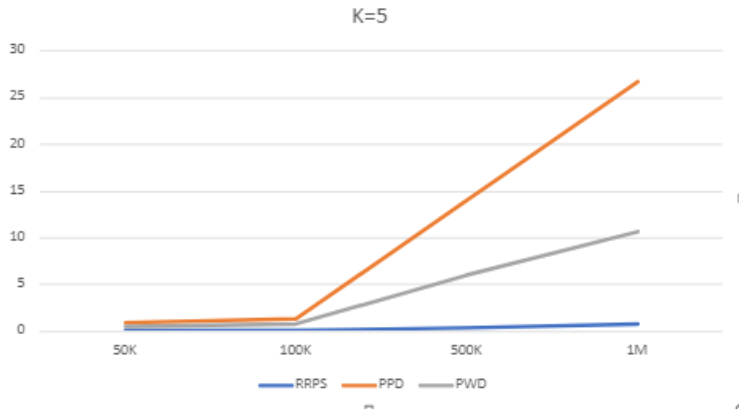
Επίδραση του μεγέθους των συνόλων σημείων

Μελετήθηκε πρώτα πως επηρεάζει η αύξηση των συνόλων σημείων P,Q τους χρόνους εκτέλεσης. Ο σειριακός "Brute Force", παρουσιάζει πολύ μεγαλύτερους χρόνους εκτέλεσης όπως φαίνεται και στον πίνακα 5.1. Γι' αυτό τον λόγο τελικά δεν συμπεριλήφθηκε στα διαγράμματα, για να υπάρχει πιο ξεκάθαρη εικόνα μεταξύ των άλλων αλγορίθμων.

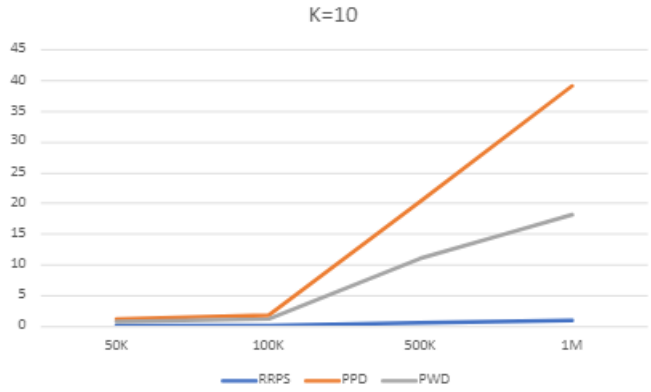


Σχήμα 5.3: Οι χρόνοι εκτέλεσης σε δευτερόλεπτα σε σχέση με τα σύνολα σημείων για K=1

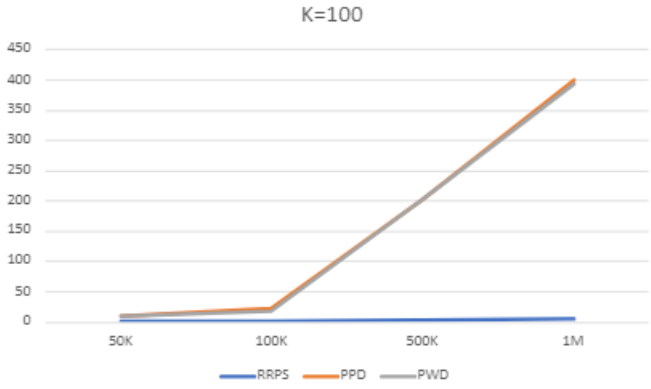
Παρατηρείται ότι για μικρό αριθμό "K", ο αλγόριθμος "PWD" είναι καλύτερος. Αυτό οφείλεται στο ότι υπάρχει μεγαλύτερη ευελιξία στην επιλογή των χωρισμάτων γεγονός που



Σχήμα 5.4: Οι χρόνοι εκτέλεσης σε δευτερόλεπτα σε σχέση με τα σύνολα σημείων για K=5



Σχήμα 5.5: Οι χρόνοι εκτέλεσης σε δευτερόλεπτα σε σχέση με τα σύνολα σημείων για K=10



Σχήμα 5.6: Οι χρόνοι εκτέλεσης σε δευτερόλεπτα σε σχέση με τα σύνολα σημείων για K=100

