ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παραλληλισμός Αλγορίθμων βασισμένων στην Πυκνότητα Ομαδοποίησης Μεγάλων Δεδομένων σε Spark

---

Συγγραφέας:
Ευάγγελος Γκιάστας

Επιβλέπων:
Επίκουρος Καθηγητής Δημήτριος Κατσαρός

Παραδίδεται για την ολοκλήρωση των απαιτήσεων
και την απόκτηση διπλώματος
στο

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Οκτώβριος, 2018

UNIVERSITY OF THESSALY

BACHELOR THESIS

Parrallel/Distributed Implementation of Density-based Clustering Algorithms on Spark

---

Author:
Evangelos Gkiastas

Supervisor:
Assistant Professor Dimitrios Katsaros

A thesis submitted in fulfillment of the requirements
for the Bachelor degree
in the

Department of Electrical and Computer Engineering

October, 2018

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Περίληψη

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προπτυχιακό Δίπλωμα

Παραλληλισμός Αλγορίθμων βασισμένων στην Πυκνότητα Ομαδοποίησης Μεγάλων Δεδομένων σε Spark

Βαγγέλης Γκιάστας

Είναι πολύ δύσκολο να βρεθεί ένα θέμα που να έχει λάβει τόση δημοσιότητα με τόσο γρήγορο ρυθμό, όση τα μεγάλα δεδομένα. Η επικρατούσα αντίληψη είναι πως διανύουμε αναμφισβήτητα την εποχή τους. Για την εξόρυξη και την ανάλυση πληροφορίας από αυτά τα αχανή δεδομένα χρησιμοποιούνται ευρέως οι αλγόριθμοι ομαδοποίησης, οι οποίοι κατηγοριοποιούν στοιχεία σε ομάδες ανάλογα με την ομοιότητά τους. Σχετικά με τους αλγορίθμους αυτούς, έχει διαπιστωθεί πως όσοι βασίζονται στην τοπική πυκνότητα των στοιχείων δημιουργούν πιο ποιοτικές ομάδες και εμφανίζουν ανοχή στο θόρυβο. Ωστόσο, η επεξεργασία τεράστιου όγκου δεδομένων περιλαμβάνει πολλούς εντατικούς κι επαναληπτικούς υπολογισμούς, το κόστος των οποίων μειώνεται αν υλοποιηθούν παράλληλα. Στην παρούσα διπλωματική εργασία αναπτύσσονται τρεις αλγόριθμοι ομαδοποίησης, δύο εκ των οποίων βασίζονται στην πυκνότητα, με κατανεμημένο τρόπο στο Spark. Δίνονται αναλυτικές λεπτομέρειες του σχεδιασμού και της υλοποίησης των προσαρμοσμένων αυτών μεθόδων και εξετάζονται ως προς την απόδοσή τους για σύνολα τυχαίων σημείων, αλλά και για σύνολα τεράστιου όγκου δεδομένων του πραγματικού κόσμου. Επιπλέον, γίνεται αξιολόγηση των σχεδιαστικών αρχών του Spark και εισάγονται προτάσεις βελτίωσης των χρόνων εκτέλεσης.

UNIVERSITY OF THESSALY

Abstract

Department of Electrical and Computer Engineering

Barchelor Degree

Parrallel/Distributed Implementation of Density-based Clustering Algorithms on Spark

by Evangelos Gkiastas

It is very difficult to recall a topic that has received so much publicity at such a fast pace as big data. The prevailing perception is that we are undoubtedly going through their time. In mining and analyzing information from these vast data, clustering algorithms play a vital role, by categorizing data into groups according to their similarity. Concerning these algorithms, it is proven that those which are based on the local density of data form more qualitative groups and are tolerant to noise. However, the process of huge volume of data involves many iterative and data-intensive calculations, the cost of which can be reduced if they are implemented in parallel. In this thesis, three clustering algorithms are developed in a distributed way on Spark, two of which are density-based. Analytical details are provided for the design and implementation of these customized methods, and they are examined for their performance for random-points datasets, but also for vast amounts of real-world big data. In addition, Spark design principles are evaluated and proposals for improving execution times are introduced.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Δημήτριο Κατσαρό για την καθοδήγηση, τα κίνητρα και την απαραίτητη γνώση που μου παρείχε σε όλη τη διάρκεια της διπλωματικής εργασίας και των μαθημάτων. Οφείλω να ευχαριστήσω επίσης την οικογένεια μου και τους φίλους μου για τη συνεχή υποστήριξη και συμπαράστασή τους όλα αυτά τα χρόνια.

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

Κεφάλαιο 1

Εισαγωγή

Η ομαδοποίηση είναι ένα πολύ καλά ερευνημένο θέμα και παίζει σημαντικό ρόλο στον τομέα της εξόρυξης δεδομένων και στην ανακάλυψη πληροφορίας. Οι αλγόριθμοι ομαδοποίησης προσπαθούν να οργανώσουν τα δεδομένα χωρίζοντάς τα σε διαφορετικές συστάδες ανάλογα με την ομοιότητά τους. Η ομαδοποίηση έχει εφαρμοστεί με επιτυχία σε διάφορους τομείς όπως η βιοπληροφορική [1,2], η κυβερνοασφάλεια [3,4], η επεξεργασία εικόνας [5,6], η αστρονομία[7], τα κοινωνικά δίκτυα [8,9] κλπ.

Οι αλγόριθμοι χωρίζονται σε διαμεριστικούς, ιεραρχικούς, με βάση την πυκνότητα και με βάση το δίκτυο. Οι πρώτοι εκτελούν πολύ γρήγορη ομαδοποίηση αλλά παρουσιάζουν αστάθεια στο αποτέλεσμα και δε φιλτράρουν το θόρυβο. Οι δεύτεροι κατασκευάζουν μια δενδρική δομή κι έχουν μεγάλη ακρίβεια στις ομάδες αλλά παρουσιάζουν μεγάλη πολυπλοκότητα στους υπολογισμούς, λόγω της επαναληπτικής συγχώνευσης και διαμέρισης. Οι αλγόριθμοι με βάση το δίκτυο, χωρίζουν το χώρο των δεδομένων σε υποδίκτυα και εφαρμόζουν την ομαδοποίηση για τα στοιχεία μέσα σε αυτά μειώνοντας σημαντικά τη διάρκεια ομαδοποίησης. Τέλος, οι μέθοδοι που βασίζονται στην τοπική πυκνότητα υπερτερούν έναντι των υπολοίπων λόγω του ότι δημιουργούν αυθαίρετα και μπλεγμένα σχήματα ομάδων ακόμα και με την παρουσία θορύβου, όμως παίρνουν πολύ χρόνο να εκτελεστούν. [10]

Το Spark[18] είναι ένα προγραμματιστικό εργαλείο για εντατικά δεδομένα σε σύμπλεγμα υπολογιστών. Παρέχει διαχείριση εργασιών, ενημέρωση σχεδιασμού τοπικά και ανοχή σε σφάλματα, όντας παρόμοιο με το προηγούμενο κατανεμημένο μοντέλο για συμπλέγματα υπολογιστών, το MapReduce[11]. Συγκεκριμένα, η καινοτομία του Spark με την ελαστική κατανεμημένη βάση δεδομένων(RDD) η οποία “κρύβεται” στη μνήμη, επιτρέπει τους επαναληπτικούς αλγορίθμους να τρέχουν σε συμπλέγματα υπολογιστών. Οι αλγόριθμοι μηχανικής μάθησης πάντα περιλαμβάνουν πολλούς εντατικούς και επαναληπτικούς υπολογισμούς οι οποίοι πρέπει να επεξεργαστούν ένα τεράστιο όγκο δεδομένων. Ως εκ τούτου, πολλοί μέθοδοι μηχανικής μάθησης αναπτύσσονται στο Spark για να επιταχύνουν την εκτέλεση των πράξεων.

1.1 Υπόβαθρο

Για να καταλάβει πιο εύκολα την εργασία ο αναγνώστης θα πρέπει να γνωρίζει τους βασικούς αλγορίθμους ομαδοποίησης δεδομένων όπως ο K-means και ο DBSCAN. Εφόσον οι αλγόριθμοι αυτοί και οι παραλλαγές τους θα υλοποιηθούν στο Spark με χρήση της Python και οι σχετικοί κώδικες θα εξηγηθούν στο κεφάλαιο 4, ο αναγνώστης θα πρέπει επίσης να έχει επίγνωση των βασικών αρχών και εννοιών του Spark και να καταλαβαίνει βασικά παραδείγματα κώδικα του Spark γραμμένα σε Python.

Στο Κεφάλαιο 2 θα δωθεί το βασικό υπόβαθρο των αλγορίθμων ομαδοποίησης, της έννοιας των big data και του Spark.

1.2 Πρόβλημα

Επί του παρόντος, υπάρχουν πολλές παραλλαγές των παραδοσιακών αλγορίθμων ομαδοποίησης και εφαρμογές τους. Κάθε ερευνητής που ενδιαφέρεται για αυτόν τον τομέα πρέπει να ξοδέψει πολύ χρόνο για να συλλέξει όλες τις δημοσιεύσεις, και ακόμα περισσότερο για να αναγνωρίσει τα θετικά και τα αρνητικά αυτών ως προς τη χρήση και τις επιδόσεις τους.

Επιπλέον, η εφαρμογή αυτών των αλγορίθμων σε σύνολα μεγάλων δεδομένων,τους αναγάγει και στον τομέα της μηχανικής μάθησης με πολλούς επαναληπτικούς και εντατικούς υπολογισμούς. Επομένως, η επεξεργασία μεγάλου όγκου δεδομένων τους καθιστά κατάλληλους να υλοποιηθούν στο Spark. Ωστόσο, μέχρι σήμερα, δεν υπάρχουν πολλές υλοποιήσεις των density-based αλγορίθμων που να εστιάζουν στη βοήθεια και τις τεχνικές που παρέχει το Spark για γρήγορη επεξεργασία δεδομένων μεγάλης μάζας.

Ως εκ τούτου το πρόβλημα στο οποίο επικεντρώνεται η παρούσα διπλωματική εργασία είναι:

Πώς υλοποιούμε density-based clustering αλγορίθμους στο Spark?

Το πρόβλημα μπορεί να χωριστεί στα ακόλουθα υπο-ερωτήματα:

1. Ποιοι είναι οι κατάλληλοι αλγόριθμοι ομαδοποίησης για υλοποίηση στο Spark?
2. Πως πρέπει να διαμορφωθούν ώστε να επεξεργαστούν μεγάλο όγκο δεδομένων?
3. Πως υλοποιούνται σωστά στο Spark?

4. Πόσο βελτιώνονται οι επιδόσεις τους?

1.3 Στόχος

Για την απάντηση του προβλήματος που τέθηκε, τρεις αλγόριθμοι έχουν υλοποιηθεί στο Spark. Οι λεπτομέρειες υλοποίησης που περιλαμβάνουν την επιλογή αλγορίθμων και το σχεδιασμό προγραμμάτων θα συζητηθούν εκτενώς όπως και τα πειραματικά αποτελέσματα. Η εργασία ήρθε εις πέρας στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας. Ο κύριος σκοπός της είναι ο παραλληλισμός density-based αλγορίθμων στο Spark, η ερμηνεία των αποτελεσμάτων τους και η αποτίμηση των επιδόσεων της διεπαφής αυτής για κατανεμημένο σχεδιασμό.

1.4 Περίγραμμα

Η εργασία είναι δομημένη ως εξής. Το Κεφάλαιο 2 παρουσιάζει εκτενώς το υπόβαθρο των αλγορίθμων ομαδοποίησης, του τομέα των μεγάλων δεδομένων και της βασικής αρχιτεκτονικής του Spark. Στο Κεφάλαιο 3 γίνεται μια βιβλιογραφική ανασκόπηση των density-based αλγορίθμων και της σχετικής έρευνας πάνω σε αυτούς μέχρι σήμερα. Στο Κεφάλαιο 4 περιγράφεται η διαδικασία κατανεμημένης υλοποίησης των τριών αλγορίθμων στο Spark. Το Κεφάλαιο 5 αναλύει την απόδοση των αλγορίθμων για σύνολο τυχαίων και μη σημείων, ενώ στο Κεφάλαιο 6 γίνεται η ερμηνεία των αποτελεσμάτων για dataset μεγάλου όγκου από τον πραγματικό κόσμο. Τέλος, στο Κεφάλαιο 7 γίνεται η συζήτηση για μελλοντική έρευνα και παρουσιάζεται το συμπέρασμα.

Κεφάλαιο 2

Υπόβαθρο

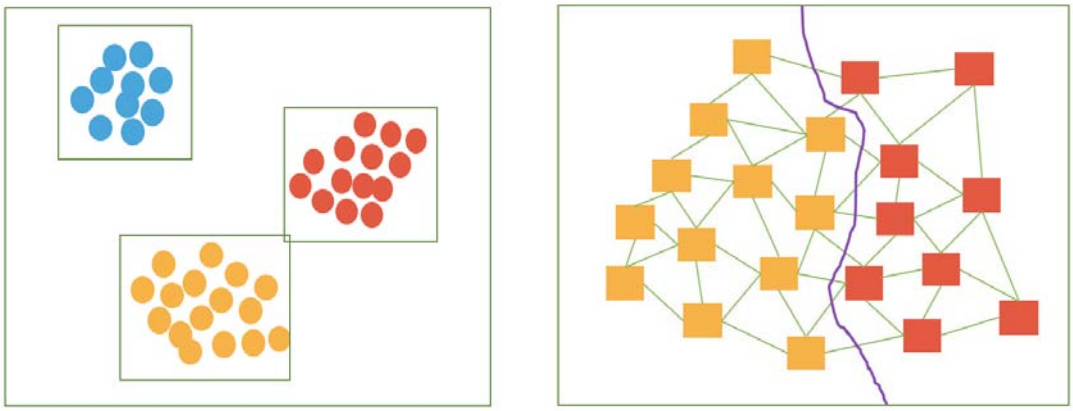
Στο κεφάλαιο αυτό παρουσιάζεται εν συντομία το απαραίτητο υπόβαθρο για την καλύτερη κατανόηση της εργασίας. Πρώτα, αναλύεται η γενική ιδέα του clustering και περιγράφονται κάποιοι παραδοσιακοί αλγόριθμοι. Στη συνέχεια, αναλύονται τα κύρια σημεία των Big Data. Τέλος, εισάγονται οι βασικές έννοιες του Spark, μικρά παραδείγματα κώδικα και αρχές σχεδιασμού.

2.1 Εισαγωγή στο clustering

Σε αυτή την ενότητα, πραγματοποιείται μια μικρή εισαγωγή στους αλγορίθμους για clustering. Αν και υπάρχουν πάρα πολλοί, ο αναγνώστης χρειάζεται να γνωρίζει τον K-means και τον DBCAN, ώστε να καταλάβει τις ακόλουθες ενότητες.

2.1.1 Τι είναι το clustering

Το clustering είναι ένα καλά μελετημένο αντικείμενο το οποίο περιλαμβάνει ομαδοποίηση δεδομένων και ομαδοποίηση γραφημάτων. Η ομαδοποίηση δεδομένων κατανέμει ένα σύνολο από σημεία δεδομένων σε ομάδες, και η ομαδοποίηση γραφημάτων διαιρεί ένα συνδεδεμένο γράφημα σε πολλά υπο-γραφήματα (Εικόνα 2.1) [13]. Τα σημεία ή αλλιώς κόμβοι που βρίσκονται στο ίδιο γκρουπ (υπο-γράφημα) εμφανίζουν περισσότερες ομοιότητες μεταξύ τους σε σχέση με τα σημεία των υπολοίπων γκρουπ. [15]



Εικόνα 2.1.1: Ομαδοποίηση δεδομένων και γραφημάτων.

2.1.2 K-means

Ο K-means είναι αλγόριθμος ομαδοποίησης για διάφορα είδη δεδομένων. Χρησιμοποιεί k κέντρα για να αναπαραστήσει k συστάδες (clusters). Ένα σημείο ανήκει στο cluster με τον κοντινότερο κέντρο [14]. Ως εκ τούτου, ο σκοπός του είναι να ελαχιστοποιήσει την ακόλουθη συνάρτηση κόστους

$$D(X,C)=\sum_{c=1}^{k}\sum_{i\in c}\|x_i-m_c\|^2 \quad ,$$

όπου το X είναι μια συλλογή δεδομένων, x_i είναι το εξεταζόμενο σημείο, C είναι ένα αποτέλεσμα clustering και m_c είναι το κέντρο του cluster.

Ο αλγόριθμος ξεκινάει αρχικοποιώντας k σημεία ως κέντρα με τυχαίο ή καθορισμένο τρόπο. Στη συνέχεια και με επαναληπτικό τρόπο ακολουθεί δύο βήματα μέχρι κάποια συγκεκριμένη επανάληψη ή μέχρι να φτάσει σε ένα ικανοποιητικό αποτέλεσμα.

- Ανάθεσε κάθε εξεταζόμενο σημείο στο κοντινότερο κέντρο.
- Ενημέρωσε κάθε κέντρο ώστε ο προσδοκώμενος αριθμός σημείων να ανήκει σε αυτό.

