ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

Μεταπτυχιακό Πρόγραμμα στην Εφαρμοσμένη Στατιστική

"Εξόρυξη Δεδομένων σε Πολυδιάστατες Χρονοσειρές"

Χρήστος Δαγκλής

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην Εφαρμοσμένη Στατιστική.

Πειραιάς 2020

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη ΓΣΕΣ του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμό …… συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική

Τα μέλη της Επιτροπής ήταν:

- Αναπληρωτής Καθηγητής Νικόλαος Πελέκης (Επιβλέπων)
- Καθηγητής Ιωάννης Θεοδωρίδης
- Αναπληρωτής Καθηγητής Ελευθέριος Κοφίδης

Η έγκριση της Διπλωματικής Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

UNIVERSITY OF PIRAEUS

School of Finance and Statistics

Department of Statistics and Insurance Science

Postgraduate Program in Applied Statistics

"Mining Multi-Dimensional timeseries"

by

Christos Daglis

MSc Dissertation

Submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of the Master of Science in Applied Statistics.

Piraeus 2020

Στην οικογένεια μου και στους φίλους μου για την αμέριστη στήριξη τους.

<u>Ευχαριστίες</u>

Ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή μου, κύριο Νικόλαο Πελέκη, για τη στήριξη, την άψογη συνεργασία και την καθοδήγηση του στην περάτωση της διπλωματικής μου εργασίας.

Περίληψη

Στην εποχή που ζούμε, καθημερινά αναπαράγονται, ενημερώνονται και αποθηκεύονται δεδομένα χρονοσειρών σε διάφορες βάσεις δεδομένων. Τα δεδομένα χρονοσειρών που συλλέγονται, πέρα από το γεγονός ότι αυξάνονται μέρα με τη μέρα, τείνουν να αποκτούν πολυδιάστατη μορφή. Με αυτό τον τρόπο δημιουργούνται πολυδιάστατες χρονοσειρές που απαιτούν νέους και καλύτερους τρόπους εξόρυξης, ευρετηρίασης και αναζήτησης. Σε αυτήν την εργασία περιγράφουμε υπάρχοντες τεχνικές της οικογένειας iSAX που χρησιμοποιούνται για εργασίες εξόρυξης μονοδιάστατων χρονοσειρών. Για εργασίες εξόρυξης πολυδιάστατων χρονοσειρών περιγράφουμε και χρησιμοποιούμε μία υπάρχουσα μέθοδο, που αποτελεί επέκταση των προηγούμενων τεχνικών και ονομάζεται hyperSAX. Για να δείξουμε τη χρησιμότητά της, η τεχνική hyperSAX εφαρμόζεται για εργασίες ευρετηρίασης και αναζήτησης ομοιότητας σε δισδιάστατες χρονοσειρές και μετριέται ο χρόνος δημιουργίας του ευρετηρίου.

Abstract

In our day, time series data is reproduced, updated and stored daily in various databases. The time series data collected, in addition to the fact that they are increasing day by day, tend to take on a multidimensional form. In this way, multidimensional time series are created that require new and better ways of data mining, indexing and searching. In this work we describe existing techniques of the iSAX family, used for one-dimensional time series data mining operations. For multidimensional time series data mining operations, we describe and use an existing method, which is an extension of the previous techniques, called hyperSAX. To demonstrate its usefulness, the hyperSAX technique is applied to two-dimensional time series for indexing and similarity search operations and the time it takes to build the index is measured.

Περιεχόμενα

Εισαγωγή

Στην εποχή που ζούμε, καθημερινά αναπαράγονται, ενημερώνονται και αποθηκεύονται δεδομένα σε διάφορες βάσεις δεδομένων (όπως για παράδειγμα δεδομένα στην πολύ γνωστή διαδικτυακή πλατφόρμα Facebook αλλά και online πλατφόρμες που αποθηκεύουν δεδομένα χρηματιστηρίου). Πολλά από αυτά τα δεδομένα αποτελούν χρονοσειρές, δηλαδή το σύνολο δεδομένων μίας μεταβλητής τα οποία συλλέγονται ανά συγκεκριμένα χρονικά διαστήματα (ωριαία, ημερήσια, μηνιαία, ετήσια κλπ.). Τα δεδομένα χρονοσειρών που συλλέγονται, πέρα από το γεγονός ότι αυξάνονται μέρα με τη μέρα, τείνουν να αποκτούν πολυδιάστατη μορφή. Αυτό σημαίνει ότι με την πάροδο του χρόνου, σε κάθε χρονικό σημείο συλλέγονται μετρήσεις για τη πολυδιάστατη χρονοσειρά που μας ενδιαφέρει, όχι για ένα χαρακτηριστικό της, αλλά για πολλά. Με αυτό τον τρόπο δημιουργούνται πολυδιάστατες χρονοσειρές που απαιτούν νέους και καλύτερους τρόπους ευρετηρίασης και αναζήτησης.