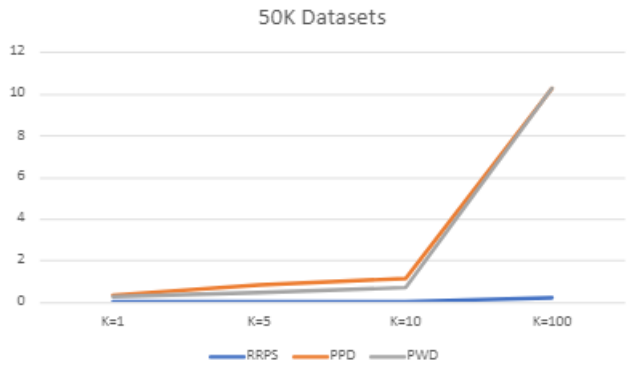
προσφέρει καλύτερους χρόνος από τον “PPD”. Η προεπεξεργασία που απαιτείται στον δεύτερο για την επιλογή των υποσυνόλων που θα μεταφερθούν στην μνήμη καθυστερεί την εκτέλεση και η επιλογή ενός μεγάλου αριθμού χωρισμάτων την καθυστερεί ακόμα περισσότερο. Όπως φαίνεται και στα Σχήματα 5.3, 5.4 και 5.5 για 1, 5 και 10 πλησιέστερα ζεύγη καθώς αυξάνονται τα σύνολα σημείων ο “PWD” αποδίδει όλο και καλύτερα. Τέλος, για 100

πλησιέστερα ζεύγη, οι δύο εκδοχές του Παράλληλου Αλγορίθμου "Κ πλησιέστερων ζευγών" έχουν παρόμοια συμπεριφορά (βλπ. Σχήμα 5.6).

Επίδραση του αριθμού "Κ" πλησιέστερων ζευγών

Σε αυτή τη σειρά συγκρίσεων διατηρείται σταθερό το πλήθος των σημείων των συνόλων P και Q, καθώς το "Κ" μεταβάλλεται. Στα Σχήματα 5.7, 5.9 και 5.10 φαίνεται ότι καθώς το "Κ" αυξάνει ο "PWD" εκτελείται γρηγορότερα. Όταν όμως το "Κ" αρχίζει να γίνεται πολύ μεγαλύτερο οι δύο εκδοχές του παράλληλου αλγορίθμου έχουν ίδιους χρόνους. Αξίζει να σημειωθεί, ότι εξαίρεση αποτελεί το μέγεθος συνόλων σημείων 100Κ (βλπ. Σχήμα 5.8). Εκεί, καθώς τα πλησιέστερα ζεύγη που αναζητώνται αυξάνονται η εκδοχή με μεταφορά ολόκληρων των συνόλων σημείων έχει όλο και καλύτερη απόδοση.

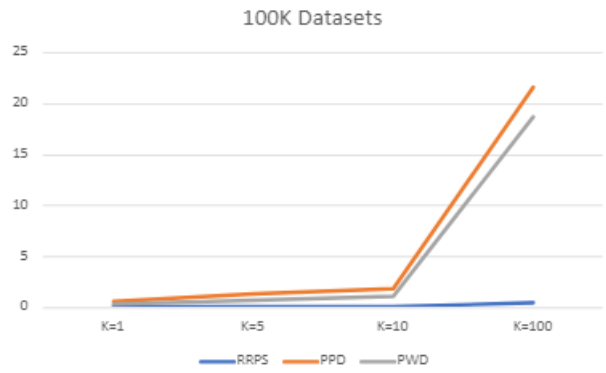
Η σειρά των πειραμάτων πραγματοποιήθηκε χωρίς να είναι γνωστός εκ των υστέρων ο βέλτιστος αριθμός χωρισμάτων και ούτε υπάρχει κάποιος μηχανισμός ή συνάρτηση που να υπολογίζει τον αριθμό αυτό. Ελέγχθηκε πειραματικά. Ίσως αυτός είναι ο λόγος που για 100Κ υπάρχει μια γρηγορότερη εκτέλεση, επειδή βρέθηκε ο βέλτιστος αριθμός χωρισμάτων που για τα άλλα σύνολα αυτό δεν συνέβει.



Σχήμα 5.7: Ο χρόνος σε δευτερόλεπτα στον κατακόρυφο άξονα και στον οριζόντιο το πλήθος πλησιέστερων ζευγών που αναζητείται, για 50Κ σημεία.

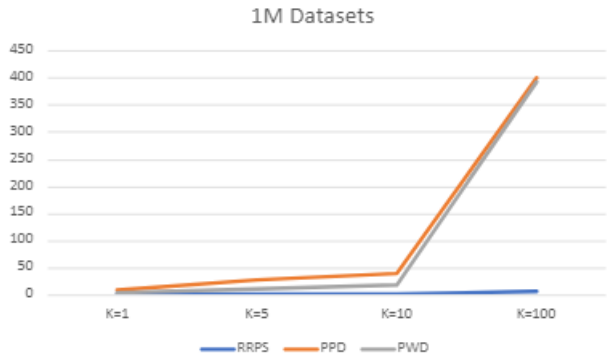
5.5.3 Σχέση χρόνου εκτέλεσης με μέγεθος συνόλων και πλήθος αναζητούμενων πλησιέστερων ζευγών

Από την παραπάνω πειραματική αξιολόγηση με σύνολα συνθετικών δεδομένων, συμπεραίνει κανείς ότι οι χρόνοι εκτέλεσης των αλγορίθμων είναι μεγαλύτεροι, καθώς το "Κ"



Σχήμα 5.8: Ο χρόνος σε δευτερόλεπτα στον κατακόρυφο άξονα και στον οριζόντιο το πλήθος πλησιέστερων ζευγών που αναζητείται, για 100K σημεία.