2.1.3 DBSCAN

Ο DBSCAN (**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise) είναι ένας αλγόριθμος συσταδοποίησης με βάση την πυκνότητα. Είναι κατάλληλος για ομάδες που έχουν υψηλή πυκνότητα σημείων, οι οποίες μπορεί να είναι διαχωρισμένες από άλλα σημεία(θόρυβος) χαμηλότερης πυκνότητας. Προϋποθέτει ότι η πυκνότητα των ομάδων είναι παρόμοια, χωρίς μεγάλες διακυμάνσεις. Δεν εμφανίζει μόνο ανοχή σε θόρυβο αλλά μπορεί επίσης να χειριστεί κάθε σχήμα του cluster χωρίς να ξέρει από πριν τον αριθμό των επιθυμητών clusters. Το MinPts είναι ο ελάχιστος αριθμός σημεί-

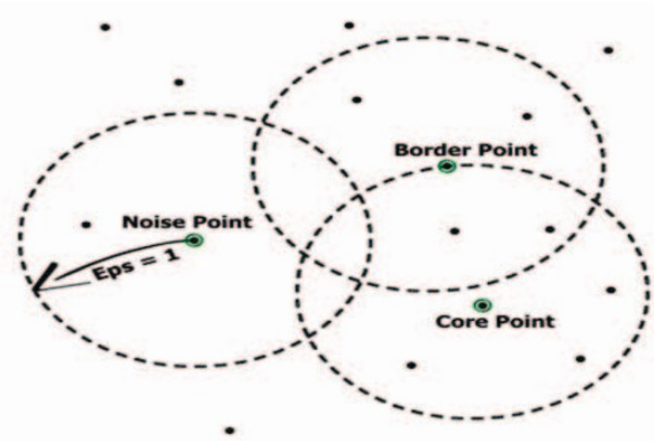
ων στο cluster και το Eps το κατώφλι της ακτίνας, και καθορίζονται από την αρχή. Υπάρχουν τρεις κατηγορίες σημείων (Εικόνα 2.1.3):

- Κεντρικό σημείο: έχει πυκνότητα μεγαλύτερη ή ίση από μία τιμή MinPts (ανήκουν στο εσωτερικό των ομάδων).
- Συνοριακό σημείο: έχει πυκνότητα μικρότερη από MinPts, αλλά απέχει από ένα κεντρικό σημείο απόσταση μικρότερη ή ίση από Eps (βρίσκονται στα όρια των ομάδων).
- Θορυβώδες σημείο: κάθε άλλο σημείο (ανήκουν στις περιοχές χαμηλής πυκνότητας)

Κάθε σημείο χαρακτηρίζεται σε μία από τις τρεις κατηγορίες. Αγνοούνται όλα τα σημεία θορύβου. Τοποθετείται μια ακμή μεταξύ όλων των κεντρικών σημείων που είναι σε απόσταση έως Eps μεταξύ τους. Θέτεται κάθε ομάδα συνδεδεμένων βασικών σημείων ως μια διαφορετική συστάδα. Αναθέτεται κάθε συνοριακό σημείο σε μία από τις συστάδες των συσχετιζόμενων βασικών σημείων με βάση την απόσταση. [12]

Τα βήματα τα αλγορίθμου περιγράφονται ως ακολούθως:

1. Αυθαίρετα επίλεξε ένα σημείο r.
2. Ανάκτησε όλα τα σημεία που είναι πυκνωτικά προσβάσιμα από το r με βάση το Eps και το MinPts.
3. Αν το r είναι κεντρικό σημείο,σχηματίζεται το cluster.
4. Αν το r είναι συνοριακό, δεν υπάρχουν πυκνωτικά προσβάσιμα σημεία από το r και ο DBSCAN επισκέπτεται το επόμενο σημείο των δεδομένων.
5. Συνέχισε τη διαδικασία μέχρις ότου όλα τα σημεία να έχουν εξεταστεί.



Εικόνα 2.1.2: DBSCAN με MinPts = 5 και Eps = 1.

2.1.4 Σύγκριση DBSCAN με K-means

Σε αυτό το σημείο παραθέτονται αναλυτικά οι διαφορές και οι ομοιότητες των δύο προαναφερθέντων αλγορίθμων. [16]

- Και οι δύο αλγόριθμοι τοποθετούν ένα σημείο σε μία μοναδική ομάδα. Αλλά, ο dbscan μπορεί να μην ομαδοποιήσει όλα τα σημεία.
- Ο K-means χρησιμοποιεί την έννοια του κέντρου της ομάδας ενώ ο dbscan της πυκνότητας.
    - Ο dbscan χειρίζεται σωστά ομάδες με διαφορετικό σχήμα και μέγεθος, σε αντίθεση με τον K-means. Και οι δύο αλγόριθμοι δεν μπορούν να χειριστούν ομάδες με διαφορετική πυκνότητα.
- Ο K-means απαιτεί να μπορεί να οριστεί το κέντρο της ομάδας, ενώ ο dbscan απαιτεί να έχει νόημα η έννοια της πυκνότητας.
- Ο K-means έχει καλή απόδοση σε αραιά, πολυδιάστατα δεδομένα (πχ., κείμενα). Ο dbscan δεν έχει καλή απόδοση σε αυτήν την περίπτωση.
- Και οι δύο μπορούν (με επεκτάσεις) να χειριστούν δεδομένα άλλων τύπων εκτός από αριθμητικά.
- Και οι δύο λαμβάνουν υπ' όψιν όλα τα χαρακτηριστικά, δηλ. δεν δημιουργούν ομάδες βάσει μόνο κάποιων χαρακτηριστικών.
- Ο dbscan συνενώνει ομάδες που εφάπτονται ή επικαλύπτονται, σε αντίθεση με τον K-means.
- Η πολυπλοκότητα του K-means είναι μικρότερη από αυτή του dbscan.
- Ο K-means παράγει διαφορετικές συστάδες για τα ίδια δεδομένα, σε αντίθεση με τον dbscan.
- Ο dbscan δεν απαιτεί τον εκ των προτέρων ορισμό του αριθμού των ομάδων, όπως κάνει ο K-means. Απαιτεί όμως, τον ορισμό παραμέτρων όπως MinPts/ Eps.

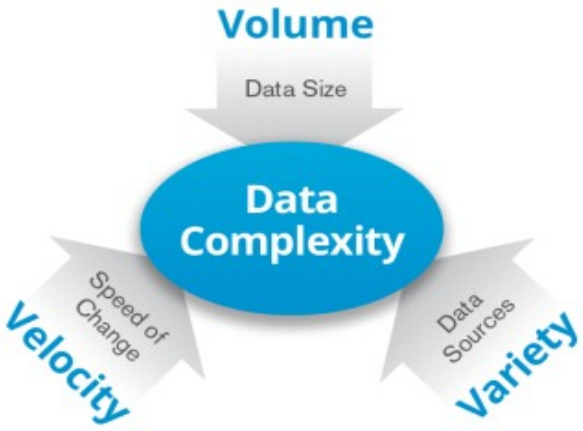
2.2 Big Data

Ενώ μέχρι πριν λίγα χρόνια ήταν μόλις και μετά βίας γνωστή αυτή η έννοια των big data, σήμερα είναι από τα πιο πολυσυζητημένα θέματα σε όλο το επιχειρηματικό κόσμο. Σε αυτό το κεφάλαιο εισάγονται οι βασικές έννοιες των big data καθώς και οι εφαρμογές και τα κύρια τεχνολογικά συστατικά στο οικοσύστημά τους.

2.2.1 Τι ονομάζουμε Big Data

Η ορολογία Big Data ορίζεται ως τεραστίων μεγεθών δεδομένα, δομημένα ημιδομημένα ή αδόμητα, με τη δυνατότητα να εξορύξουμε πληροφορία από αυτά, όπου λόγω όγκου δεν έχει πλέον σημασία μια μεμονωμένη καταγραφή αλλά το σύνολο του μεγέθους τους. Οι κύριες έννοιες οι οποίες αποτελούν τη βάση του ορισμού των Big Data αποτελούνται από το μοντέλο των τριών V (Εικόνα 2.2.1):

- Volume: Τεράστιες ποσότητες δεδομένων από σύνολα με μεγέθη terabytes και zettabytes.
- Velocity: Τεράστιες ποσότητες δεδομένων από συναλλαγές με υψηλό ρυθμό ανανέωσης δημιουργούν ροές δεδομένων (streams) με μεγάλη ταχύτητα και μειώνουν τον διαθέσιμο χρόνο επεξεργασίας σε αυτά. Υπάρχει μια αλλαγή από την επεξεργασία στατικών δεδομένων σε επεξεργασία streams πραγματικού χρόνου.
- Variety: Τα δεδομένα προέρχονται από διαφορετικές πηγές. Μπορούν να έρχονται σε ποικίλες μορφές όπως:
  - Δομημένα: πινακες SQL, Excel Sheets κλπ.
  - Ημιδομημένα: Xml δεδομένα όπως e-mails, tweets κλπ.
  - Αδόμητα: Κείμενο, φωτογραφίες, βίντεο κλπ.



Εικόνα 2.2.1: Τα τρία V των Big Data.

Στο [21] τα Big Data ορίζονται ως πληροφοριακά στοιχεία υψηλού όγκου, υψηλής ταχύτητας και μεγάλης ποικιλίας που απαιτούν καινοτόμα μοντέλα επεξεργασίας με βελτιωμένη γνώση, αυτοματισμό και λήψη αποφάσεων. Είναι κατανοητό πλέον πως ο όρος “big” δεν αναφέρεται μόνο στον όγκο, όπως και το ότι τα big data του σήμερα μπορεί να μην αποτελούν τα big data του αύριο καθώς οι τεχνολογία εξελίσσεται ταχύτατα.

Κάποια παραδείγματα για τα δεδομένα που ανακτούν όλοι οι οργανισμοί σήμερα είναι:

- τα δεδομένα του Web (πλοήγηση χρηστών,cookies),
- τα δεδομένα κειμένου (Facebook posts, ειδήσεις, email),
- τα δεδομένα ώρας και τοποθεσίας (μέσω GPS,WI-FI,smartphones),
- τα δεδομένα από αισθητήρες και έξυπνα δίκτυα (από αυτοκίνητα,αγωγούς πετρελαίου κλπ),
- τα δεδομένα των κοινωνικών δικτύων(όπως Facebook, Instagram κλπ).

Όλα αυτά χρησιμοποιούνται είτε για εξόρυξη γνώσης από τα προσωπικά δεδομένα και ενδιαφέροντα των χρηστών και εν συνεχεία την εμπορική εκμετάλλευση, είτε για βελτίωση των επιδόσεων όλων των αναφερθέντων τεχνολογιών.

2.2.2 Το τεχνολογικό περιβάλλον των Big Data

Παραδοσιακά, τα δεδομένα είναι αποθηκευμένα σε σχεσιακές βάσεις (π.χ. CRM System) και όποτε χρειαστεί εξάγονται, μετασχηματίζονται και φορτώνονται στις αποθήκες δεδομένων για ανάλυση. Αυτή η διαδικασία δυσκολεύει όταν αντιμετωπίζει μεγάλα δεδομένα. Για παράδειγμα, ένα από τα πολυσυζητημένα computing clusters του Hadoop(Yahoo) ήταν στα 455 petabytes το 2014 κι έχει μεγαλώσει από τότε. Δεν υπάρχει παράλληλη σχεσιακή βάση ή αποθήκη δεδομένων που να πλησιάζει αυτά τα νούμερα. Ένα ακόμα δυνατό σημείο του Hadoop έναντι της σχεσιακής τεχνολογίας είναι τα αδόμητα δεδομένα όπως ήχος, βίντεο και κείμενο.[22]

Πρέπει να σημειωθεί ότι υπάρχει γενικά μια λανθασμένη εντύπωση ότι η νέα τεχνολογία, όπως το Hadoop, αντικαθιστά τις παλιές. Δεν είναι έτσι όμως. Το πιο σωστό είναι να πούμε πως η μία αλληλοσυμπληρώνει την άλλη. Για παράδειγμα ένα πλεονέκτημα μια τεράστιας σχεσιακής βάσης, είναι η διαχείριση δομημένων δεδομένων που συναλλάσσονται με υψηλούς ρυθμούς, και χρειάζεται υποστήριξη για πολλούς χρήστες και εφαρμογές των οποίων οι αιτήσεις είναι πάνω σε γνωστά δεδομένα, διασφαλίζοντας με προκαθορισμένα σχήματα και βελτιστοποιήσεις εγγύηση απόδοσης και επιχειρηματική ασφάλεια. [23]

Όταν ερευνώνται τα διαφορετικά επίπεδα των τεχνολογιών που χρησιμοποιούνται στα big data, αναφέρεται ο όρος “ Οικοσύστημα του Hadoop”. Η πλήρης λίστα παρέχεται στη διεύθυνση [42], ενώ κάποια δημοφιλή παραδείγματα είναι τα κάτωθι:

- Apache HDFS(Hadoop Distributed File System) για κατανεμημένο σύστημα αρχείων,
- Amazon web service (χρησιμοποιείται στο Cloud),
- MapReduce και το Spark για κατανεμημένο μοντέλο προγραμματισμού,
- Cassandra ή HBase για σύστημα μη σχεσιακής παράλληλης βάσης δεδομένων,

- Mahout για βιβλιοθήκη μηχανικής μάθησης, πάνω στο MapReduce,
- Hive για SQL πάνω στο Hadoop,
- R για ανάλυση και οπτικοποίηση δεδομένων.

Τέλος, οι πιο ευρέως χρησιμοποιημένες τεχνικές ανάλυσης δεδομένων θεωρούνται οι στατιστικές μέθοδοι, η πρόβλεψη (forecasting), η παλινδρόμηση (regression), η ομαδοποίση (clustering), οι αιτήσεις στη database, η μηχανική μάθηση(machine learning) και η εξόρυξη(data mining). [22][23]

2.3 Τα βασικά του Spark

Στην παρούσα εργασία αναπτύσσονται τρεις αλγόριθμοι στο Spark, και γι αυτό το λόγο σ αυτή την ενότητα εισάγονται σε θεωρητικό επίπεδο η γενική ιδέα του Spark,οι ελαστικές κατανεμημένες βάσεις δεδομένων (RDDs), απλά παραδείγματα κώδικα, και κάποιες σχεδιαστικές αρχές.

2.3.1 RDDs

Μια εφαρμογή με δεδομένα υψηλής έντασης απαιτεί πολλούς υπολογισμούς οι οποίοι επεξεργάζονται τεράστιο αριθμό δεδομένων ανεξάρτητα στις ίδιες λειτουργίες. Εφόσον αυτοί οι υπολογισμοί δεν εξαρτώνται από τα δεδομένα,μπορούν να κατανεμηθούν επιταχύνοντας τη διαδικασία.

Αρχιτεκτονικές συμπλέγματος υπολογισμών όπως το MapReduce και το Dryad έχουν υλοποιήσει με επιτυχία εφαρμογές μέγαλου μεγέθους δεδομένων και υπολογισμών σε εμπορικά συμπλέγματα (commodity clusters). Παρέχουν με αυτοματοποιημένο τρόπο τοπολογικής επίγνωσης προγραμματισμό, ανοχή σε σφάλματα, και ισορροπία φόρτου εργασίας. Ωστόσο δεν μπορούν να εφαρμοστούν σε μια σημαντική κατηγορία εφαρμογών, αυτές που επαναχρησιμοποιούν δεδομένα (π.χ., επαναληπτικό machine learning). Αυτό συμβαίνει διότι είναι χτισμένα γύρω από ένα απεριοδικό μοντέλο ροής δεδομένων και αποθηκεύουν ενδιάμεσα δεδομένα σε εξωτερικό χώρο αποθήκευσης, ο οποίος προκαλεί αναποτελεσματική επαναχρησιμοποιήση δεδομένων εξαιτίας της εισόδου/εξόδου στο δίσκο (I/O), της σειριοποίησης και της αντιγραφής των δεδομένων. [17][18]

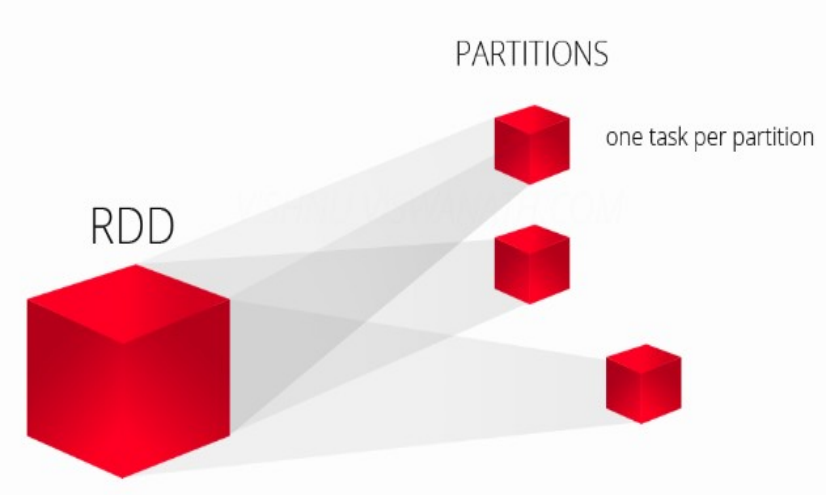
Το Spark λύνει το πρόβλημα αυτό εισάγοντας ένα στρώμα αφαίρεσης, που ονομάζονται ελαστικές κατανεμημένες βάσεις δεδομένων(RDDs) (Εικόνα 2.3.1). Η RDD είναι μια read-only συλλογή αντικειμένων διαμοιρασμένα σε ένα σύνολο μηχανημάτων τα οποία μπορούν να δομηθούν ξανά αν μια διανομή (partition) χαθεί. Οι RDDs χαρακτηρίζονται συχνά ότι λειτουργούν με “τεμπέλικο” τρόπο, γι’ αυτό οι παλιές RDDs μπορεί να διαγραφούν αν η μνήμη είναι περιορισμένη. Ωστόσο, αν ο χρήστης κρατάει

στη μνήμη κρυφή μνήμη (cache) των μηχανημάτων την RDD, μπορεί να χρησιμοποιηθεί ξανά και ξανά χωρίς να διαγραφεί. Επομένως, το Spark μπορεί να ασχοληθεί με τις εφαρμογές που επαναχρησιμοποιούν δεδομένα. [17][18]

Μια RDD μπορεί να δημιουργηθεί μόνο από λειτουργίες μετασχηματισμού (transformations) από σταθερά αποθηκευμένα δεδομένα ή από άλλες RDDs. Μπορούν να χρησιμοποιηθούν από ενέργειες(actions) επιστροφής τιμών ή εξαγωγής δεδομένων σε ένα αποθηκευτικό σύστημα. Ο λόγος που χαρακτηρίζονται "τεμπέλες" είναι επειδή εκτελούν μόνο actions παρότι έχουν δημιουργηθεί από transformations προηγουμένως. Το Spark, επιπλέον, παρέχει κλιμάκωση και ανοχή σε σφάλματα αυτοματοποιημένα. Η υπολογιστική διαδικασία ενός προγράμματος σχεδιάζεται σαν χρονοδιάγραμμα, οπότε σε τυχόν απώλειες οι κατεστραμένες RDDs μπορούν να επαναδομηθούν σύμφωνα με γραμμικό χρονοδιάγραμμα των πράξεων.