Ένα **ευρετήριο** (index) βάσης δεδομένων είναι μια δομή δεδομένων (μια μορφή οργάνωσης, διαχείρισης και αποθήκευσης δεδομένων) που βελτιώνει την ταχύτητα των εργασιών ανάκτησης των αρχικών δεδομένων χωρίς να απαιτείται αναζήτηση σε κάθε σειρά. Τα ευρετήρια μπορούν να δημιουργηθούν χρησιμοποιώντας μία ή περισσότερες στήλες ενός πίνακα βάσης δεδομένων. Η **αναζήτηση ομοιότητας,** είναι ένας τρόπος αναζήτησης ενός χαρακτηριστικού που μας ενδιαφέρει σε μία βάση δεδομένων. Στο 1ο Κεφάλαιο παρουσιάζεται μια γενική προσέγγιση εξόρυξης δεδομένων καθώς και ορισμοί τους.

Τις τελευταίες δεκαετίες, υπάρχει μεγάλο ενδιαφέρον για το πρόβλημα της αναζήτησης ομοιότητας σε βάσεις δεδομένων χρονοσειρών. Η αναζήτηση ομοιότητας είναι χρήσιμη ως ένα εργαλείο για την εξερεύνηση τέτοιων βάσεων δεδομένων χρονοσειρών, όπως επίσης είναι σημαντική σε πολλές εφαρμογές όπως η ομαδοποίηση (clustering) [1], η ταξινόμηση (classification) [2] και εξόρυξη κανόνων συσχέτισης (mining of association rules) [3].

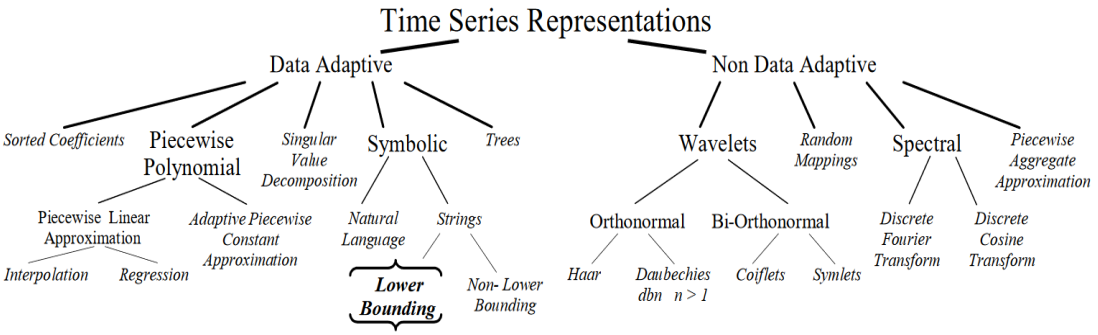
Σε όλες τις διαδικασίες εξόρυξης μονοδιάστατων χρονοσειρών μεγάλου μεγέθους, το πιο σημαντικό πρόβλημα που αντιμετωπίζεται είναι ο πολύ μεγάλος χρόνος δημιουργίας του ευρετηρίου (index). Επιπλέον, αφού δημιουργηθεί το ευρετήριο, οι χρόνοι ανάκτησης πρέπει να μειωθούν στο ελάχιστο, καθώς τόσο το ευρετήριο όσο και τέτοια μεγάλου μεγέθους δεδομένα να καταλαμβάνουν σημαντικό μέρος του δίσκου. Οι μέθοδοι της οικογένειας iSAX [4]–[8] που περιγράφονται στο 2ο Κεφάλαιο προσπαθούν να επιλύσουν αυτά τα δύο προβλήματα.

Στο 3ο Κεφάλαιο παρουσιάζεται μια μέθοδος που ονομάζεται hyperSAX [9], η οποία μπορεί να χρησιμοποιηθεί για εργασίες εξόρυξης, ευρετηρίασης και αναζήτησης ομοιότητας πολυδιάστατων χρονοσειρών.