Σχήμα 5.9: Ο χρόνος σε δευτερόλεπτα στον κατακόρυφο άξονα και στον οριζόντιο το πλήθος πλησιέστερων ζευγών που αναζητείται, για 500K σημεία.



Σχήμα 5.10: Ο χρόνος σε δευτερόλεπτα στον κατακόρυφο άξονα και στον οριζόντιο το πλήθος πλησιέστερων ζευγών που αναζητείται, για 1M σημεία.

και το μέγεθος των συνόλων P, Q αυξάνεται. Συγκριτικά με τον "RRPS" αλγόριθμο οι δύο εκδοχές του παράλληλου αλγορίθμου έχουν πιο αργή εκτέλεση. Για μικρό πλήθος αναζητούμενων πλησιέστερων ζευγών και μικρά μεγέθη συνόλων σημείων η διαφορά δεν είναι πολύ

μεγάλη. Καθώς όμως οι παραπάνω παράμετροι αυξάνονται, ειδικά καθώς το "K" μεγαλώνει, η διαφορά είναι εμφανέστερη.

5.6 Αξιολόγηση με πραγματικά σύνολα δεδομένων

Τα πραγματικά δεδομένα που χρησιμοποιούνται για την πειραματική αξιολόγηση αντλήθηκαν από τη βάση δεδομένων του "SpatialHadoop" και συγκεκριμένα από το σύνολο σημείων "All points on the planet" και μπορούν να βρεθούν στην ιστοσελίδα `http://spatialhadoop.cs.umn.edu/datasets.html`. Στα πειράματα της παρούσας διπλωματικής εργασίας τα σύνολα δεδομένων αποτελούνται από 600.238 (600K), 613.404 (600K), 1.173.515 (1M) και 1.172.564 (1M) πραγματικά σημεία του πλανήτη. Χρησιμοποιήθηκαν μόνο συνδυασμοί για τα σύνολα P και Q με περίπου ίσο αριθμό σημείων δηλαδή 600K και 600K, 1M και 1M αντίστοιχα. Οι συντεταγμένες των σημείων δεν είναι κανονικοποιημένες σε κάποιο διάστημα και οι τιμές είναι πραγματικές (περιέχονται ακόμα και αρνητικές τιμές).

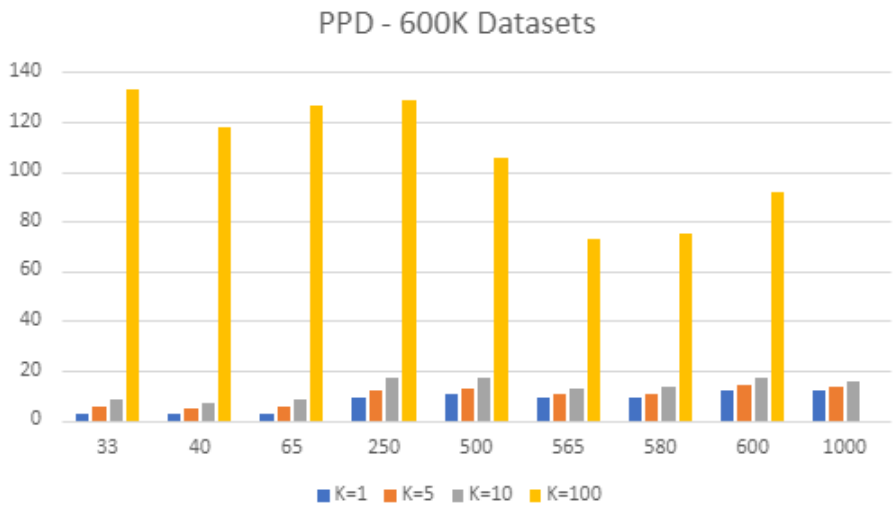
5.6.1 Βέλτιστος αριθμός χωρισμάτων για τον Παράλληλο Αλγόριθμο "K πλησιέστερων ζευγών"

Στα Σχήματα 5.11, 5.12, 5.13 και 5.14 φαίνεται πως επηρεάζει ο αριθμός των χωρισμάτων τον συνολικό χρόνο εκτέλεσης για διάφορες τιμές του "K". Για την εκδοχή με μερική μεταφορά των συνόλων σημείων και για μικρές τιμές του "K", το πλήθος χωρισμάτων που μεγιστοποιεί την απόδοση είναι μικρό ενώ για 100 πλησιέστερα ζεύγη τα χωρίσματα θα πρέπει να είναι 565 και 535 για 600K και 1M σύνολα σημείων αντίστοιχα. Η ίδια ακριβώς συμπεριφορά παρατηρείται και για τον αλγόριθμο "PWD".