Αξίζει επίσης να σημειωθεί πως όσο μεγαλύτερος είναι ο αριθμός των partitions τόσο μεγαλύτερος παραλληλισμός θα υπάρχει. Το Spark αποφασίζει αυτόματα για τον αριθμό αυτό αλλά μπορεί να οριστεί και από το χρήστη όταν δημιουργεί την RDD στους κόμβους του δικτύου.

Τέλος,οι εξαρτήσεις μεταξύ των RDDs χωρίζονται σε εξαρτήσεις στενής και ευρείας σύνδεσης. Οι πρώτη ονομασία σημαίνει πως κάθε διανομή της γονικής RDD χρησιμοποιείται από μια το πολύ διανομή της RDD του παιδιού επιτρέποντας εκτέλεση με pipelines σε έναν κόμβο του cluster. Η ανάκτηση μετά την αποτυχία ενός κόμβου είναι πιο αποτελεσματική με στενή σύνδεση, επειδή μόνο οι χαμένες διανομές του γονέα πρέπει να υπολογιστούν ξανά. Οι συνδέσεις ευρείας κλίμακας από την άλλη επιτρέπουν σε πολλές διανομές παιδιών να εξαρτώνται από την RDD. Η ανάκτηση μετά την αποτυχία ενός κόμβου όμως πλήρη επανεκτέλεση όλων των διανομών. Συμπεραίνεται λοιπόν, πως αν μια RDD έχει ευρεία εξάρτηση με άλλες RDDs είναι καλύτερο να την κρατάμε στην προσωρινή κρυφή μνήμη (cache). [17][18]



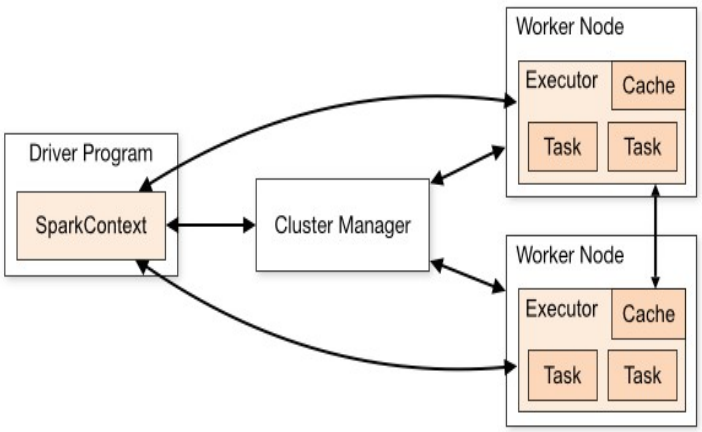
Εικόνα 2.3.1: Η RDD και τα μικρότερα εξαρτώμενα κομμάτια(partitions).

2.3.2 Το Spark

Είναι μια ολοκληρωμένη προγραμματιστική διεπαφή εφαρμογών (API) για RDDs σε μια αμετάβλητη έκδοση της Scala. Το σύστημα τρέχει πάνω από το Mesos cluster manager και μπορεί να διαβάζει δεδομένα από κάθε πηγή εισόδου (input) του Hadoop χρησιμοποιώντας τις υπάρχουσες διεπαφές εισόδου του δευτέρου. [17][18]

Το Spark αποτελείται από ένα πρόγραμμα οδηγό κι ένα cluster εργατών. Το πρόγραμμα οδηγός υλοποιεί στο υψηλό επίπεδο τον έλεγχο της εφαρμογής, ενώ οι εργάτες αποθηκεύουν τις RDD partitions δια μέσου των εργασιών (Εικόνα 2.3.2).

Εκτός από τις RDDs, άλλα δύο στρώματα του Spark για ταυτόχρονο προγραμματισμό είναι οι παράλληλες λειτουργίες και οι κοινές μεταβλητές. Οι πρώτες θα συζητηθούν στην Ενότητα 2.3.3 με παραδείγματα κώδικα. Υπάρχουν δύο τύποι κοινών μεταβλητών, οι broadcast μεταβλητές και οι συσσωρευτές (accumulators). Οι broadcast μεταβλητές είναι αμετάβλητες και κατανεμημένες στα νήματα των εργατών. Εάν ένας χρήστης θέλει να μοιραστεί μια τροποποιημένη μεταβλητή -και διαφορετική από τις τοπικές μεταβλητές- με όλους τους εργάτες,τότε η broadcast μεταβλητή αποτελεί μια καλή λύση. Οι συσσωρευτές από την άλλη,ναι μεν επιτρέπουν την ενημέρωση της τιμής μιας broadcast μεταβλητής, αλλά επιτρέπουν μόνο την πρόσθεση,όπως υποδηλώνει και το όνομα τους άλλωστε. [17][18]



Εικόνα 2.3.2: Το πρόγραμμα οδηγός και οι κόμβοι των εργατών σε ένα Spark cluster.

2.3.3 Παραδείγματα κώδικα σε Spark

Σε αυτή την ενότητα αναλύονται κάποιες από τις πιο βασικές λειτουργίες παραλληλισμού που χρησιμοποιούνται στις RDDs. [19][20]

Transformations

- **map (func):** Επιστρέφει ένα καινούργιο σύνολο δεδομένων διαμορφώνοντας κάθε στοιχείο του πηγαίου συνόλου με βάση μια συνάρτηση *func*. Για παράδειγμα,

  *rdd2 = rdd1.map (_ * 10) ,*

  όπου κάθε στοιχείο της rdd1 πολλαπλασιάζεται με 10 και αποθηκεύεται στην rdd2.

- **GroupByKey:** Τοποθετεί σε ομάδες(groups) τιμές με το ίδιο κλειδί και οι ομαδοποιημένες τιμές αποθηκεύονται στο επαναληπτικό [V] το οποίο επιστρέφεται μαζί με το κλειδί. Για παράδειγμα,

  keyGroup = rdd1.groupByKey () ,

  επιστρέφει μια λίστα με ζευγάρια του τύπου (key,group).

- **Join (otherDataset, [numTasks]) :** Όταν καλείται σε datasets του τύπου (K, V) και (K, W) επιστρέφει ένα dataset της μορφής (K, (V, W)) . Για παράδειγμα,

  join_result = rdd1.join (rdd2, rdd1.name == rdd2.name) ,

  Ενώνει τα στοιχεία των δύο συνόλων που έχουν ίδιο όνομα.

Actions

- **reduce (func):** Συνδυάζει στοιχεία του dataset και τα αθροίζει χρησιμοποιώντας μια συνάρτηση *func* (συνήθως παίρνει δύο ορίσματα και επιστρέφει ένα). Η συνάρτηση πρέπει να είναι προσεταιριστική ώστε να υπολογιστεί σωστά παράλληλα και να επιστρέψει το αποτέλεσμα στο πρόγραμμα οδηγό. Για παράδειγμα η εντολή

  sum = rdd1.reduce (_ + _) ,

  που υπολογίζει το άθροισμα των στοιχείων στην rdd1 είναι πιο εύκολο να εκφραστεί ως

  sum = rdd1.sum ().

- **foreach (func):** Περνάει κάθε στοιχείο μέσα από μία συνάρτηση *func* παρεχόμενη από το χρήστη. Χρησιμοποιείται συνήθως για ενημέρωση accumulators ή για αλληλεπίδραση με εξωτερικά συστήματα αποθήκευσης. Για παράδειγμα,

  rdd1. foreach (println (_)) ,

  κάθε στοιχείο της rdd1 εκτυπώνεται.

  Η *map* και η *foreach* είναι διαφορετικές,καθώς το αποτέλεσμα της πρώτης αποθηκεύεται σε καινούργια rdd, σε αντίθεση με τη δεύτερη η οποία χρησιμοποιεί κάθε στοιχείο της rdd χωρίς να αποθηκεύσει τίποτα.

- **Collect () :** Επιστρέφει όλα τα στοιχεία ενός dataset ως πίνακα στο πρόγραμμα οδηγό. Είναι συνήθως χρήσιμο μετά από το *filter* ή άλλη ενέργεια η οποία θα

έχει επιστρέψει ένα αρκετά μικρότερο υποσύνολο των δεδομένων. Για παράδειγμα,

keyGroup = rdd1.collect () ,

στέλνει την rdd1 στο πρόγραμμα οδηγό.

2.3.4 Βασικές αρχές σχεδιασμού ενός προγράμματος

Έχοντας αναλύσει τις βασικές έννοιες των RDDs και του Spark, συμπεραίνονται οι ακόλουθες προγραμματιστικές τεχνικές.

- Παραλλήλισε τους υπολογισμούς όσο το δυνατόν περισσότερο.
- Κράτα στην μνήμη cache τις RDDs ,αν πρόκειται να ξαναχρησιμοποιηθούν αρκετές φορές, ή αν υπάρχει μια καθολική παράμετρος από κάποια επανάληψη η οποία δεν μπορεί να υπολογιστεί ξανά.
- Αποδέσμευσε από την μνήμη cache όποιες RDDs δε χρησιμοποιηθούν μελλοντικά στο πρόγραμμα.
- Χρησιμοποίησε τις broadcast μεταβλητές και τους accumulators κατάλληλα και όταν πρέπει.

Κεφάλαιο 3

Βιβλιογραφική Ανασκόπηση

Στο κεφάλαιο αυτό εισάγεται ο βασικός αλγόριθμος ο οποίος αποτέλεσε το κύριο αντικείμενο έρευνας και ανάπτυξης της εργασίας. Επίσης παρουσιάζεται εν συντομία η μέχρι στιγμής έρευνα πάνω σε αυτόν και κάποιες παραλλαγές του.

3.1 Ορισμοί και σημειώσεις

Αρχικά παρέχονται κάποιοι ορισμοί και όροι κλειδιά στους οποίους γίνεται συχνή αναφορά στη συνέχεια. Έχουν υιοθετηθεί κυρίως από το “Clustering by fast search and find of density peaks” [24]. Αυτός ο αλγόριθμος ομαδοποίησης ανήκει στην κατηγορία των density-based, οπότε ο κύριος ορισμός είναι η τοπική πυκνότητα ενός αντικειμένου.

ΟΡΙΣΜΟΣ 1. *Τοπική πυκνότητα (Local Density)* ενός αντικειμένου - δηλωμένης ως ρ - είναι η τιμή μιας συνάρτησης αποστάσεων του εξεταζόμενου αντικειμένου από όλα τα άλλα αντικείμενα του συνόλου των δεδομένων.

Πιο απλά, μπορούμε να πούμε πως η τοπική πυκνότητα ενός αντικειμένου είναι ο αριθμός των αντικειμένων που είναι πιο κοντά σε αυτό από την *απόσταση αποκοπής*.

ΟΡΙΣΜΟΣ 2. Η *απόσταση αποκοπής (Cutoff distance)* είναι η ακτίνα της γειτονιάς των αντικειμένων στο dataset.

Στο [24] προτείνεται η απόσταση cutoff να θέτεται σε κάποια τιμή ώστε να εξασφαλίζει ότι ο μέσος αριθμός των γειτονικών σημείων είναι περίπου το 1-2% του συνολικού αριθμού των σημείων στο dataset. Επιπλέον, αποδεικνύεται πως τα αποτελέσματα του clustering είναι παρόμοια και συνεπή σε αποστάσεις αποκοπής όπου ο μέσος όρος των

αριθμών των γειτόνων ποικίλει μεταξύ 0.2% και 10%, παρέχοντας μεγάλη ευρωστία στην επιλογή παραμέτρων, μια σημαντική και επιθυμητή ιδιότητα.

ΟΡΙΣΜΟΣ 3. Η *μικρότερη απόσταση από σημείο μεγαλύτερης πυκνότητας (Minimum distance to a higher density point)* - δηλωμένη ως δ - είναι η *μικρότερη* απόσταση προς οποιοδήποτε γείτονα από τα δεδομένα σημεία που έχει *υψηλότερη* τοπική πυκνότητα. Στην ειδική περίπτωση που δεν υπάρχει κανένας, για παράδειγμα το σημείο με την υψηλότερη πυκνότητα από όλα, η τιμή θέτεται στο μικρότερο από τα δ για όλα τα υπόλοιπα αντικείμενα. Κατ' αυτόν τον τρόπο, το δ του αντικειμένου με την υψηλότερη πυκνότητα αναθέτεται τελευταίο.

Για κάθε σημείο *i* στο dataset, η τοπική πυκνότητα p_i και η ελάχιστη απόσταση δ_i σε ένα σημείο υψηλότερης πυκνότητας είναι οι κύριες ιδιότητες που επιτρέπουν την αναγνώριση των υποψήφιων *cluster centers* (*density peaks*).

ΟΡΙΣΜΟΣ 4. Τα *υποψήφια κέντρα των clusters (candidate cluster centers)* είναι αντικείμενα που έχουν τις υψηλότερες τιμές γ_i , το οποίο υπολογίζεται ως εξής:

$$\gamma_i = \rho_i \delta_i \quad ,$$

για κάθε αντικείμενο *i* στο dataset.

Το σκεπτικό πίσω από τα κριτήρια επιλογής για τα υποψήφια *cluster centers* είναι ότι τα γ με τις υψηλότερες τιμές ανήκουν σε σημεία που συνδυάζουν δύο μεγάλες τιμές: υψηλή τιμή τοπικής πυκνότητας και υψηλή τιμή απόστασης από το κοντινότερο σημείο με υψηλότερη τοπική πυκνότητα,μακρυά από άλλα *density peaks.*

Ο αριθμός των clusters *k* μπορεί είτε να προκαθοριστεί – από τον χρήστη – είτε να προσδιοριστεί σύμφωνα με τις γ τιμές των αντικειμένων στο dataset. Ερωτήσεις για το βέλτιστο αριθμό clusters είναι ανεπαρκώς ορισμένες μέχρι στιγμής [25], και δεν απαντώνται σε αυτή την εργασία,η οποία επικεντρώνεται στην αρχική απλή προσέγγιση που περιγράφεται στην ενότητα 3.2, και μέχρι στιγμής αποδεικνύεται πολύ ισχυρή σε διάφορους τομείς.

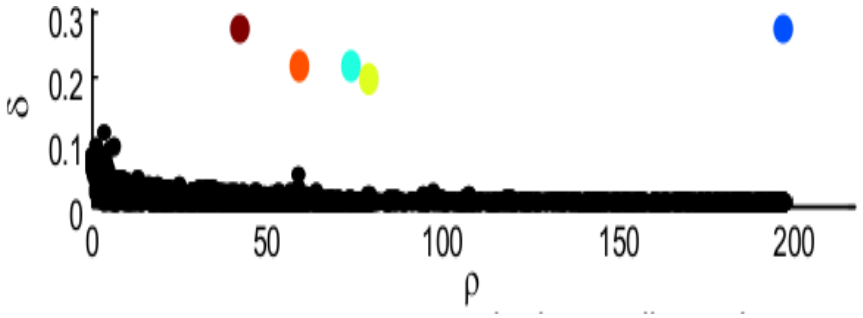
3.2 Η μέθοδος CFSFDP

Στο [24] προτάθηκε μια καινούργια προσέγγιση στο clustering στην οποία θα αναφερόμαστε ως CFSFDP (**C**lustering by **F**ast **S**earch and **F**ind of **D**ensity **P**eaks). Ο αλγόριθμος αποτελείται από δύο βήματα : το πρώτο βήμα βρίσκει τα cluster centers και το δεύτερο αναθέτει τα στοιχεία στα κατάλληλα clusters. Το τελευταίο βήμα αφήνει ανοιχτό το ενδεχόμενο για κάποια σημεία να μείνουν μη καταχωρημένα σε κάποιο cluster και να θεωρούνται ως θόρυβος. Για να βρούμε τα cluster centers, είναι απαραίτητο να υπολογίσουμε δύο τιμές για καθένα από τα αντικείμενα του dataset. Η πρώτη τιμή είναι η *local density* ρ. Η δεύτερη τιμή είναι η *απόσταση* δ από το σημείο στον

κοντινότερο γείτονα με υψηλότερη *local density*. Ο πολλαπλασιασμός αυτών των δύο τιμών δίνει την τιμή γ, της οποίας η λειτουργία είναι η πρόβλεψη και επιλογή των cluster centers.

Στην ουσία, μια υψηλή τιμή γ καταγράφει διαισθητικά ότι το εξεταζόμενο αντικείμενο είναι στο κέντρο μιας πυκνής περιοχής (ρ) και δεν υπάρχει κοντινό σημείο κάποιας πυκνότερης περιοχής (δ) που θα είχε μεγαλύτερη αξία στο να επιλεχθεί για κέντρο του cluster. Ο αριθμός των clusters μπορεί να αποφασιστεί είτε από τον χρήστη είτε με τον ευρετικό τρόπο: οι τιμές γ όλων των σημείων ταξινομούνται σε φθίνουσα σειρά και ο αριθμός των cluster καθορίζεται ως ο αριθμός εκείνων των σημείων, που οι τιμές γ έχουν σημαντική διαφορά με αυτές των γειτόνων τους.