Στο 4ο Κεφάλαιο η μέθοδος hyperSAX εφαρμόζεται σε πολυδιάστατες χρονοσειρές (2-διαστάσεων), όπου παρουσιάζονται τα σχετικά πειράματα και στο 5ο Κεφάλαιο αναφέρονται τα συμπεράσματα.

Κεφάλαιο 1 – Ιστορικό υπόβαθρο / Ορισμοί

Πολλές αναπαραστάσεις χρονοσειρών έχουν προταθεί για την εξόρυξη δεδομένων. Αναπαράσταση χρονοσειράς είναι ένα μοντέλο αποθήκευσης και παρουσίασης της αρχικής χρονοσειράς με μειωμένο μήκος, διατηρώντας τα χαρακτηριστικά της. Μήκος χρονοσειράς είναι το σύνολο των χρονικών στιγμών για τα οποία συλλέχτηκαν δεδομένα (παράδειγμα, για μηνιαία δεδομένα ενός έτους το μήκος της χρονοσειράς είναι 12). Το Σχήμα 1 απεικονίζει μια ιεραρχία διάφορων αναπαραστάσεων χρονοσειρών στη βιβλιογραφία [10]–[15]. Μερικές από αυτές τις αναπαραστάσεις μετατρέπουν τις χρονοσειρές σε συμβολικούς χαρακτήρες. Οι τεχνικές της οικογένειας iSAX είναι από τις ελάχιστες συμβολικές αναπαραστάσεις που επιτρέπουν τη μέτρηση ομοιότητας δύο χρονοσειρών στο συμβολικό χώρο με εγγύηση ότι αυτή η απόσταση είναι κατώτερη από την πραγματική απόστασή τους. (Η ιδιότητα του κάτω ορίου αναλύεται παρακάτω)



Σχήμα 1: Μια ιεραρχία διάφορων αναπαραστάσεων χρονοσειρών στη βιβλιογραφία [5].

Η ακόλουθη λίστα συνοψίζει τους τομείς που έχουν δει το μεγαλύτερο μέρος του ερευνητικού ενδιαφέροντος για την εξόρυξη δεδομένων χρονοσειρών.

- **Ευρετηρίαση (indexing):** Δεδομένου ενός ερωτήματος αναζήτησης χρονοσειρών (time series query) και κάποιου μέτρου ομοιότητας /ανομοιότητας, βρείτε τις πιο παρόμοιες χρονοσειρές στη βάση δεδομένων (χρησιμοποιώντας το ευρετήριο) [6].
- **Ομαδοποίηση (clustering):** Βρείτε φυσικές ομαδοποιήσεις των χρονοσειρών στη βάση δεδομένων με κάποιο μέτρο ομοιότητας / ανομοιότητας [1].
- **Ταξινόμηση (classification):** Λαμβάνοντας υπόψη μια χρονοσειρά, αντιστοιχίστε την σε μία από τις δύο ή περισσότερες προκαθορισμένες κατηγορίες [2].
- **Περίληψη (Summarization):** Λαμβάνοντας υπόψη μια χρονοσειρά που περιέχει *n* σημεία, όπου το *n* είναι ένας εξαιρετικά μεγάλος αριθμός, δημιουργήστε μια προσέγγιση της χρονοσειράς που διατηρεί τα βασικά χαρακτηριστικά της, αλλά ταιριάζει σε μία γραφική απεικόνιση [16].
- **Ανίχνευση ανωμαλιών (anomaly detection):** Λαμβάνοντας υπόψη μια χρονοσειρά, και κάποιο μοντέλο «κανονικής» συμπεριφοράς, βρείτε όλα τα τμήματα της χρονοσειράς που περιέχουν ανωμαλίες ή απροσδόκητη συμπεριφορά [17].

Γενική Προσέγγιση Εξόρυξης Δεδομένων:
Τα σύνολα δεδομένων χρονοσειρών κατά την εξόρυξη δεδομένων συνήθως δεν χωρούν στην κύρια μνήμη. Η βασική προσέγγιση ενός απλού γενικού πλαισίου για την εξόρυξη δεδομένων χρονοσειρών περιγράφεται στον Πίνακα 1 [10].

1.	Δημιουργία μιας προσέγγισης των δεδομένων, τα οποία θα χωρέσουν στην κύρια μνήμη, αλλά διατηρεί τα βασικά χαρακτηριστικά τους
2.	Λύση του προβλήματος στην κύρια μνήμη
3.	Δημιουργία (ελπίζουμε πολύ λίγων) προσβάσεων στα αρχικά δεδομένα στο δίσκο για επιβεβαίωση της λύσης που αποκτήθηκε στο Βήμα 2 ή τροποποίηση της λύσης έτσι ώστε να συμφωνεί με τη λύση που θα είχαμε λάβει στα αρχικά δεδομένα.