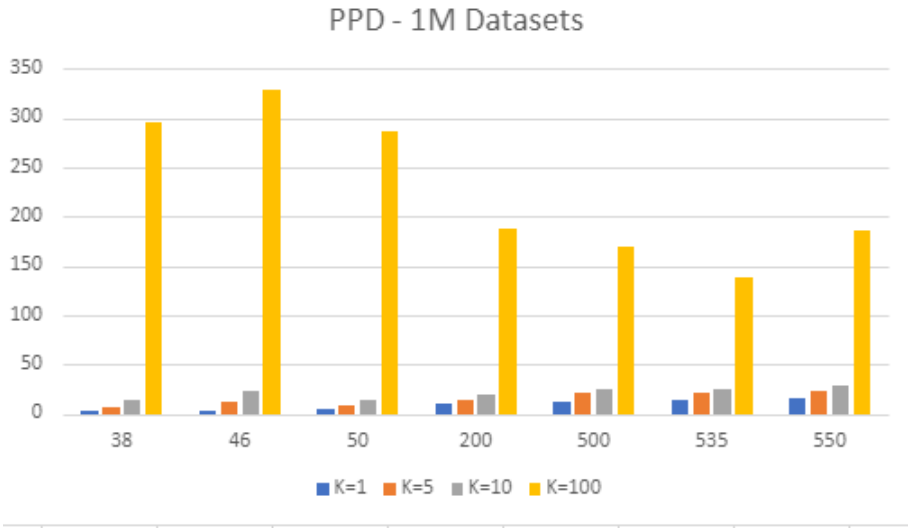
5.6.2 Σύγκριση Σειριακών με τον Παράλληλο Αλγόριθμο

Παρακάτω γίνεται σύγκριση του σειριακού "Reverse Run Plane Sweep Algorithm" με τις δύο εκδοχές του Παράλληλου Αλγορίθμου. Στα γραφήματα δεν συμπεριλήφθηκε ο σειριακός "Brute Force" λόγω της πολύ αργής εκτέλεσης του, παρόλα αυτά οι χρόνοι παρουσιάζονται αναλυτικά στον πίνακα 5.1.

Όπως φαίνεται και στα Σχήματα 5.15, 5.16 ο "RRPS" σειριακός αλγόριθμος έχει πιο σταθερή και γρηγορότερη εκτέλεση. Οι λόγοι που συμβαίνει αυτό είναι πολλοί. Πρώτον λόγω

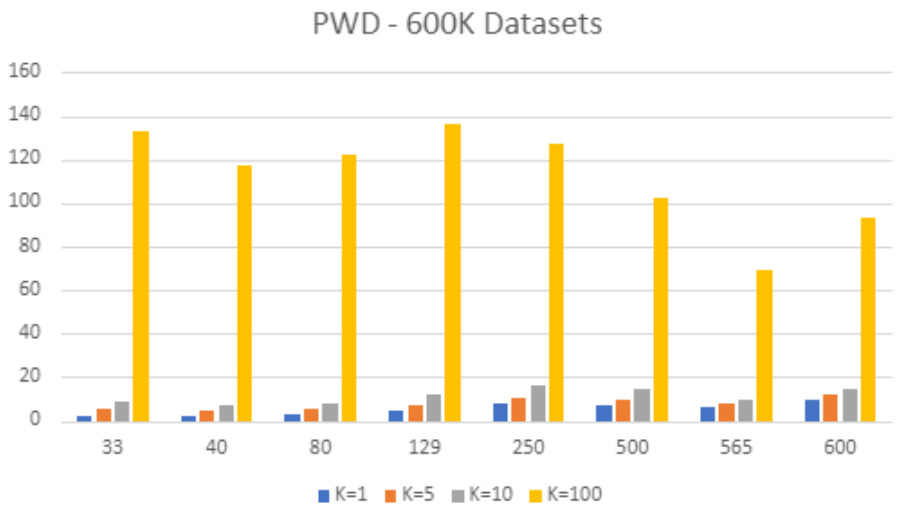


Σχήμα 5.11: Το πλήθος των χωρισμάτων στον οριζόντιο άξονα και ο χρόνος εκτέλεσης σε δευτερόλεπτα στον κατακόρυφο άξονα για σύνολα 600K σημείων με μερική μεταφορά στην μνήμη της κάρτας γραφικών, για διάφορα “K”.