Στην Εικόνα 3.2 παρουσιάζεται το γράφημα απόφασης για το γινόμενο γ ενός συνόλου δεδομένων με τυχαία σημεία. Τα σημεία που αναπαριστώνται με χρώμα έχουν τις υψηλότερες γ τιμές ($\gamma_i = \rho_i \delta_i$) και αντιπροσωπεύουν τα cluster centers. Όσο πιο ψηλά και όσο πιο δεξιά είναι ένα σημείο, τόσο πιο σοβαρή υποψηφιότητα θέτει για να γίνει cluster center.



Εικόνα 3.2.1: Γραφική παράσταση των τιμών ρ και δ για τα σημεία του dataset.

Ο αλγόριθμος είναι αρκετά απλός και ευθύς στη λειτουργία του. Η διαδικασία που ακολουθεί περιγράφεται ως εξής:

i. Υπολογισμός των αποστάσεων για όλα τα ζεύγη σημείων του συνόλου δεδομένων.

ii. Εύρεση της *τοπικής πυκνότητας ρ* κάθε εξεταζόμενου σημείου ως το άθροισμα των αποστάσεών του από τα άλλα, το οποίο πρέπει να είναι μικρότερο από το *cutoff distance.*

iii. Υπολογισμός της *μικρότερης απόστασης από σημείο μεγαλύτερης πυκνότητας δ*

iv. Υπολογισμός του γινομένου *γ*

v. Ταξινόμηση των τιμών *γ* για όλα τα στοιχεία και ανάδειξη αυτών με τις υψηλότερες τιμές ως cluster centers.

vi. Ανάθεση των υπολοίπων σημείων σε clusters με βάση τις τιμές δ ,δηλαδή σε κοντινά clusters.

3.3 Σχετική Έρευνα

Όπως αναφέρθηκε και σε προηγούμενες ενότητες η ομαδοποίηση με βάση την πυκνότητα των σημείων έχει αποκτήσει μεγάλη φήμη και αξιοπιστία, με αποτέλεσμα να γίνονται όλο και περισσότερες δημοσιεύσεις με παραλλαγές των μεθόδων και εφαρμογή τους σε διάφορους τομείς. Πριν περάσουμε στη προσέγγιση του παραπάνω αλγορίθμου από την παρούσα διπλωματική, γίνεται μια σύντομη ανασκόπηση της μέχρι τώρα έρευνας και εξέλιξης πάνω στην επαναστατική αυτή density-based μέθοδο.

Αρχικά στο paper με τίτλο “Adaptive fuzzy clustering by fast search and find of density peaks”[26] προτείνεται μια διαφοροποίηση που δεν επιλέγει ως κέντρα τα σημεία με υψηλότερες τιμές γ αλλά με έναν ευρετικό τρόπο που επιτρέπει περισσότερα του ενός Density Peaks σημεία μέσα στο ίδιο cluster. Χρησιμοποιεί το γράφημα απόφασης με διαφορετικό τρόπο ξεχωρίζοντας τα clusters που περιέχουν θορυβώδη σημεία από τα υπόλοιπα που δεν έχουν.

Ακόμη, στο άρθρο “Clustering by fast search and find of density peaks via heat diffusion” [27] παρουσιάζεται μια πρόταση όπου ο υπολογισμός των τοπικών πυκνοτήτων γίνεται με τη μέθοδο heat diffusion η οποία κάνει εκτίμηση πυρήνων πυκνών περιοχών και τους βλέπει ως λύση μιας διαφορικής εξίσωσης της διάχυσης με βάση το έυρος ζώνης τους.

Στο “Extended fast search clustering algorithm: widely density clusters, no Density Peaks”[28] περιγράφεται μια επέκταση του αλγορίθμου, στην οποία αφού παραχθούν τα clusters με τον CFSFDP υπολογίζει ένας πίνακας ομοιοτήτων τους και συγχωνεύονται πολλές υπο-ομάδες με βαση τη διασυνδεσιμότητά τους,προσπαθώντας με αυτό τον τρόπο να προσπεράσουν τους περιορισμούς της πρώτης μεθόδου.

Επιπλέον, στην έρευνα με τίτλο “Density-based algorithm in MapReduce”[29] παρουσιάζεται ένας νέος CFSFDP βασισμένος στο μοντέλο Map-Reduce και υλοποιημένος στο Hadoop. Αναπαριστά τα δεδομένα με ζευγάρια(key, value). Υλοποιεί και τα 6 βήματα που αναφέρθηκαν προηγουμένως καλώντας πρώτα τη συνάρτηση Map για να γίνουν παράλληλα οι υπολογισμοί και στη συνέχεια τη Reduce για να συγκεντρωθούν οι values με βάση τα keys. Παρουσιάζει πολύ καλά αποτελέσματα με βάση τους χρόνους αλλά μόνο για αριθμητικά δεδομένα.

Στο “Improved density peak clustering for large datasets”[30] όπως υποδηλώνει και το όνομα γίνεται ανάπτυξη του αλγορίθμου για μεγάλα σύνολα δεδομένων, επιδιώκοντας να ξεπεραστούν οι περιορισμοί των παραμέτρων του [24] και να προσαρμοστεί σε τεράστια datasets.

Τέλος, στο “Clustering in the Face of Fast Changing Streams”[31], ερευνάται η δυνατότητα ανάπτυξης του αλγορίθμου για μη στατικά δεδομένα. Αυτό γίνεται με πλη-

θώρα προσαρμογών και τμηματοποιήσεων των clusters ώστε να υποδέχονται τα καινούργια streaming στοιχεία.

Κεφάλαιο 4

Κατανεμημένη Υλοποίηση Αλγορίθμων

Στο κεφάλαιο αυτό, περιγράφεται ο τρόπος ανάπτυξης των τριών αλγορίθμων στο Spark με σκοπό να υπάρξει λύση στο πρόβλημα που τέθηκε στην ενότητα 1.2. Η ενότητα 4.1 εξηγεί τον τρόπο επιλογής των τριών αλγορίθμων ενώ οι επόμενες 4.2, 4.3, και 4.4 παρουσιάζουν τη διαδικασία ανάπτυξης στο Spark του K-means, του DBSCAN και του CFSFDP. Πρέπει να σημειωθεί πως οι τρεις αυτές υλοποιήσεις επεξεργάζονται στατικά δεδομένα (batched dataset) και όχι ροή δεδομένων (data stream).

4.1 Επιλέγοντας αλγορίθμους

Οι Density-based αλγόριθμοι παρουσιάζουν ιδιαίτερο ενδιαφέρον λόγω του τρόπου λειτουργίας τους και της δυνατότητας να αναζητούν και να βρίσκουν clusters μη σταθερών σχημάτων. Εκτός από τον CFSFDP που αναλύθηκε στο Κεφάλαιο 3 , ο DBSCAN είναι ένας παραδοσιακός και ευρέως αποδεκτός αλγόριθμος, όπως επίσης και ο κλασσικός K-means που όμως δεν εκτιμά τις πυκνότητες αλλά μόνο τις αποστάσεις.

Πριν παρουσιαστούν οι μεθοδολογίες και των τριών,πρέπει να σημειωθεί πως σε αυτή τη εργασία, οι υπολογισμοί των αποστάσεων λαμβάνουν υπόψιν την Ευκλίδεια μετρική και όχι κάποια διαφορετική. Επιπλέον, στο κεφάλαιο αυτό δεν περιγράφεται η δημιουργία dataset και η απαραίτητη προεπεξεργασία δεδομένων,αλλά δίνεται έμφαση στις εργασίες του εκάστοτε αλγορίθμου.

4.2 K-means στο Spark

Στην ενότητα 2.1.2 παρουσιάστηκε εν συντομία ένας από τους πιο παραδοσιακούς αλγορίθμους για clustering, ο K-means. Εδώ θα δωθούν περισσότερες λεπτομέρειες για τη λειτουργία του και τον τρόπο ανάπτυξης του στο Spark. Η διαδικασία που ακολουθείται στην σειριακή εκδοχή του αλγορίθμου,θεωρώντας ότι έχουμε εισόδους $x_1, x_2, ..., x_n$ και μια τιμή k αποτελείται από τα εξής βήματα:

1. Διάλεξε k τυχαία σημεία ως cluster centers,που ονομάζονται πλέον κεντροειδή.
2. Ανάθεσε κάθε x_i στο κοντινότερο cluster υπολογίζοντας την απόσταση του από κάθε κεντροειδές.
3. Μετακίνησε το κέντρο του cluster υπολογίζοντας τον μέσο όρο των ομαδοποιημένων σημείων.
4. Επανέλαβε τα βήματα 2 και 3 έως ότου κανένα cluster να μην αλλάζει.

Η παραλληλοποιημένη εκδοχή του K-means που ακολουθήσαμε για την υλοποίησή του στο Spark περιγράφεται ως ακολούθως:

Βήμα 1.

Έχοντας μια συλλογή με σημεία επιλέγουμε με τυχαίο τρόπο τα αρχικά cluster centers. Επίσης,κατανέμουμε τη συλλογή σε partitions με τη μέθοδο ***parallelize*** ώστε να δημιουργήσει μια RDD.

Βήμα 2.

Για κάθε στοιχείο της συλλογής καλούμε το μετασχηματισμό ***map*** στον οποίο εφαρμόζουμε τη συνάρτηση δημιουργίας των clusters (*formCluster)* και επιστρέφεται μια καινούργια RDD. Η συνάρτηση αυτή υπολογίζει την απόσταση κάθε σημείου της RDD από τα επιλεγμένα κέντρα και το αναθέτει στο cluster του πιο κοντινού.

Βήμα 3.

Στην καινούργια RDD που δημιουργήθηκε προηγουμένως εφαρμόζουμε το μετασχηματισμό ***reduceByKey*** ο οποίος υπολογίζει το μέσο όρο των σημείων του κάθε cluster και δημιουργεί μια λίστα με τα καινούργια κεντροειδή.

Βήμα 4.

Επαναλαμβάνουμε τα βήματα **2** και **3** έως ότου οι αλλαγές στα κέντρα είναι πολύ μικρές και πιο συγκεκριμένα συγκλίνουν σε ένα πολύ μικρό κατώφλι.

4.3 DBSCAN στο Spark

Σε προηγούμενο κεφάλαιο, και συγκεκριμένα στην ενότητα 2.1.3 συζητήθηκαν οι βασικές ιδιότητες και ο τρόπος λειτουργίας του DBSCAN. Έγινε, επιπλέον, η περιγραφή της διαδικασίας που ακολουθεί ο αλγόριθμος στη σειριακή του εκτέλεση. Στην εργασία όμως, η μέθοδος αναπτύχθηκε με κατανεμημένο τρόπο και ακολουθήθηκαν οι εξής εργασίες:

Βήμα 1.

Έχοντας ένα σύνολο δεδομένων αποτελούμενων από σημεία, το διανέμουμε σε partitions με τη μέθοδο ***parallelize*** και δημιουργούμε την RDD. Οι διανομές των στοιχείων όμως γίνονται με μια κλάση διαχωριστή (partitioner) η οποία πρώτα αποθηκεύει το dataset στην μνήμη με τη λειτουργιά ***cache*** και προσπαθεί να χωρίσει τα δεδομένα σε ισομεγέθη σετ με βάση την απόκλισή τους η οποία υπολογίζεται παράλληλα με τις εντολές ***map*** και ***filter***. Η διάσπαση αυτή πετυχαίνεται με τη δημιουργία γεωμετρικών κουτιών, τα οποία ορίζουν τα όρια της κάθε partition και επαναλαμβάνεται ώσπου να φτάσει στο μέγιστο επιθυμητό αριθμό partitions. Τέλος, συναθροίζει αυτά τα κουτιά σε ένα με την εντολή ***aggregate.***

Βήμα 2.

Αυτά τα κουτιά επεκτείνονται δύο φορές την τιμή της ακτίνας του DBSCAN και χρησιμοποιούνται για να δημιουργηθούν γειτονιές σημείων ελέγχοντας το κάθε ένα με την εντολή ***filter***, αν έχει την ίδια ετικέτα κουτιού με τα υπόλοιπα γειτονικά σημεία. Έπειτα οι γειτονιές συγχωνεύονται με αποτέλεσμα να υπάρχουν διπλότυπα σημεία λόγω της επικάλυψης που δημιουργούν τα διευρυμένα κουτιά μεταξύ τους. Αυτό αντιμετωπίζεται με τον επαναδιαχωρισμό της RDD με την εντολή ***partitionBy*** με βάση το id της κάθε γειτονιάς.

Βήμα 3.

Σε κάθε partition χρησιμοποιούμε τη λειτουργία ***mapPartitions*** ,εφαρμόζοντας τον αλγόριθμο DBSCAN του sklearn πακέτου της python και κάθε σημείο φέρει το *ID* της partition, του cluster κι αν είναι κεντρικό σημείο. Διατηρούμε ξανά τα καινούργια δεδομένα στη μνήμη με την εντολή ***cache.***

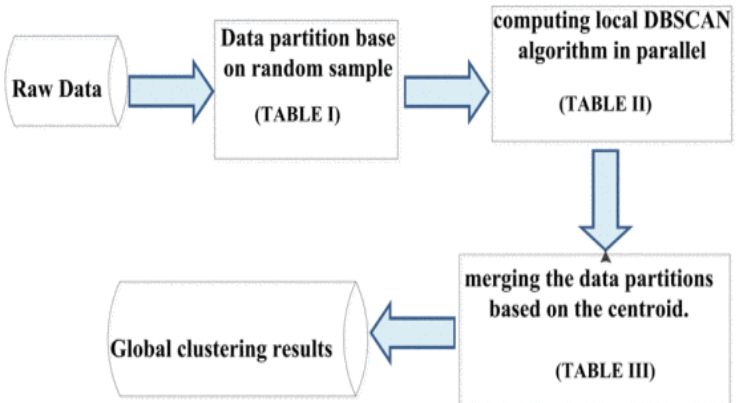
Βήμα 4.

Στη συνέχεια τα σημεία ομαδοποιούνται και συγχωνεύονται με την λειτουργία ***groupByKey***, αλλά μπορεί να υπάρξει σύγχυση ανάμεσα στα clustering *IDs* και στα partition-level *IDs.* Επιπλέον,επειδή ο DBSCAN είναι άπληστος αλγόριθμος πολλά σημεία που δεν είναι κεντρικά μπορεί να ανατεθούν σε οποιοδήποτε cluster εφόσον είναι προσβάσιμα από πολλαπλά cluster centers. Την καινούργια RDD την κρατάμε στη μνήμη ως labeled για το τελευταίο βήμα.

Βήμα 5.

Τέλος, για κάθε σημείο στη labeled RDD χρησιμοποιούμε το ***map*** και το ***sortByKey*** εφαρμόζοντας μια συνάρτηση επανασχεδιασμού των clusters η οποία λύνει την προηγούμενη σύγχυση. Για κάθε σημείο με πολλαπλά *IDs* μόνο το πρώτο θα λαμβάνεται υπόψιν.

Πρέπει να σημειωθεί πως για την παραλληλοποίηση του DBSCAN μελετήθηκε εκτενώς το paper "A Parallel DBSCAN Algorithm Based on Spark"[32]. Ωστόσο, υπάρχουν μεγάλες διαφοροποιήσεις και πιο συγκεκριμένα ο τρόπος διαχωρισμού των δεδομένων στο ***Βήμα 1*** και η συγχώνευση των cluster και partition *IDs* στο ***Βήμα 4*** *και* **Βήμα 5.**



Εικόνα 4.3.1: Το προγραμματιστικό πλαίσιο του κατανεμημένου DBSCAN.

4.4 CFSFDP στο Spark

Στο κεφάλαιο 3 και συγκεκριμένα στην ενότητα 3.2 παρουσιάστηκε αναλυτικά ο αλγόριθμος "**C**lustering by **F**ast **S**earch and **F**ind of **D**ensity **P**eaks" [24] και παρατέθηκε η μεθοδολογία, που ακολουθεί αυτή η πρόταση για clustering, στη ανάπτυξη της σε σειριακό περιβάλλον εκτέλεσης. Σε παράλληλο προγραμματιστικό περιβάλλον, δηλαδή στο Spark, η διαδικασία που ακολουθήσαμε περιγράφεται ως ακολούθως:

Βήμα 1.

Έχοντας δημιουργήσει ένα σύνολο από δεδομένα το κατανέμουμε παράλληλα (***parallelize***) ώστε να δημιουργηθεί η RDD και το διατηρούμε στη μνήμη (***cache***) για εύκολη πρόσβαση στις επόμενες ενέργειες. Τα στοιχεία αποθηκεύονται ως λίστες με τα απαραίτητα χαρακτηριστικά όπως id ,συντεταγμένες, local density, cluster κ.ο.κ.

Βήμα 2.

Για κάθε στοιχείο της RDD καλείται η ***map*** η οποία εφαρμόζει τη *set_density()*. Η συνάρτηση αυτή αθροίζει όσα ζεύγη αποστάσεων είναι μικρότερα από το

cutoff_distance και επιστρέφει την τοπική πυκνότητα ρ κάθε σημείου. Όλα τα στοιχεία της καινούργιας RDD επιστρέφονται σε λίστα με την ενέργεια ***toLocalIterator***.

Βήμα 3.