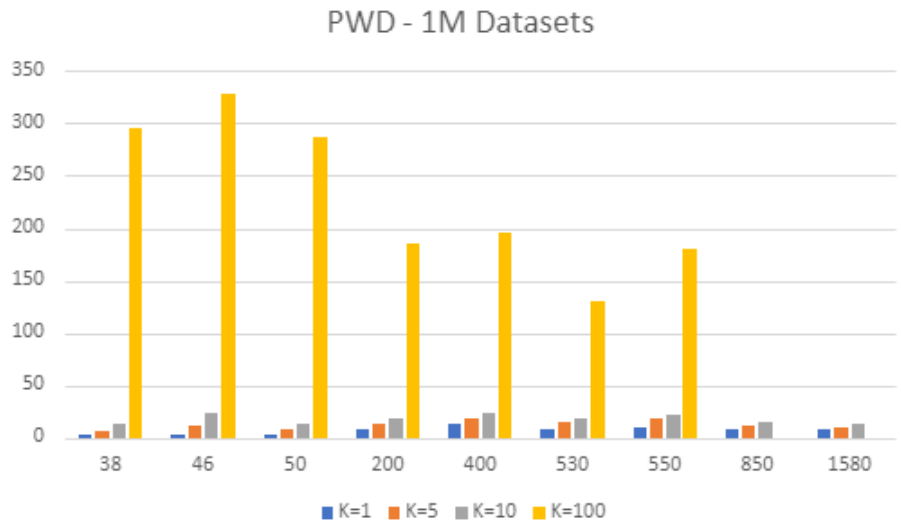


Σχήμα 5.12: Το πλήθος των χωρισμάτων στον οριζόντιο άξονα και ο χρόνος εκτέλεσης σε δευτερόλεπτα στον κατακόρυφο άξονα για σύνολα 1M σημείων με μερική μεταφορά στην μνήμη της κάρτας γραφικών, για διάφορα “K”.

της απουσίας μεταφορών στην μνήμη της κάρτας γραφικών στον σειριακό αλγόριθμο. Όσο το μέγεθος των συνόλων και το πλήθος αναζητούμενων ζευγών “K” αυξάνεται, οι μεταφορές είναι μεγαλύτερες και ο χρόνος μεταφοράς αυξάνεται, με συνέπεια να αυξάνει και ο συνολικός χρόνος εκτέλεσης. Δεύτερον, στον Παράλληλο Αλγόριθμο, είτε στην εκδοχή “PWD” είτε στην εκδοχή “PPD” απαιτείται κάποιου είδους προεπεξεργασία. Τέλος, ο Παράλληλος

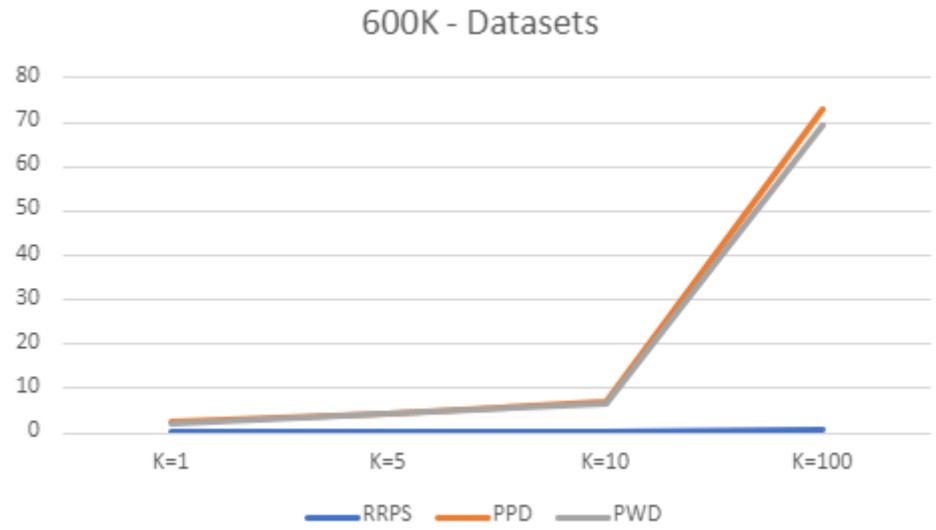


Σχήμα 5.13: Το πλήθος των χωρισμάτων στον οριζόντιο άξονα και ο χρόνος εκτέλεσης σε δευτερόλεπτα στον κατακόρυφο άξονα για σύνολα 600K σημείων με ολόκληρη μεταφορά στην μνήμη της κάρτας γραφικών, για διάφορα “K”.

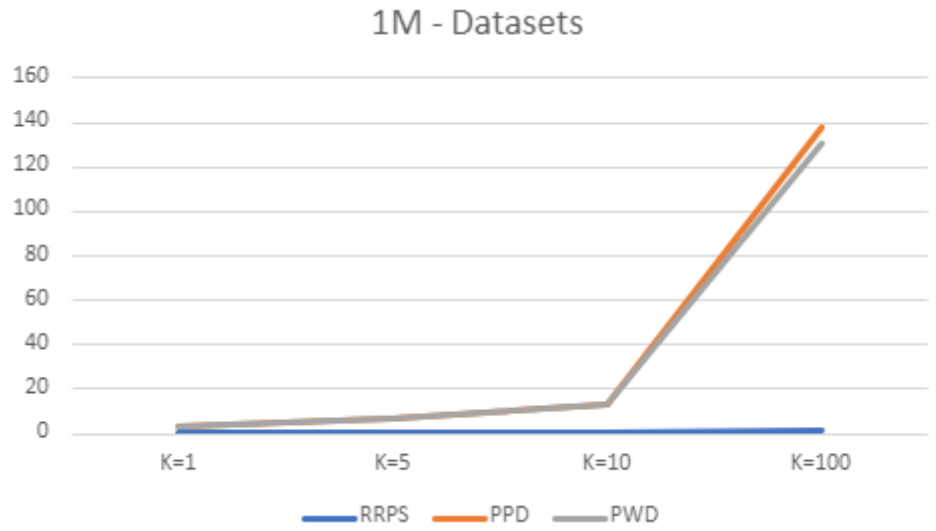


Σχήμα 5.14: Το πλήθος των χωρισμάτων στον οριζόντιο άξονα και ο χρόνος εκτέλεσης σε δευτερόλεπτα στον κατακόρυφο άξονα για σύνολα 1M σημείων με ολόκληρη μεταφορά στην μνήμη της κάρτας γραφικών, για διάφορα “K”.

Αλγόριθμος με μεταφορά ολόκληρων των συνόλων σημείων στην μνήμη της κάρτας γραφικών, φαίνεται ότι για μεγαλύτερο “K” έχει ελαφρώς καλύτερη απόδοση από την εκδοχή με μερική μεταφορά των συνόλων σημείων. Η διαδικασία επιλογής και προετοιμασίας των υποσυνόλων που απαιτούνται να περαστούν στην μνήμη της GPU για τον υπολογισμό των “K” πλησιέστερων ζευγών στην εκδοχή “PPD”, τελικά επιβαρύνει τον χρόνο εκτέλεσης.



Σχήμα 5.15: Στον κατακόρυφο άξονα οι χρόνοι εκτέλεσης σε δευτερόλεπτα και στον οριζόντιο το πλήθος αναζητούμενων ζευγών, "K".



Σχήμα 5.16: Στον κατακόρυφο άξονα οι χρόνοι εκτέλεσης σε δευτερόλεπτα και στον οριζόντιο το πλήθος αναζητούμενων ζευγών, "K".

Σειριακός Αλγόριθμος	Παράλληλος Αλγόριθμος	Μέγεθος Συνόλων/ Είδος συνόλων	Χρόνος Εκτέλεσης Σειριακού (s)	Χρόνος Εκτέλεσης Παράλληλου (s)	Πηλίκο Χρόνου Εκτέλεσης Παράλληλου/ Σειριακού
RRPS	PWD	50K,50K/συνθ.	0,04	0,69	17,25
RRPS	PPD	50K,50K/συνθ.	0,04	1,132	28,3
BF	PWD	50K,50K/συνθ.	143,45	0,69	0,004
BF	PPD	50K,50K/συνθ.	143,45	1,132	0,007
RRPS	PWD	100K,100K/συνθ.	0,08	1,093	13,66
RRPS	PPD	100K,100K/συνθ.	0,08	1,831	22,88
BF	PWD	100K,100K/συνθ.	567,429	1,093	0,001
BF	PPD	100K,100K/συνθ.	567,429	1,831	0,003
RRPS	PWD	500K,500K/συνθ.	0,48	10,984	22,88
RRPS	PPD	500K,500K/συνθ.	0,48	20,359	42,41
RRPS	PWD	1M,1M/συνθ.	0,95	18,185	19,14
RRPS	PPD	1M,1M/συνθ.	0,95	39,205	41,26
RRPS	PWD	600K,600K/πραγ.	0,363	6,699	18,45
RRPS	PPD	600K,600K/πραγ.	0,363	6,949	19,14
RRPS	PWD	1M,1M/πραγ.	0,719	12,824	17,83
RRPS	PPD	1M,1M/πραγ.	0,719	13,471	19,11

Πίνακας 5.1: Σύγκριση χρόνων εκτέλεσης σειριακών και παράλληλων αλγορίθμων για 10 πλησιέστερα ζεύγη