Στην RDD των σημείων με υπολογισμένες τις τοπικές πυκνότητες καλείται η ***map***, με τη συνάρτηση set_distance_to_higher_density_point(), η οποία συγκρίνει τις αποστάσεις κάθε σημείου με τη λίστα των σημείων μεγαλύτερης πυκνότητας και αποθηκεύει την απόσταση δ και το id από το πιο "πυκνό" και κοντινό σημείο.

Βήμα 4.

Η δημιουργημένη RDD μετά τον υπολογισμό των ανωτέρω παραμέτρων διατηρείται στη μνήμη με την εντολή ***cache***. Υπολογίζεται το γινόμενο $\gamma_i = \rho_i \delta_i$ (συζητήθηκε αναλυτικά στην ενότητα 3.2) και γίνεται παράλληλη ταξινόμηση των υπολογισμένων γ με τη λειτουργία ***sortBy***. Με βάση αυτή την ταξινόμηση θέτονται τα cluster centers, τα λεγόμενα D*ensity Peaks.*

Βήμα 5.

Τελικό βήμα είναι η ανάθεση των υπολοίπων σημείων σε clusters. Επαναληπτικά ελέγχοναι όλα τα εναπομείναντα στοιχεία με τη λειτουργία ***filter*** αν ανήκουν σε συστάδα και αν όχι καλείται η ***map*** ώστε να τοποθετηθεί παράλληλα κάθε σημείο της RDD στο κατάλληλο cluster. Δηλαδή στο cluster του πιο κοντινού σημείου με τη μεγαλύτερη πυκνότητα.

Υπάρχει μια σημαντική διαφοροποίηση σχετικά με το αναφερθέν paper και την πρότασή του και αφορά τον αριθμό των clusters τον οποίο προκαθορίζουμε από πριν. Δε χρησιμοποιούμε κάποιον ευρετικό τρόπο που να αποφασίζει για αυτό με βάση τις αποκλίσεις των τιμών γ των σημείων, να αποκλείει δηλαδή σημεία με σχετικά χαμηλές πυκνότητες.

Κεφάλαιο 5

Αποτίμηση Επίδοσης Αλγορίθμων για Τυχαία Σημεία

Σε αυτό το κεφάλαιο επιδιώκεται η ερμηνεία και επίλυση του προβλήματος που τέθηκε στην αρχή (1.2). Ο κύριος σκοπός των πειραμάτων είναι να ελεγχθεί αν οι τρεις αλγόριθμοι είναι υλοποιημένοι με σωστό τρόπο στο Spark, αν αυξάνουν την αποδοτικότητα τους και αν παράγουν ποιοτικά clusters.

5.1 Πειραματική ερμηνεία των παραμέτρων του CFSFDP

Όπως αναφέρθηκε και στο Κεφάλαιο 3 , μεγάλο μέρος της παρούσας διπλωματικής αφιερώθηκε στη μελέτη density-based αλγορίθμων και την ανάπτυξη τους στο Spark. Εδώ θα προσπαθήσουμε να εξάγουμε κάποια συμπεράσματα και να ερμηνεύσουμε τη συμπεριφορά του CFSFDP και των παραμέτρων του, όπως επίσης να σχολιάσουμε και κάποια στοιχεία του Spark. Η ανάλυση αφορά μόνο τους χρόνους εκτέλεσης και όχι την ποιότητα των παραγόμενων clusters τα οποία διαπιστώθηκε πως δεν έχουν καμία διαφορά. Πρέπει ,επίσης, να σημειωθεί ότι το dataset για τα πειράματα σε αυτή την ενότητα είναι τυχαία σημεία στο δισδιάστατο επίπεδο των αξόνων και στο σύνολο από 0 έως 10.

5.1.1 Πρώτη απλή υλοποίηση

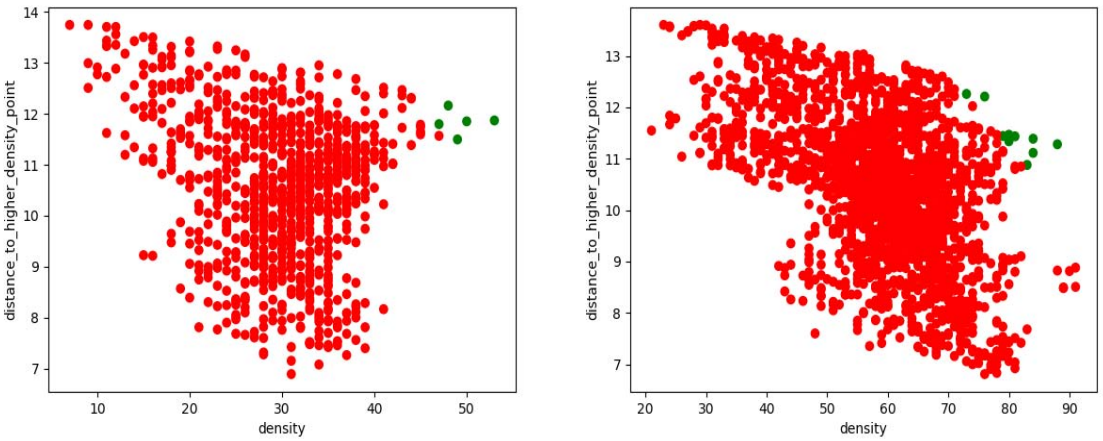
Αρχικά ήταν απαραίτητο να γίνει ένας έλεγχος με απλές τυπικές τιμές παραμέτρων για να διαπιστωθεί η εγκυρότητα του σκεπτικού και του ανεπτυγμένου αλγορίθμου στο Spark. Ο πιο ασφαλής τρόπος για να πραγματοποιηθεί αυτό είναι η εγκυρότητα του

γινομένου $\gamma_i = \rho_i \delta_i$ το οποίο λαμβάνεται υπόψιν για την ανάδειξη των cluster centers

Για δύο απλές περιπτώσεις clustering με τιμές παραμέτρων:

- Αριθμός clusters = 5 στην πρώτη περίπτωση και 10 στη δεύτερη,
- Αριθμός σημείων = 1000 στην πρώτη περίπτωση και 2000 στη δεύτερη,
- Cutoff distance = 1 και στις δύο περιπτώσεις.

Εξάγουμε το γράφημα απόφασης και για τις δύο υλοποιήσεις το οποίο απεικονίζει στον άξονα των x τις τοπικές πυκνότητες ρ των σημείων και στον άξονα y τις αποστάσεις προς σημεία μεγαλύτερης πυκνότητας (Εικόνα 5.1.1). Με κόκκινο χρώμα αναπαριστώνται τα σημεία που δεν έχουν εισχωρήσει ακόμα σε κάποιο cluster, ενώ με πράσινο είναι cluster centers ή αλλιώς τα Density Peaks. Συμπεραίνεται εύκολα η ορθή επιλογή των κέντρων καθώς όσο πιο δεξιά(μεγάλο ρ) και όσο πιο πάνω(μεγάλο δ) είναι ένα σημείο,τόσο μεγαλύτερη πιθανότητα έχει να γίνει κέντρο κάποιου cluster. Επιπλέον,βλέπουμε ότι στη δεξιά εικόνα με τα περισσότερα σημεία, τα κέντρα έχουν μεγαλύτερες πυκνότητες.



Εικόνα 5.1.1: Το γράφημα απόφασης των Density Peaks στην πρώτη και στη δεύτερη περίπτωση.

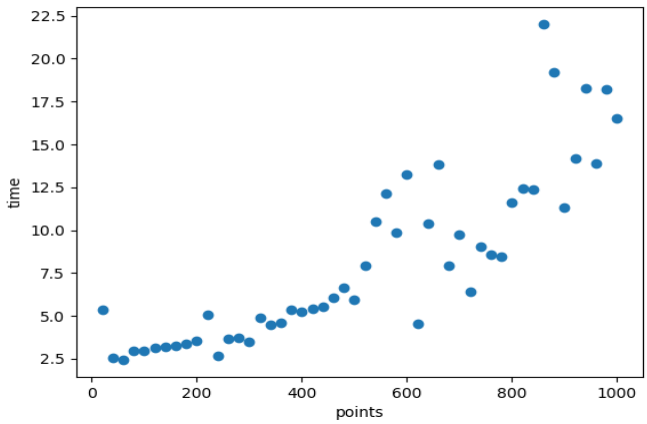
5.1.2 Πολυπλοκότητα μεταβαλλόμενου αριθμού σημείων

Σε αυτή την ενότητα εμβαθύνουμε στην αξιολόγηση της υλοποίησης για αυξανόμενο αριθμό σημείων. Αξιολογείται η επίδοση του CFSFDP με βάση τους χρόνους που κάνει για να ομαδοποιήσει δεδομένα σε σταθερό αριθμό συστάδων αλλά όχι και σημείων.

Οι προδιαγραφές της συγκεκριμένης υλοποίησης έχουν τις εξής εισόδους:

- Αριθμός clusters = 5,
- Αριθμός σημείων = από 20 έως 1000 με βήμα αλλαγής 20,
- Cutoff distance = 1.

Η γραφική παράσταση (Εικόνα 5.1.2) απεικονίζει στον άξονα x τα διαφορετικά σύνολα δεδομένων και στον άξονα y το χρόνο εκτέλεσης σε δευτερόλεπτα. Παρατηρούμε ότι ο χρόνος εκτέλεσης αυξάνεται με σχεδόν εκθετικό ρυθμό καθώς έχει πολλές αποκλίσεις, ενώ ο μέσος χρόνος για όλες τις περιπτώσεις είναι 8,16 δευτερόλεπτα. Είναι απολύτως λογική αυτή η συμπεριφορά καθώς όσο αυξάνονται τα σημεία τόσο πιο χρονοβόρα γίνεται η διαδικασία ανάθεσης σημείων σε clusters, διότι ελέγχονται επαναληπτικά τα μη συσταδοποιημένα στοιχεία.



Εικόνα 5.1.2: Πολυπλοκότητα του CFSFDP για μεταβαλλόμενα dataset σημείων.

5.1.3 Πολυπλοκότητα μεταβαλλόμενου αριθμού clusters

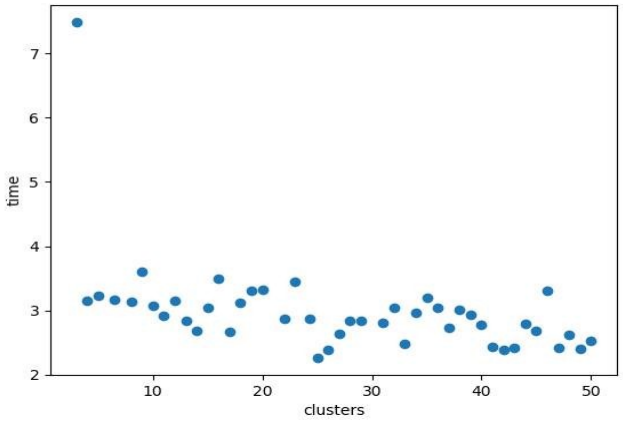
Ενώ στην προηγούμενη ενότητα (5.1.2) είδαμε πως ανταποκρίνεται ο αλγόριθμος σε εκτελέσεις για διαφορετικό αριθμό σημείων, εδώ πειραματιζόμαστε με τον αριθμό των clusters που έχει να δημιουργήσει κάθε φορά το πρόγραμμα.

Οι είσοδοι που χρησιμοποιήθηκαν για αυτή την περίπτωση έχουν ως εξής:

- Αριθμός clusters = από 3 έως 50 με βήμα αλλαγής 1,
- Αριθμός σημείων = 300,
- Cutoff distance = 1.

Η γραφική παράσταση (Εικόνα 5.1.3) απεικονίζει στον άξονα x τους διαφορετικούς αριθμούς των clusters και στον άξονα y το χρόνο εκτέλεσης σε δευτερόλεπτα. Ο μέσος χρόνος εκτέλεσης για τα διαφορετικά μεγέθη των clusters είναι 2,98 δευτερόλεπτα. Βλέπουμε πως η πρώτη εκτέλεση που διήρκησε διπλάσιο περίπου χρόνο από τις υπόλοιπες και αυτό οφείλεται στο γεγονός ότι φορτώνει όλα τα δεδομένα που έχουν παραχθεί, δηλαδή το dataset, και το διατηρεί στην τοπική μνήμη. Αντιθέτως οι επόμενες εκτελέσεις παίρνουν κατευθείαν από τη μνήμη τα δεδομένα χωρίς να χρειάζεται να φορτώσουν από την RDD. Ο λόγος όμως που είναι σταθερή η διάρκεια εκτέλεσης

και ανεξάρτητη από το πόσα clusters επιθυμεί ο χρήστης, είναι ότι ταξινομούνται τα γινόμενα γ όλων των στοιχείων. Οπότε, είτε χρειαστούμε τα δέκα είτε τριάντα καλύτερα, η ανάθεση των cluster centers χρειάζεται τον ίδιο χρόνο σε αντίθεση με την ανάθεση μεταβαλλόμενων σημείων σε clusters της προηγούμενου πειράματος (5.1.2).



Εικόνα 5.1.3: Πολυπλοκότητα του CFSFDP για μεταβαλλόμενα clusters.

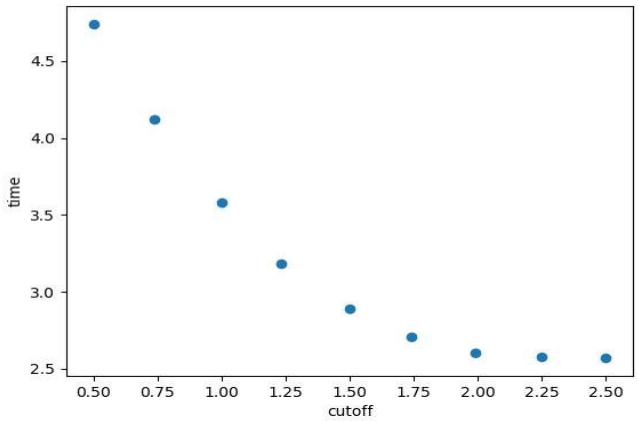
5.1.4 Πολυπλοκότητα μεταβαλλόμενου cutoff distance

Μια ενδιαφέρουσα παράμετρος είναι και η απόσταση αποκοπής η οποία χρησιμοποιείται για τον υπολογισμό των τοπικών πυκνοτήτων όπως αναλύθηκε στις ενότητες 3.1 και 3.2. Σκοπός αυτής της υλοποίησης είναι να αξιολογηθεί το αντίκτυπο που έχουν οι διαφορετικές τιμές του cutoff στους χρόνους εκτέλεσης.

Οι τιμές των παραμέτρων που εισάγονται σε αυτή την υλοποίηση έχουν ως εξής:

- Αριθμός clusters = 5,
- Αριθμός σημείων = 100,
- Cutoff distance = Από 0.50 έως 2.50 με βήμα αλλαγής 0.25.

Σχεδιάζοντας τη γραφική παράσταση (Εικόνα 5.1.4) διακρίνουμε πόσο σημαντική μετρική είναι το cutoff ως προς τον απαιτούμενο χρόνο του clustering των δεδομένων. Όσο μικρότερο είναι το cutoff τόσο πιο δύσκολο είναι να το ξεπερνάνε οι αποστάσεις ζευγαριών σημείων και να αποκτούν έτσι μεγαλύτερη τοπική πυκνότητα. Επομένως,σε ένα θεωρητικά μη πυκνό γράφημα γίνεται πιο δύσκολα η εύρεση των density peaks κι αυτό κοστίζει περισσότερο στην όλη διαδικασία όσον αφορά το χρόνο.



Εικόνα 5.1.4: Πολυπλοκότητα του CFSFDP για μεταβαλλόμενες cutoff αποστάσεις.

5.1.5 Ένα σχόλιο για το Spark

Όπως έχει προαναφερθεί σε αυτή την εργασία, το Spark είναι μια κατανεμημένη διεπαφή εφαρμογών που προσφέρει γρηγορότερη επεξεργασία δεδομένων,ανοχή σε σφάλματα των worker nodes και ανάκτηση δεδομένων κατευθείαν από τη μνήμη. Όλα αυτά τα πλεονεκτήματα,αν δεν διαμορφωθούν κατάλληλα, μπορεί να σπαταληθούν άσκοπα. Για παράδειγμα, αν ρυθμίσουμε την υλοποίηση να χρησιμοποιήσει μόνο τον έναν πυρήνα του συστήματος, θα έχει ένα worker node μαζί με τον driver και στην ουσία οι υπολογισμοί θα γίνονται σειριακά. Αντίστοιχα,μπορεί να έχει δύο ή τέσσερις worker nodes και πάει λέγοντας, ανάλογα με το σύστημα ή το cluster που έχει δημιουργηθεί. Πιο συγκεκριμένα,για το απλό παράδειγμα της ενότητας 5.1.1 για τα 1000 σημεία, έχουμε τους εξής χρόνους εκτέλεσης:

- 1 worker node: χρόνος = 42.12 δευτερόλεπτα,
- 2 worker nodes: χρόνος = 33.30 δευτερόλεπτα,
- 4 worker nodes: χρόνος = 23.56 δευτερόλεπτα.