Κεφάλαιο 6

Συμπεράσματα και μελλοντικές προοπτικές

6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία έγινε ανάπτυξη και υλοποίηση ενός παράλληλου αλγορίθμου που να εκτελείται σε μία μονάδα επεξεργασίας γραφικών. Αρχικά, μελετήθηκε το μοντέλο CUDA και οι δυνατότητες που μπορεί να προσφέρει για μία παράλληλη υλοποίηση. Στην συνέχεια ερευνήθηκαν οι διάφορες εργασίες σε κεντρικά, παράλληλα και κατανεμήμενα συστήματα που να υλοποιούν αλγορίθμους που επεξεργάζονται χωρικά ερωτήματα αποδοτικά και γρήγορα. Διερευνήθηκαν σε βάθος οι διάφορες δομές δεδομένων, οι μηχανισμοί και οι τεχνικές επεξεργασίας που μπορούν να οδηγήσουν σε μια ικανοποιητική υλοποίηση σε μία παράλληλη αρχιτεκτονική GPU. Με βάση όλα τα παραπάνω, σχεδιάστηκε ένας αλγόριθμος που να τηρεί όλα τα κριτήρια για μια παράλληλη εκτέλεση. Επιλέχτηκε ο αλγόριθμος "Σάρωσης Επιπέδου" που φαίνεται να έχει την καλύτερη αποτελεσματικότητα και ταχύτητα. Η παρούσα υλοποίηση βασίστηκε κυρίως στις τεχνικές που παρουσιάζονται στο [3].

Ο στόχος ήταν να παρουσιαστεί ένας αλγόριθμος που να εκτελείται σε GPU για το ερώτημα "Κ πλησιέστερων ζευγών", αφού στη βιβλιογραφία δεν υπάρχει παρόμοια υλοποίηση αυτού του ερωτήματος σε κάρτα γραφικών. Το αποτέλεσμα της πειραματικής αξιολόγησης δείχνει ότι για μικρά σύνολα δέδομενων και μικρό πλήθος αναζητούμενων πλησιέστερων ζευγών η διαφορά στον χρόνο εκτέλεσης με τον "RRPS" είναι σχετικά μικρή. Το πρόβλημα όμως έγκειται στο γεγονός ότι όταν τα σύνολα δεδομένων και ο αριθμός "Κ" μεγαλώνει, οι

μεταφορές είναι χρονοβόρες επιβαρύνοντας σημαντικά την συνολική απόδοση. Όσο αφορά τη αξιολόγηση του Παράλληλου Αλγορίθμου με πραγματικά δεδομένα, η εκδοχή με μεταφορά ολόκληρων των συνόλων σημείων φαίνεται να αποδίδει ελαφρώς καλύτερα για μικρά "K" ενώ όσο το "K" αυξάνεται γίνεται όλο και πιο κατάλληλη. Για συνθετικά δεδομένα δεν συμβαίνει το ίδιο αφού για 100 πλησιέστερα ζεύγη και ανεξαρτήτως μεγέθους των συνόλων οι δύο εκδοχές "PWD" και "PPD" έχουν πολύ παρόμοιους χρόνους. Όταν όμως πρόκεται για μικρά "K" και πραγματικά σύνολα δεδομένων τα οποία να ξεπερνούν τα 100K, η εκδοχή με μεταφορά ολόκληρων των συνόλων σημείων είναι αρκετά αποδοτικότερη.

6.2 Μελλοντική έρευνα

Στην μελλοντική έρευνα συμπεριλαμβάνεται η πειραματική αξιολόγηση του Παράλληλου Αλγορίθμου για μεγαλύτερα σύνολα δεδομένων αλλά και για μεγαλύτερο αριθμό "K" πλησιέστερων ζευγών. Επίσης, μία άλλη προοπτική είναι η υλοποίηση παράλληλης ταξινόμησης του πίνακα με τα "K πλησιέστερα ζεύγη" από κάθε νήμα, αξιολόγωντας την εκτέλεση με διάφορες σειρές πειραμάτων για μεγάλα σύνολα δεδομένων.

Μια άλλη προσέγγιση αποτελεί η αξιολόγηση του αλγορίθμου για να προσδιοριστεί κατά πόσο επηρεάζει ο αριθμός των νημάτων των μπλοκ. Στην εν λόγω διπλωματική εργασία, ο αριθμός των νημάτων στα μπλοκ είναι 256, παρόλα αυτά μπορούν να εξεταστούν και 64, 128, 512 και 1024 νήματα.