Επίσης η ανάκτηση δεδομένων από τη μνήμη είναι ένα αξιοσημείωτο χαρακτηριστικό. Η δυνατότητα διατήρησης ολόκληρης της RDD σε ένα προκαθορισμένο επίπεδο αποθήκευσης, είναι μια τεχνική βελτιστοποίησης του Spark που παρέχεται με τις εντολές ***cache()*** και ***persist()***. Όπως εξηγήθηκε στην ενότητα 4.4 στις εργασίες που ακολουθεί ο αλγόριθμος στο Spark, χρησιμοποιήθηκε η μέθοδος ***cache()***. Για να γίνει αντιληπτή η διαφορά της αποθήκευσης ή όχι των RDD στη μνήμη, πρέπει να δούμε τους χρόνους για το παράδειγμα με τα 1000 σημεία της ενότητας 5.1.1 υλοποιημένο χωρίς τη μέθοδο ***cache()***:

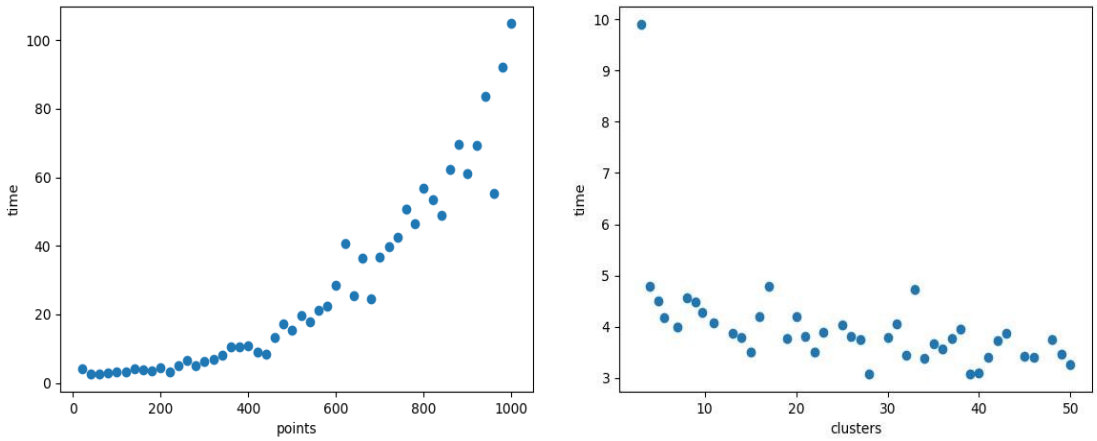
- 1 worker node: χρόνος = 156.25 δευτερόλεπτα,
- 2 worker nodes: χρόνος = 104.07 δευτερόλεπτα,

- 4 worker nodes: χρόνος = 92.37 δευτερόλεπτα.

Εκτός από αυτό, γίνεται και η σύγκριση των παραδειγμάτων των ενοτήτων 5.1.2 και 5.1.3, οι χρόνοι των οποίων απεικονίζονται στις εικόνες 5.1.5α και 5.1.5β. Όταν αλλάζει ο αριθμός των clusters ο μέσος χρόνος είναι 3,67 ο οποίος δεν έχει μεγάλη διαφορά από την υλοποίηση με ***cache***,πράγμα που φαίνεται και αν συγκρίνουμε τις Εικόνες 5.1.3 και 5.1.5β,παρά μόνο στην πρώτη εκτέλεση που είναι κατά 2 περίπου δευτερόλεπτα πιο αργή.

Από την άλλη όταν έχουμε αυξανόμενο αριθμό σημείων στο dataset ο μέσος χρόνος για όλες τις εκτελέσεις ανεβαίνει υπερβολικά (27,62) χωρίς την ***cache***. Αυτό διαπιστώνεται και από τις Εικόνες 5.1.2 και 5.1.5α, και η αιτία είναι ότι αντί να αποθηκεύσει την RDD στην μνήμη την πρώτη φορά και να είναι εύκολα προσβάσιμη για τους επόμενους υπολογισμούς, ξαναφορτώνει τα δεδομένα της κάθε φορά και για κάθε υπολογισμό.

Περισσότερα για την αναγκαιότητα ανάκτησης της RDD από τη μνήμη θα παρουσιαστούν στην Ενότητα 6.3.4 για την επεξεργασία ενός τεράστιου dataset πραγματικού κόσμου.



Εικόνα 5.1.5: Πολυπλοκότητα χωρίς cache(partitions) του CFSFDP για μεταβαλλόμενα clusters και σημεία.

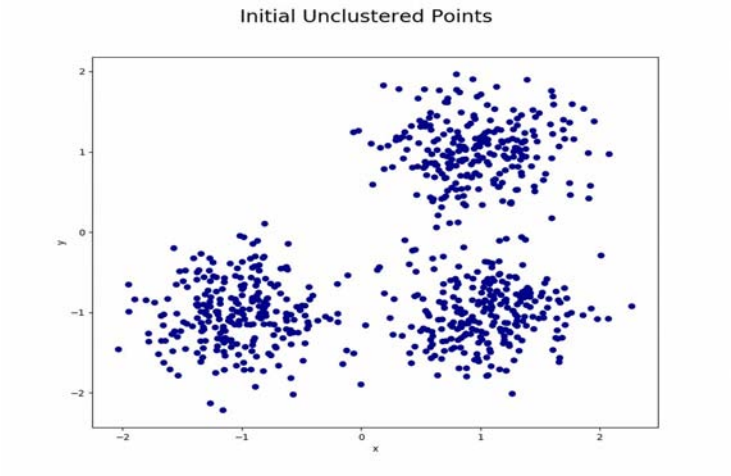
5.2 Συγκριτική αξιολόγηση αλγορίθμων

Η ενότητα αυτή ελέγχει και αξιολογεί τα αποτελέσματα των αλγορίθμων CFSFDP, DBSCAN και K-means που περιγράφηκαν στο Κεφάλαιο 4 . Παρουσιάζονται οι επιδόσεις των κατανεμημένων αυτών μεθόδων και συγκρίνονται τόσο οι χρόνοι όσο και η ποιότητα των παραγόμενων clusters.

5.2.1 Το dataset

Ένα σύνολο δεδομένων αποτελούμενο από τυχαία σημεία δεν είναι πάντα καλό μέτρο για να ελεγχθεί ένας αλγόριθμος ομαδοποίησης, πόσο μάλλον όταν ερευνούμε κάποιον density-based. Γι' αυτό το λόγο το dataset σε αυτή την ενότητα παράγεται με τη Γκαουσιανή Κατανομή και συγκεκριμένα από τη βιβλιοθήκη *sklearn* της python που δημιουργεί ισοτροπικές μάζες [33].

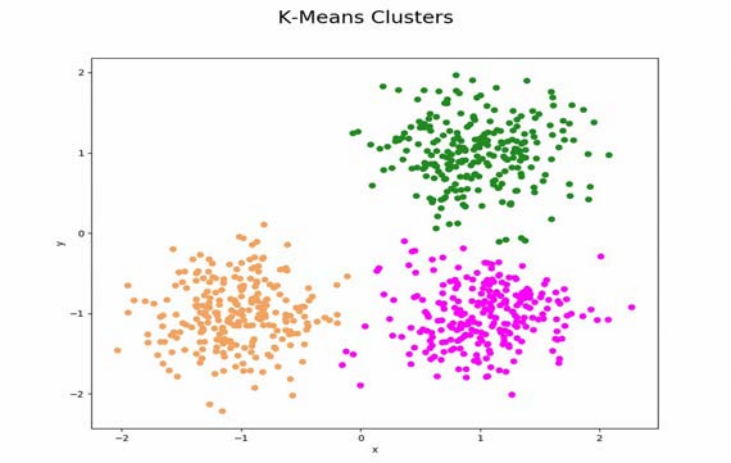
Αυτό το μοντέλο δημιουργίας δεδομένων αποτελεί μια ευρέως διαδεδομένη τεχνική αποτίμησης αλγορίθμων μηχανικής μάθησης. Πιο αναλυτικά, στην παρούσα εργασία δημιουργούνται 750 σημεία μοιρασμένα με βάση την κατανομή Gauss σε 3 άμορφες μάζες ανεπτυγμένες γύρω από 3 κέντρα (Εικόνα 5.2.1).



Εικόνα 5.2.1: Το αρχικό dataset 3 μαζών

5.2.2 Ανάλυση των παραγόμενων clusters

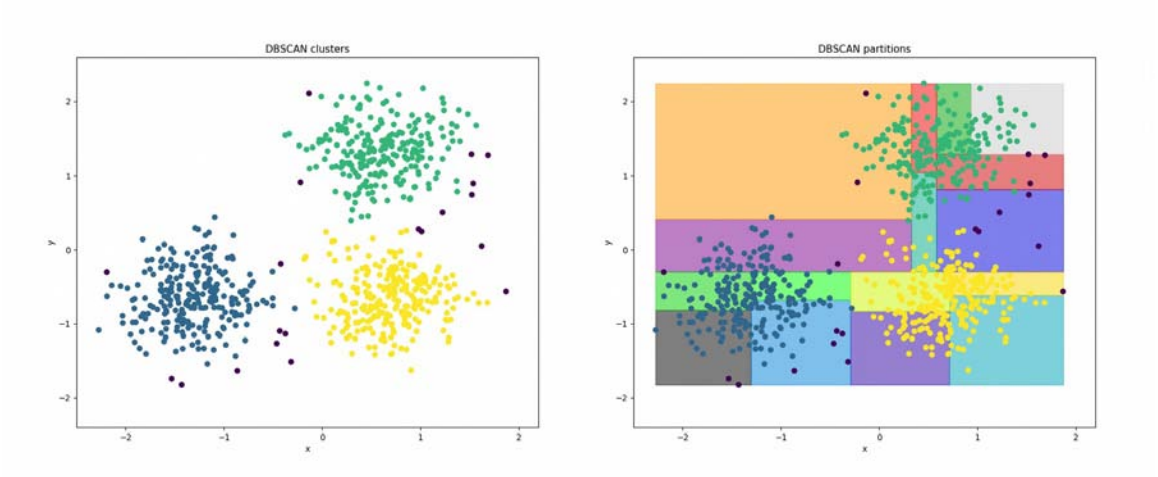
Ξεκινάμε από τον πιο απλό και κλασικό αλγόριθμο K-means, τα δημιουργημένα cluster του οποίου αποτυπώνονται στην Εικόνα 5.2.2. Παρατηρούμε πως έγινε σωστή ανάθεση των clusters σε ικανοποιητικό βαθμό. Οι συστάδες είναι ευδιάκριτα διαχωρι-



Εικόνα 5.2.2: Δημιουργία clusters με K-means.

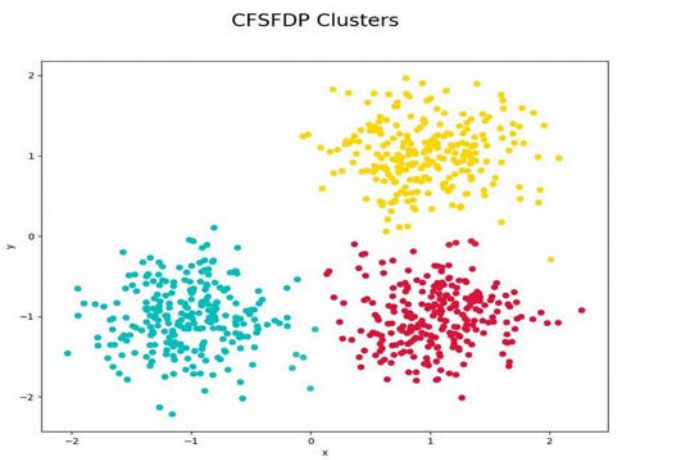
σμένες μεταξύ τους παρά την τυχαία ανάθεση κέντρων στην αρχή, γεγονός που αποδεικνύει ότι ο επαναληπτικός επανυπολογισμός κέντρων ήταν πετυχημένος.

Στη συνέχεια αξιολογείται η εγκυρότητα των παραγόμενων clusters από τον δεύτερο αλγόριθμο που συζητήθηκε στο κεφάλαιο 4, τον DBSCAN. Ο αριθμός της ακτίνας eps είναι 0.3 και η τιμή του MinPts είναι 10. Από την Εικόνα 5.2.3α παρατηρούμε ότι τα σημεία ανατέθηκαν στις σωστές ομάδες. Επιπλέον απεικονίζονται με μωβ χρώμα τα σημεία θορύβου, τα οποία σωστά όπως διακρίνεται δεν ανατέθηκαν σε κάποιο cluster λόγω της θέσης τους σε αραιές περιοχές. Τέλος, στην Εικόνα 5.2.3β αναπαριστώνται οι διανομές του συνόλου των σημείων στις οποίες υπολογίστηκε παραλληλοποιημένα ο DBSCAN.



Εικόνα 5.2.3: Τα clusters και τα partitions του DBSCAN.

Τελευταίος αλγόριθμος που θα αποτιμηθεί είναι ο έτερος density-based και κύριο αντικείμενο μελέτης της εργασίας, ο CFSFDP. Εκτός από τους χρόνους εκτελέσεών του για τυχαία σημεία και την ανάλυση των ιδιοτήτων του που έγινε στην ενότητα 5.1, πρέπει να ελεγχθεί και ως προς τα clusters που δημιουργεί για όχι τελείως τυχαία σημεία. Παρατηρώντας την Εικόνα 5.2.4, συμπεραίνουμε πως λειτούργησε ικανοποιητικά, βρήκε τα σωστά clusters και δεν είχε αποκλίσεις σημείων ή "παράσιτα".



Εικόνα 5.2.4: Δημιουργία clusters με CFSFDP.

Εκτός από τη ανωτέρω σύγκριση των παραγόμενων clusters, με την οποία διαπιστώσαμε ότι η υλοποίηση των αλγορίθμων στο Spark λειτουργεί αποτελεσματικά, παραθέτουμε και τους χρόνους εκτέλεσής τους για να υπάρχει ολοκληρωμένη εικόνα και αποτίμηση της επίδοσης τους. Η διάρκεια ομαδοποίησης για τον καθένα ξεχωριστά έχει ως εξής (σε δευτερόλεπτα):

- K-means: χρόνος = 7,62
- DBSCAN: χρόνος = 22,52
- CFSFDP: χρόνος = 20,86

Προσπαθώντας να εξάγουμε κάποια χρήσιμα συμπεράσματα μετά από αυτούς τους χρόνους πρώτα πρέπει να αναφερθούν δύο παρατηρήσεις. Πρώτον, ότι παρά το μεγάλο αριθμό σημείων(750) στον οποίο εφάρμοσαν clustering οι αλγόριθμοι, οι χρόνοι εκτέλεσης σε αυτή την κατανεμημένη υλοποίηση είναι θεαματικά μικρότεροι σε σχέση με τις υπάρχουσες σειριακές υλοποιήσεις. Δεύτερον, ότι ο K-means ομαδοποίησε τα στοιχεία σχεδόν τρεις φορές πιο γρήγορα από τους άλλους δύο, πράγμα που ήταν αναμενόμενο λόγω των μεθόδων που χρησιμοποιούν. Για να γίνει καλύτερα αντιληπτή αυτή η διαφορά αναφέρουμε τις χρονικές πολυπλοκότητες των τριών αλγορίθμων [34][35] :

- Πολυπλοκότητα K-means = $O(Nkq)$, όπου N ο αριθμός των σημείων, k αριθμός των κέντρων και q οι επαναλήψεις.
- Πολυπλοκότητα DBSCAN = $O(N^2)$, όπου N ο αριθμός των σημείων.
- Πολυπλοκότητα CFSFDP = $O(N^2)$, όπου N ο αριθμός των σημείων

Ο μεν πρώτος επικεντρώνεται μόνο στις αποστάσεις, ενώ οι άλλοι δύο στον υπολογισμό των πυκνοτήτων με πολύ πιο έντονους υπολογισμούς. Συμπεραίνουμε λοιπόν, ότι οι διάρκειες εκτέλεσης ικανοποιούν τις ανωτέρω πολυπλοκότητες και μάλιστα οι δύο density-based μειώνουν κατά πολύ την αναμενόμενη διαφορά από τον K-means. Όσον αφορά τους δύο τελευταίους, βλέπουμε ότι ο CFSFDP υλοποιεί το clustering ταχύτερα, κάτα 1,5 περίπου δευτερόλεπτα, διαφορά που μπορεί να μη δείχνει τόσο σημαντική σε αυτό τον όγκο δεδομένων, αλλά σε μεγαλύτερο να παίζει πιο καθοριστικό ρόλο.

Κεφάλαιο 6

Αποτίμηση Επίδοσης Αλγορίθμων για Μεγάλα Δεδομένα

Όταν ερευνάται ο τομέας του big data μαζί με το Spark, ένα απαραίτητο προαπαιτούμενο είναι οι αλγόριθμοι που αναπτύσσονται να επεξεργαστούν και δεδομένα μεγάλου όγκου. Εκτός από την αξιολόγηση των αλγορίθμων για dataset τυχαίων, και μη, σημείων στους άξονες, που έγινε στο Κεφάλαιο 5 , εδώ αποτιμούνται και για δεδομένα συγκεντρωμένα από τον πραγματικό κόσμο. Στόχος αυτού του κεφαλαίου είναι ο έλεγχος της επίδοσης ενός προσαρμοσμένου σε πραγματικά δεδομένα CFSFDP και η σύγκριση του με τον αντίστοιχα προσαρμοσμένο K-means.

6.1 Το dataset

Το σύνολα δεδομένων στα οποία εφαρμόστηκαν οι αλγόριθμοι του Κεφαλαίου 4 συγκεντρώθηκαν από τον ιστότοπο [36] που περιέχει εκατομύρια ανοιχτά δεδομένα σε πάρα πολλούς τομείς για όλες τις χώρες της Ευρωπαικής Ένωσης. Προτιμήθηκε να μην είναι ένα μόνο αρχείο αλλά τρία, με κλιμακούμενη απόκλιση του όγκου των δεδομένων για να ελεγχθεί και να ερμηνευτεί η επίδοση των αλγορίθμων για διαφορετικές περιπτώσεις. Τα αρχεία αυτά χωρίζονται ως εξής:

1. Δεδομένα μικρού μεγέθους με 1054 μετρήσεις,
2. Δεδομένα μεσαίου μεγέθους με 2109 μετρήσεις,
3. Δεδομένα μεγάλου μεγέθους με 8428 μετρήσεις.

Ειδικότερα, το dataset αφορά μια συγκεκριμένη πόλη της βόρειας Ευρώπης στην οποία έγιναν πολλές μετρήσεις -για κάθε μέρα ενός ημερολογιακού έτους- κάποιων

μετεωρολογικών στοιχείων όπως η θερμοκρασία, η υγρασία και η πίεση(Εικόνα 6.1.1). Επίσης περιλαμβάνει τα αιωρούμενα σωματίδια(**P**articulate **M**atters) που χωρίζονται σε τρείς κατηγορίες ανάλογα με τη διάμετρό τους (pm1,pm2.5 και pm10). Τα σωματίδια αυτά θεωρούνται ρύποι λόγω της επικινδυνότητάς τους για την ανθρώπινη υγεία, και αν εισπνέονται σε μεγάλες ποσότητες μπορεί να προκαλέσουν σειρά προβλήματων επηρεάζοντας το κυκλοφοριακό, το καρδιαγγειακό, το dna, την εγκυμοσύνη και πολλά ακόμη [37]. Λόγω της ραγδαίας αύξησης των μετρούμενων ποσοτήτων των σωματιδίων στην ατμόσφαιρα των μεγάλων πόλεων έχει ξεκινήσει κύκλος ερευνών και συζητήσεων για την αντιμετώπιση του φαινομένου, πράγμα που δεν επιδιώκεται στην παρούσα διπλωματική.

Αυτό στο οποίο εστιάζει η εργασία είναι η ομαδοποίηση των τιμών του pm1 με κάποιο από τα άλλα μετεωρολογικά στοιχεία, ώστε να γίνει προσπάθεια αιτιολόγησης των υψηλών ποσοτήτων του κάποιες συγκεκριμένες περιόδους. Ο λόγος επιλογής του pm1 έχει να κάνει με το μεγάλο χρόνο παραμονής του στην ατμόσφαιρα και της μορφής του, πχ αιθαλομίχλη,άνθρακας κλπ. [37][38]

time	temperature	humidity	pressure	pm1	pm25	pm10
2017-01-01T07:00:00	0	50	102301	106	122	189
2017-01-01T15:00:00	4	41	101857	83	92	142
2017-01-01T23:00:00	0	51	101654	99	112	174
2017-01-02T08:00:00	3	47	101455	33	32	60
2017-01-02T16:00:00	3	59	101451	41	43	79
2017-01-03T00:00:00	3	57	101820	39	41	75
2017-01-03T08:00:00	3	50	101885	34	33	60
2017-01-03T16:00:00	3	62	101570	30	27	49
2017-01-04T00:00:00	1	63	100746	6	0	5
2017-01-04T08:00:00	3	63	100077	7	0	7
2017-01-04T16:00:00	3	60	99775	9	2	11
2017-01-05T00:00:00	1	55	100391	6	0	6
2017-01-05T08:00:00	-1	55	101374	8	1	10
2017-01-05T16:00:00	-2	44	102354	28	25	46

Εικόνα 6.1.1: Μικρό δείγμα των δεδομένων.

6.2 Προεπεξεργασία δεδομένων

Για το σύνολο δεδομένων που αναλύθηκε στην προηγούμενη ενότητα, η μεθοδολογία του CFSFDP που δόθηκε αναλυτικά στην Ενότητα 4.4 παραμένει ίδια. Το ίδιο ισχύει και για τον K-means(Ενότητα 4.2) με τον οποίο θα γίνει σύγκριση. Είναι απαραίτητο όμως να προστεθεί ένα βήμα στην αρχή το οποίο θα φιλτράρει τα δεδομένα(Εικόνα 6.1) από το αρχείο εισόδου και θα τα προετοιμάζει ώστε να εφαρμοστεί ο αλγόριθμος ομαδοποίησης με τις ίδιες εργασίες που ακολουθήθηκαν για τυχαία σημεία στους άξονες. Αυτή τη προσαρμογή των αλγορίθμων στην αρχή τους ας την ονομάσουμε Βήμα 0. Πιο συγκεκριμένα:

Βήμα 0.

Διαβάζεται το αρχείο εισόδου γραμμή προς γραμμή και χωρίζονται τα πεδία των διαφορετικών χαρακτηριστικών. Διαχωρίζονται από το πρώτο πεδίο η ώρα με την ημερο-

μηνία και προστίθεται το πεδίο της μέρας του χρόνου (1 ως 365). Επίσης οι τιμές του πεδίου “pressure” διαιρούνται με το 100 και μονάδα μέτρησης γίνεται το kilopascal. Στη συνέχεια, τοποθετείται ένα αναγνωριστικό id για κάθε σημείο μαζί με τα υπόλοιπα πεδία όπως “humidity”, “temperature” κλπ. Ακόμη, κάθε σημείο έχει “main data” που είναι οι δύο ιδιότητες στις οποίες θέλουμε να επικεντρωθεί το clustering. Τέλος, όλα αυτά τα χαρακτηριστικά κάθε σημείου, θα ενωθούν με τα υπόλοιπα (“cluster”, “local density” κλπ) στο Βήμα 1.

6.3 Ανάλυση και αξιολόγηση αποτελεσμάτων

Μετά την περιγραφή των δεδομένων και των χαρακτηριστικών τους, στην ενότητα αυτή παρουσιάζονται τα clusters που παρήγαγε ο CFSFDP και γίνεται η σύγκριση με τα αντίστοιχα του K-means. Τα πεδία που αποφασίστηκε να είναι τα “main data” και για τα τρία σύνολα δεδομένων είναι η ***θερμοκρασία*** και το ***pm1***. Ο λόγος αυτής της επιλογής δόθηκε αναλυτικά στην ενότητα 6.1.

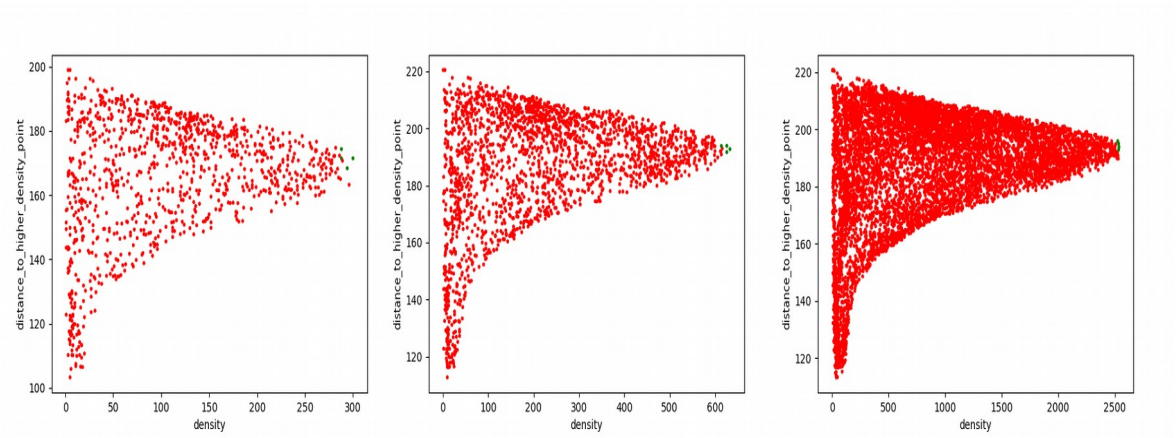
Ο αριθμός των clusters που προτιμήθηκε σε όλες τις υλοποιήσεις είναι τέσσερα, για να υπολογιστεί η ομοιότητα των αποτελεσμάτων με μια τυχαία ομαδοποίηση με προκαθορισμένο τρόπο, και πιο συγκεκριμένα με βάση τις τέσσερις εποχές του χρόνου. Αυτό θα γίνει με το μέτρο ομοιότητας για δημιουργημένα clusters, το Adjusted Rand Index. Περισσότερα για αυτή την αποτίμηση δίνονται στην Ενότητα 6.3.3.

Συγκεντρωτικά, η αποτίμηση της επίδοσης των αλγορίθμων χωρίζεται σε τρία μέρη:

1. Ποιότητα της ομαδοποίησης,
2. Χρόνος εκτέλεσης,
3. Μέτρηση της επίδοσης με βάση το Adjusted Rand Index.

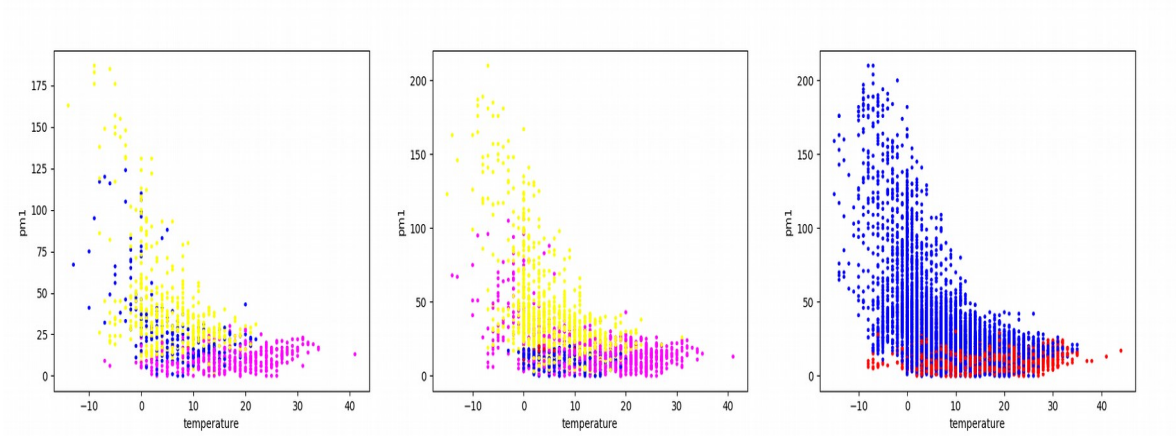
6.3.1 Αποτίμηση των παραγόμενων clusters

Ξεκινάμε,παρουσιάζοντας τα αποτελέσματα της μεθόδου Density Peaks και για τα τρία datasets. Στην Εικόνα 6.3.1 απεικονίζεται το γράφημα ανάθεσης κέντρων με βάση το γινόμενο γ. Είναι ξεκάθαρο πως η επιλογή των σημείων με πράσινο χρώμα ως κέντρα είναι σωστή. Οι λόγοι αυτής της επιλογής εξηγήθηκαν αναλυτικά και στην Ενότητα 5.1.1.



Εικόνα 6.3.1: Τα γραφήματα απόφασης του CFSFDP για τα 3 σύνολα δεδομένων.

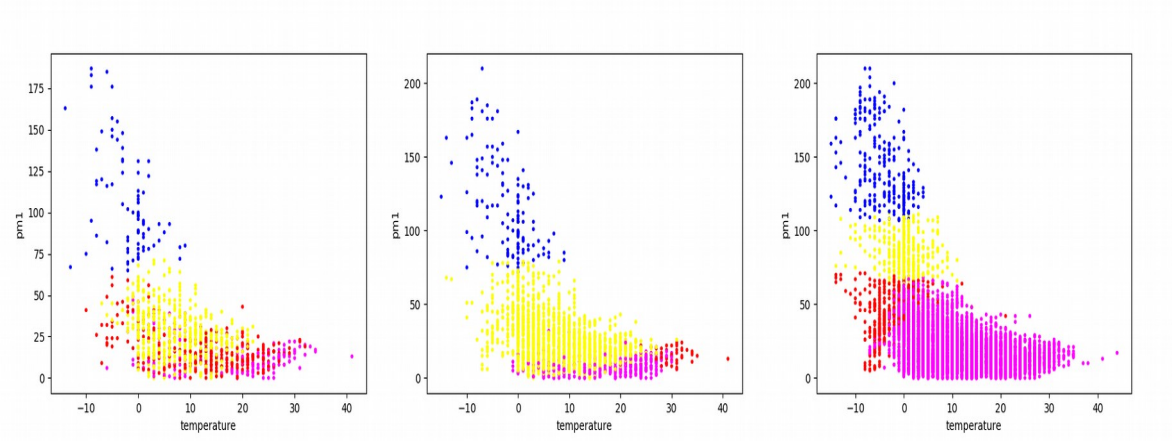
Στις γραφικές παραστάσεις της επόμενης Εικόνας (6.3.2), παρουσιάζονται τα clusters που δημιουργεί ο CFSFDP για τα τρία datasets. Στα δύο πρώτα σχήματα παρατηρούμε τρία ξεκάθαρα clusters, ενώ το τέταρτο -αυτό με το κόκκινο χρώμα- έχει πολύ λίγα σημεία και χάνεται στην απεικόνιση ανάμεσα στα υπόλοιπα. Στο τρίτο σχήμα αντιθέτως "χάνονται" δύο clusters στην απεικόνιση λόγω του υπερβολικά τεράστιου όγκου των στοιχείων.



Εικόνα 6.3.2: Τα clusters του CFSFDP για τα 3 σύνολα δεδομένων.

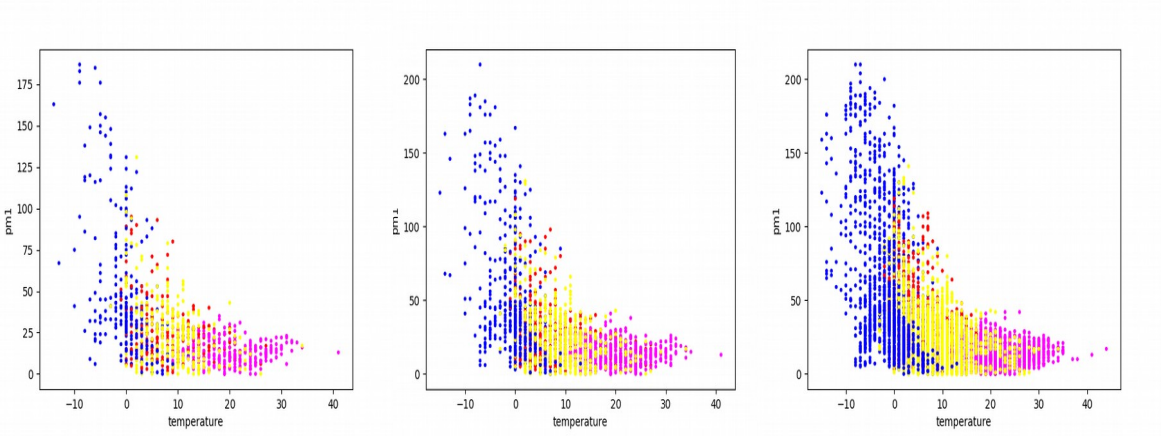
Στη συνέχεια δίνονται τα γραφήματα των συστάδων που προέκυψαν από τον αλγόριθμο του K-means (Εικόνα 6.3.3). Βλέπουμε πως όσο αυξάνεται το σύνολο δεδομένων τόσο πιο καθαρά φαίνεται ο διαχωρισμός των σημείων σε κομμάτια, γεγονός που οφείλεται στην νοοτροπία της μεθόδου να ελέγχει μόνο τις αποστάσεις.

Μπορεί διαισθητικά να φαίνεται καλύτερο στη γραφική παράσταση των αξόνων, επειδή οι ομάδες δεν επικαλύπτονται όπως τα clusters της Εικόνας 6.3.2, αλλά στο τρέχον σενάριο με το πολυσύνθετο σύνολο δεδομένων και τα χιλιάδες πολύ κοντινά στοιχεία, ο υπολογισμός των πυκνοτήτων, η εύρεση των Density Peaks και η δημιουργία ομάδων γύρω από αυτά παράγει αποτελέσματα πιο χρήσιμα για να διερευνηθούν και να εξηγηθούν οι εμφανίσεις του σωματιδίου pm1.



Εικόνα 6.3.3: Τα clusters του K-means για τα 3 σύνολα δεδομένων.

Τέλος, παραθέτονται τα clusters(Εικόνα 6.3.4) τα οποία παρήχθησαν προκαθορισμένα με βάση τις τέσσερις εποχές του χρόνου, όπως αναφέρθηκε στην αρχή του τρέχοντος κεφαλαίου και χρησιμοποιούνται ως μέτρο ομοιότητας στην Ενότητα 6.3.3.



Εικόνα 6.3.4: Τα clusters των 4 εποχών του χρόνου για τα 3 σύνολα δεδομένων.

6.3.2 Διάρκεια εκτελέσεων

Μετά την ερμηνεία των αποτελεσμάτων που έδωσαν οι δύο αλγόριθμοι παρουσιάζονται και οι χρόνοι εκτέλεσής τους, έτσι ώστε να ελεγχθούν οι επιδόσεις του CFSFDP ως προς την ταχύτητα και ως προς τη διάρκεια των αντίστοιχων εκτελέσεων του K-means.

	CFSFDP	*K-means*
1054 στοιχεία	2,31	1,25
2109 στοιχεία	10,90	6,34
8428 στοιχεία	205,83	93,84

Πίνακας 6.3.2: Χρόνοι εκτέλεσης των δύο αλγορίθμων για τα 3 datasets (σε λεπτά)

Παρατηρούμε πως η διαφορά στους χρόνους έχει μειωθεί σε σχέση με το σενάριο για σημεία στους άξονες και τις πολυπλοκότητες που περιγράφηκαν στην Ενότητα 5.2.2. Εκεί ο CFSFDP ήταν 3 φορές πιο χρονοβόρος από τον K-means, ενώ εδώ είναι περίπου 1.8 φορές πιο αργός στα δύο μικρότερα datasets και 2.2 φορές πιο αργός στο τεράστιο σύνολο των 8.5 χιλιάδων σημείων.