Η εύρεση του κατάλληλου πλήθους χωρισμάτων φαίνεται να είναι πολύ σημαντική. Παρατηρήθηκε ότι η απόδοση αλλάζει σημαντικά προς το καλύτερο όταν ο αριθμός των χωρισμάτων είναι ο βέλτιστος. Επομένως, η ανακάλυψη κάποιας τεχνικής που να προσδιορίζει με ακρίβεια αυτόν τον αριθμό, πριν την παράλληλη εκτέλεση, στην φάση της προεπεξεργασίας ίσως βελτιώσει τον αλγόριθμο. Τέλος, ο παράλληλος αλγόριθμος εξαρτάται αρκετά από την εύρεση ενός άνω ορίου και η αναζήτηση τεχνικών που να βρίσκουν ένα ακόμα μικρότερο άνω όριο, επίσης θα βελτίωνε την απόδοση.

Βιβλιογραφία

[1] Cuda toolkit. https://docs.nvidia.com/cuda/cuda-c-programming-guide/. Ημερομηνία πρόσβασης: 21-01-2021.

[2] Γ. Παπαγεωργάκης. Ανάπτυξη αλγόριθμου closest pair με cuda api. Διπλωματική εργασία, Εθνικό και Καποδιστριακό Πανεπιστήμιο, Σεπ. 2014.

[3] G. Mavrommatis, P. Moutafis, M. Vassilakopoulos, F. Garcia-Garcia, and A. Corral. Slicenbound: Solving closest pairs and distance join queries in apache spark. In *ADBIS: European Conference on Advances in Databases and Information Systems*, Nicosia, Cyprus, Sep. 2017.

[4] K. Hinrichs, J. Nievergelt, and P. Schorn. Plane-sweep solves the closest pair problem elegantly. *Information Processing Letters*, 26(5):255–261, Jan. 1988.

[5] G. Roumelis, M. Vassilakopoulos, A. Corral, and Y. Manolopoulos. A new plane-sweep algorithm for the k-closest-pairs query. In *SOFSEM 2014: Theory and Practice of Computer Science*, Nový Smokovec, Slovakia, Jan. 2014.

[6] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Algorithms for processing k-closest-pair queries in spatial databases. *Data & Knowledge Engineering*, 49(1):67–104, Apr. 2004.

[7] Spatial database. http://wiki.gis.com/wiki/index.php/Geodatabase. Ημερομηνία πρόσβασης: 06-02-2021.

[8] Spatial query. https://www.igi-global.com/dictionary/query-processing-spatial-databases/27906. Ημερομηνία πρόσβασης: 06-02-2021.

[9] Differences between gpu and cpu. https://www.intel.com/content/www/us/en/products/docs/processors/cpu-vs-gpu.html. Ημερομηνία πρόσβασης: 06-02-2021.

[10] N. Thakur. Skyline queries. In Shashi Shekhar and Hui Xiong, editors, *Encyclopedia of GIS*, pages 62–76. Springer, 2008. https://doi.org/10.1007/978-0-387-35973-1.

[11] P. Moutafis, G. Mavrommatis, M. Vassilakopoulos, and S. Sioutas. Efficient processing of all-k-nearest-neighbor queries in the mapreduce programming framework. *Data & Knowledge Engineering*, 121:42–70, May 2019.

[12] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *20th International Conference on Data Engineering*, Boston, MA, USA, Apr. 2004.

[13] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Closest pair queries in spatial databases. In *SIGMOD Conference*, 2000.

[14] A. Corral and M. Vassilakopoulos. Query processing in spatial databases. *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends*, pages 269–278, Feb. 2009.

[15] R. Güting. An introduction to spatial database systems. *VLDB Journal*, 3(4):357–399, 1994.

[16] P. Velentzas, M. Vassilakopoulos, and A. Corral. In-memory k nearest neighbor gpu-based query processing. In *6th International Conference on Geographical Information Systems Theory, Applications and Management*, Online Streaming, May 2020.

[17] P. Velentzas, M. Vassilakopoulos, and A. Corral. A partitioning gpu-based algorithm for processing the k nearest-neighbor query. In *The 12th International Conference on Management of Digital EcoSystems (MEDES'20)*, Abu Dhabi, UAE, Nov. 2020.

[18] G. Roumelis, M. Vassilakopoulos, A. Corral, and Y. Manolopoulos. Plane-sweep algorithms for the k group nearest-neighbor query. In *International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM)*, Barcelona, Spain, Apr. 2015.

[19] F. Garcia-Garcia, A. Corral, L. Iribarne, M. Vassilakopoulos, and Y. Manolopoulos. Enhancing spatialhadoop with closest pair queries. In *20th East European Conference*, Prague, Czech Republic, Aug. 2016.

[20] G. Mavrommatis, P. Moutafis, and M. Vassilakopoulos. Binary space partitioning for parallel and distributed closest-pairs query processing. *International Journal on Advances in Software*, 10(3 & 4):275 – 285, 2017.