Ο λόγος της χρονικής βελτίωσης του CFSFDP οφείλεται στην υλοποίηση για μεγάλα σύνολα, καθώς υπάρχουν αρκετές πυκνές περιοχές που ευνοούν από τη μία αυτή τη μέθοδο και αντιθέτως δυσκολεύουν τον επαναληπτικό υπολογισμό αποστάσεων του K-means. Ακόμη, πρέπει να σημειωθεί πως η τεράστια απόκλιση των χρόνων των αλγορίθμων στην τρίτη υλοποίηση οφείλεται στη τεράστια διαφορά του όγκου του τρίτου dataset από τα πρώτα δύο.

6.3.3 Αξιολόγηση με βάση τον τυχαίο δείκτη ARI

Όταν επιτελείται ομαδοποίηση δεδομένων με διάφορες μεθόδους, είναι απαραίτητο να ελεγχθεί η επιτυχής δημιουργία clusters. Υπάρχουν πολλοί τρόποι και μετρικές για να γίνει αυτό και στην παρούσα διπλωματική επιλέχτηκε ένα μέτρο ομοιότητας, ο Adjusted Random Index, ο οποίος αποτελεί προέκταση του απλού Rand Index ρυθμισμένου για τυχαία ομοιότητα. Ο Rand Index υπολογίζει την ομοιότητα μεταξύ δύο clusterings παίρνοντας όλα τα ζευγάρια δειγμάτων και μετρώντας αυτά που έχουν ανατεθεί (labeling) σε ίδια ή διαφορετικά clusters. Ο Adjusted Rand Index είναι μια ρυθμισμένη έκδοση του πρώτου που συγκρίνει τα clustered στοιχεία μιας μεθόδου με αυτά ενός τυχαίου μοντέλου. Η τιμή του ARI είναι 0.0 για clusterings που δε μοιάζουν καθόλου,και 1.0 όταν τα clusterings ταυτίζονται τελείως. [39][40]

Το τυχαίο μοντέλο που επιλέχθηκε είναι η προκαθορισμένη ομαδοποίηση των στοιχείων με βάση τις τέσσερις εποχές του χρόνου και περιγράφηκε στην αρχή του κεφαλαίου και την Εικόνα 6.3.4. Βλέπουμε, ότι και στις έξι εκτελέσεις ο Adjusted Random Index είναι πολύ κοντά στο μηδέν (Πίνακας 6.3.3). Αυτό το γεγονός υποδηλώνει το επιτυχημένο labeling των στοιχείων χωρίς καθόλου λάθη, τόσο από τον CFSFDP όσο και από τον K-means.

	CFSFDP	*K-means*
1054 στοιχεία	0,0934	0,0640
2109 στοιχεία	0,0444	0,0283
8428 στοιχεία	0,0177	0,0476

Πίνακας 6.3.3: Adjusted Random Index των δύο αλγορίθμων για τα 3 datasets

6.3.4 Σχόλιο για το Spark cache

Σε συνέχεια όσων παρουσιάστηκαν στην Ενότητα 5.1.5 για τη σημαντικότητα της ανάκτησης των δεδομένων από τη μνήμη, αξίζει να τονιστεί η βελτιστοποίηση που παρέχει στην εφαρμογή η εντολή ***cache()***. Οι χρόνοι που δόθηκαν στην ενότητα 6.3.2 ακολουθούν τις εργασίες της ενότητας 4.4 οι οποίες χρησιμοποιούν αυτή τη δυνατότητα. Ενδεικτικά αναφέρονται οι διάρκειες εκτελέσεων για το μικρό dataset με τις 1054 μετρήσεις χωρίς τη μέθοδο ***cache()***:

- CFSFDP: χρόνος = 119.57 λεπτά.
- K-means: χρόνος = 3.45 λεπτά.

Ο λόγος που ο χρόνος εκτέλεσης του K-means δεν αυξάνεται τόσο πολύ όσο του DensityPeaks είναι ότι χρησιμοποιεί πολύ πιο "ανώδυνους" υπολογισμούς όπως έχει εξηγηθεί πολλάκις στην εργασία. Επίσης, η εκτέλεση του CFSFDP για τα δύο μεγαλύτερα dataset θα έπαιρνε ώρες ή και μέρες.

Για να γίνει ακόμη πιο κατανοητή πόσο χρονοβόρα είναι η επαναληπτική ανάκτηση των ίδιων δεδομένων παραθέτονται δύο στιγμιότυπα οθόνης από τη διεπαφή στο web του Spark (Εικόνα 6.3.5 και Εικόνα 6.3.6). Η διεπαφή αυτή είναι προσβάσιμη στον browser σε τοπικό κανάλι (:4040) και περιέχει χρήσιμες πληροφορίες για κάθε εφαρμογή που εκτελείται, όπως:

- Λίστα των διεργασιών και των σταδίων εκτέλεσης,
- Σύνοψη του μεγέθους της RDD και της χρήσης μνήμης,
- Πληροφορίες για τους τρέχοντες worker κόμβους.

Περισσότερες πληροφορίες παρέχονται στο [41]

Completed Stages (18)

Stage Id ▼	Description		Submitted	Duration
17	count at DensityPeaks.py:188	+details	2018/10/04 21:56:03	0.2 s
16	collect at DensityPeaks.py:185	+details	2018/10/04 21:56:03	86 ms
15	collect at DensityPeaks.py:184	+details	2018/10/04 21:56:03	82 ms
14	count at DensityPeaks.py:181	+details	2018/10/04 21:56:03	0.1 s
13	collect at DensityPeaks.py:135	+details	2018/10/04 21:56:02	99 ms
12	collect at DensityPeaks.py:164	+details	2018/10/04 21:56:02	86 ms
11	runJob at PythonRDD.scala:149	+details	2018/10/04 21:56:02	73 ms
10	sortBy at DensityPeaks.py:157	+details	2018/10/04 21:56:02	0.2 s
9	sortBy at DensityPeaks.py:157	+details	2018/10/04 21:56:02	97 ms
8	sortBy at DensityPeaks.py:157	+details	2018/10/04 21:55:13	49 s

Εικόνα 6.3.5: Διάρκεια λειτουργιών του Spark με χρήση cache.

Stages (18)

Description		Submitted	Duration
count at DensityPeaks.py:187	+details	2018/10/04 22:25:53	59 s
collect at DensityPeaks.py:184	+details	2018/10/04 22:24:50	1.0 min
collect at DensityPeaks.py:183	+details	2018/10/04 22:23:50	1.0 min
count at DensityPeaks.py:180	+details	2018/10/04 22:22:49	1.0 min
collect at DensityPeaks.py:134	+details	2018/10/04 22:21:46	1.0 min
collect at DensityPeaks.py:163	+details	2018/10/04 22:20:45	1.0 min
runJob at PythonRDD.scala:149	+details	2018/10/04 22:20:44	89 ms
sortBy at DensityPeaks.py:156	+details	2018/10/04 22:19:42	1.0 min
sortBy at DensityPeaks.py:156	+details	2018/10/04 22:18:38	1.1 min
sortBy at DensityPeaks.py:156	+details	2018/10/04 22:17:29	1.1 min

Εικόνα 6.3.6: Διάρκεια λειτουργιών του Spark χωρίς χρήση cache.

Παρατηρούμε ότι εκτός από το στάδιο με id=8 όπου η λειτουργία ***sortBy*** παίρνει 1 περίπου λεπτό, οι υπόλοιπες εντολές διαρκούν ελάχιστο χρόνο στην πρώτη περίπτωση, σε αντίθεση με τη δεύτερη που η διάρκεια είναι η ίδια επειδή τα δεδομένα ξαναφορτώνονται κάθε φορά για τους υπολογισμούς.

Αυτά τα αποτελέσματα μας οδηγούν στο συμπέρασμα πως το Spark μπορεί να παρέχει πλήθος μετασχηματισμών και ενεργειών πάνω στην RDD, αλλά αν δε χρησιμοποιηθούν κατάλληλα μπορεί να οδηγήσουν σε ανεπιθύμητα αποτελέσματα.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική Έρευνα

7.1 Συμπεράσματα

Στην παρούσα διπλωματική παρουσιάστηκε μια εκτεταμένη ανάλυση του κύριου συστατικού της εξόρυξης μεγάλων δεδομένων, της ομαδοποίησης. Επίσης, εξηγήθηκε ολόκληρη η διαδικασίας επίλυσης του προβλήματος(1.2) και του στόχου(1.3) που τέθηκε στην αρχή. Πρώτα διεξήχθη μια αναλυτική έρευνα πάνω στα μεγάλα δεδομένα, το clustering και το Spark. Στη συνέχεια περιγράφηκε η σημαντικότητα των density-based αλγορίθμων και η διαδικασία παραλληλισμού τους στο Spark. Δόθηκαν σχεδιαστικές λεπτομέρειες για κάθε μέθοδο(CFSFDP, DBSCAN, K-means) και εφαρμόστηκαν σε δύο διαφορετικά datasets.

Στο πρώτο σύνολο, που περιείχε τυχαία σημεία, έγινε η σύγκριση των τριών τόσο ως προς την εγκυρότητα των clusters που δημιουργούν όσο και ως προς τους χρόνους εκτέλεσης. Στο δεύτερο σύνολο, που περιελάμβανε χιλιάδες πραγματικά στοιχεία αναπτύχθηκε ένας προσαρμοσμένος CFSFDP και τα αποτελέσματά του συγκρίθηκαν με τα αντίστοιχα του K-means και αποτιμήθηκαν με τον Adjusted Rand Index. Και στις δύο περιπτώσεις, επιβεβαιώθηκε ότι οι αλγόριθμοι υλοποιήθηκαν σωστά και παραλληλίστηκαν κατάλληλα παράγοντας σωστά clusters και βελτιώνοντας κατά πολύ τις πολυπλοκότητές τους. Επιπλέον, τονίστηκαν οι πιο σημαντικές σχεδιαστικές λειτουργίες του Spark και προτάθηκαν τεχνικές για καλύτερη χρήση τους και βελτιστοποίηση της διάρκειας των εργασιών.

7.2 Μελλοντική Έρευνα

Υπάρχουν τεράστιες δυνατότητες για περαιτέρω έρευνα και ο λόγος είναι ότι τα μεγάλα δεδομένα του σήμερα δε γνωρίζουμε αν θεωρούνται τα μεγάλα δεδομένα του αύριο με τις ταχύτητες που δημιουργούνται από όλες τις πηγές.

Η πρώτη πρόταση η οποία θα επέκτεινε τη δουλειά της παρούσας διπλωματικής είναι η δημιουργία ενός πραγματικού cluster, πολλών υπολογιστών, διαμορφωμένου από το Spark για την υλοποίηση αλγορίθμων σε δεδομένα ακόμη μεγαλύτερου όγκου.

Ιδιαίτερο ενδιαφέρον παρουσιάζει επίσης η δημιουργία καινούργιων αλγορίθμων, κατάλληλων για ακόμη καλύτερη επεξεργασία, πρόβλεψη και ομαδοποίηση των διαφόρων τύπων δεδομένων.

Τέλος και πιο σημαντικό, είναι η υλοποίηση αλγορίθμων σε μη στατικά δεδομένα, δηλαδή σε ροές(streaming data). Τα δεδομένα που παράγονται σήμερα αλλάζουν συνεχώς και υπάρχουν τομείς που επιτάσσουν την ανάκτηση και ανάλυση των στοιχείων σε πραγματικό χρόνο. Το Spark παρέχει αυτή τη δυνατότητα με τη διεπαφή “Streaming”, η οποία δέχεται σαν είσοδο ροές δεδομένων, τις παραλληλίζει και εφαρμόζει συγκεκριμένο προγραμματιστικό σχεδιασμό.

Βιβλιογραφία

1. Lana Yeganova, Won Kim, Sun Kim, W.J. Wilbur, Retro: concept-based clustering of biomedical topical sets, Bioinformatics 30 (22) (2014) 3240–3248.
2. Chen Xu, Zhengchang Su, Identification of cell types from single-cell transcriptomes using a novel clustering method, Bioinformatics 37 (10) (2015) 2041–2256.
3. Y. Yan, Y. Qian, H. Sharif, D. Tipper, A survey on cyber security for smart grid communications, IEEE Commun. Surv. Tutor. 14 (4) (2012) 998–1010.
4. Portnoy, Leonid, Eleazar Eskin, Sal Stolfo, Intrusion detection with unlabeled data using clustering, in: Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001), 2001, pp. 5–8.
5. Jiwen Lu, Xiuzhuang Zhou, Yap-Peng Tan, Yuanyuan Shang, Jie Zhou, Neighborhood repulsed metric learning for kinship verification, IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) 36 (2) (2014) 331–345.
6. Jiwen Lu, Yap-Peng Tan, Gang Wang, Discriminative multimanifold analysis for face recognition from a single training sample per person, IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) 35 (1) (2013) 39–51.
7. Jones, M. Kristen, M. Lacy, Measuring the clustering around normal and dust obscured quasars at 2 in the Spitzer extragalactic representative volume survey (SERVS), in: American Astronomical Society Meeting Abstracts, vol. 223, no. 223, 2014.
8. Chakrabarti, Soumen, Data mining for hypertext: a tutorial survey, in: ACM SIGKDD Explorations Newsletter, vol. 1, no. 2, 2000, pp. 1–11.
9. Maw-Shang Chang, Li-Hsuan Chen, Ling-Ju Hung, Peter Rossmanith, Guan-Han Wu, Exact algorithms for problems related to the densest k-set problem, Inf. Process. Lett. 114 (9) (2014) 510–513.
10. Cheng-Fa Tsai and Chun-Yi Sung "DBSCALE: An Efficient Density-Based Clustering Algorithm for Data Mining in Large Databases", 2010 Second Pacific-Asia Conference on Circuits, Communications and System (PACCS-2010), pp. 98 – 101, 2010.
11. Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: Communications of the ACM 51.1 (2008), pp. 107–113.
12. Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise."
13. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. Springer series in statistics New York, 2001.

14. Stuart Lloyd. “Least squares quantization in PCM”. In: IEEE transactions on information theory 28.2 (1982), pp. 129–137.

15. Satu Elisa Schaeffer. “Graph clustering”. In: Computer science review 1.1 (2007),pp. 27–64.

16. A Comparative Analysis of DBSCAN, K-means, and Quadratic Variation Algorithms for Automatic Identification of Swallows from Swallowing Accelerometry Signals Joshua M. Dudik, Atsuko Kurosu, James L Coyle, Ervin Sejdić.

17. Matei Zaharia et al. “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing”. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association. 2012, pp. 2–2.

18. Matei Zaharia et al. “Spark: Cluster computing with working sets.” In: HotCloud 10.10-10 (2010), p. 95.

19. Spark Transformations. Url: https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

20. Spark Actions. Url: https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions

21. “The Importance of 'Big Data': A Definition”.Mark A. Beyer and Douglas Laney. Gartner, 2012.

22. “Data science for business”.Foster Provost and Tom Fawcett. O’Relly, 2013.

23. “Taming the big data tidal wave”. Bill Franks.Wiley, 2012.

24. “Clustering by fast search and find of density peaks”. Alex Rodriguez and Alessandro Laio,Science 344, page 1492,2014.

25. C. Sugar, G. James. “Finding the number of clusters in a data set: An information theoretic approach,” Journal of the American Statistical Association 98 (2003), pp. 750–763

26. “Adaptive fuzzy clustering by fast search and find of density peaks”. Rongfang Bie, Rashid Mehmood, Shanshan Ruan, Yunchuan Sun, Hussian Dawood, 2016.

27. “Clustering by fast search and find of density peaks via heat diffusion”. Rashid Mehmood, Guangzhi Zhang, Rongfang Bie, Hassan Dawood, Haseeb Ahmad, 2016.

28. “Extended fast search clustering algorithm: widely density clusters, no Density Peaks”. Zhang WenKai and Li Jing.

29. “Density-based algorithm in MapReduce”. Pang Lin, Liu Fang-ai, 9th International Symposium on Computational Intelligence and Design, 2016.

30. “Improved density peak clustering for large datasets”. Vincent Courjault-Radé, Ludovic D’Estampes, Stéphane Puechmorel. 2016.

31. “Clustering in the Face of Fast Changing Streams”. Liudmila Ulanova, Nurjahan Begum, Mohammad Shokoohi-Yekta, Eamonn Keogh.

32. “A Parallel DBSCAN Algorithm Based on Spark” Guangchun Luo, Xiaoyu Luo, Thomas Fairley Gooch, Ling Tian, Ke Qin.

33. Generate Gaussian Blobs : http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

34. The Complexity of the K-means Method∗ Tim Roughgarden and Joshua R. Wang 5.2.2

35. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu 5.2.2

36. Dataset Retrieved From: http://data.europa.eu/euodp/en/data/

37. Particulate Matters Information: http://www.mie.uth.gr/ekp_yliko/3_particulates.pdf

38. Particulate Matters Information: http://www.physics.ntua.gr/~papayannis/Articles%20for%20tamex/PMs-TSI.pdf

39. Adjusted Random Index: https://www.researchgate.net/profile/Ka_Yee_Yeung/publication/239537124_Details_of_the_Adjusted_Rand_index_and_Clustering_algorithms_Supplement_to_the_paper_An_empirical_study_on_Principal_Component_Analysis_for_clustering_gene_expression_data_to_appear_in_Bioinformatics/links/543b62a30cf24a6ddb976fd3/Details-of-the-Adjusted-Rand-index-and-Clustering-algorithms-Supplement-to-the-paper-An-empirical-study-on-Principal-Component-Analysis-for-clustering-gene-expression-data-to-appear-in-Bioinformatics.pdf

40. Adjusted Random Score in Python: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

41. Spark Localhost Control: https://spark.apache.org/docs/latest/monitoring.html .

42. Technological ecosystem of Hadoop: https://hadoopecosystemtable.github.io/