Πίνακας 1: Μια γενική προσέγγιση εξόρυξης δεδομένων χρονοσειρών [5].

Θα πρέπει να είναι σαφές ότι η χρησιμότητα αυτού του πλαισίου εξαρτάται σε μεγάλο βαθμό από την ποιότητα της προσέγγισης που δημιουργήθηκε στο Βήμα 1 (index). Εάν η προσέγγιση είναι πολύ πιστή στα αρχικά δεδομένα, τότε η λύση που λαμβάνεται στην κύρια μνήμη είναι πιθανό να είναι η ίδια, ή πολύ κοντά στη λύση, που θα είχαμε πάρει στα αρχικά δεδομένα. Οι προσπελάσεις δίσκου που έγιναν στο Βήμα 3 για επιβεβαίωση ή τροποποίηση ελαφρώς της λύσης, θα είναι άνευ σημασίας σε σύγκριση με τον αριθμό των απαιτούμενων προσβάσεων δίσκου εάν εργαζόμασταν στα αρχικά δεδομένα.

Υπάρχουν πολλές ιδιαίτερα επιθυμητές ιδιότητες για οποιοδήποτε τεχνική ευρετηρίου [10].

1. Πρέπει να είναι αρκετά πιο γρήγορο από τη διαδοχική σάρωση.
2. Η μέθοδος πρέπει να απαιτεί μικρό χώρο στο δίσκο.
3. Η μέθοδος πρέπει να μπορεί να διαχειρίζεται ερωτήματα διαφόρων μηκών.
4. Δε θα πρέπει να εμφανίζονται ψευδείς απορρίψεις.
5. Πρέπει να είναι δυνατή η δημιουργία του ευρετηρίου σε εύλογο χρόνο.

Σε αυτή την εργασία επικεντρωνόμαστε σε συμβολικές αναπαραστάσεις, δηλαδή σε μοντέλα μετατροπής των χρονοσειρών σε σύμβολα.

Ιδιότητα Κάτω ορίων: Η δυνατότητα ορισμού ενός μέτρου απόστασης (στα δεδομένα τα οποία έχουμε μετατρέψει ανάλογα με τη μέθοδο) το οποίο εγγυείται ότι είναι μικρότερο ή ίσο με την πραγματική απόσταση που μετριέται στα πρωτογενή (αρχικά) δεδομένα αποτελεί την ιδιότητα κάτω ορίων. Η βασική παρατήρηση είναι ότι μέσω της μεταβατικότητας μεταφέρεται αυτή η ιδιότητα από την πρώτη μέθοδο και στις υπόλοιπες μεθόδους. Με αυτόν τον τρόπο μπορούμε να συγκρίνουμε την απόσταση δύο χρονοσειρών (αφού έχουν μετατραπεί ανάλογα με τη μέθοδο) και αυτή η απόσταση θα είναι εγγυημένα χαμηλότερη ή ίση από την πραγματική ευκλείδεια απόσταση στον αρχικό χώρο. Η ιδιότητα του κατώτερου ορίου μας επιτρέπει να εκτελούμε συγκεκριμένους αλγόριθμους εξόρυξης δεδομένων στη συμβολική αναπαράσταση, παράγοντας ταυτόσημα αποτελέσματα με τους αλγόριθμους που λειτουργούν στα αρχικά δεδομένα.

Μείωση μήκους χρονοσειράς: Τα περισσότερα σύνολα χρονοσειρών έχουν πάρα πολύ μεγάλο μήκος. Αυτή η παρατήρηση αποκτά μεγάλη σημασία, διότι η απόδοση των περισσότερων αλγόριθμων εξόρυξης και ευρετηρίασης δεδομένων χρονοσειρών υποβαθμίζεται με εκθετικό τρόπο καθώς αυξάνεται το μήκος της χρονοσειράς. Για παράδειγμα, για μεγάλο αριθμό μήκους χρονοσειράς, οι δομές ευρετηρίου τείνουν να υποβαθμίζονται σε δομές διαδοχικής σάρωσης (sequential scanning) [18]. Ωστόσο, οι μέθοδοι που παρουσιάζονται σε αυτή την εργασία επιτρέπουν τη μείωση του μήκους των πρωτογενών δεδομένων χρονοσειρών. Η μείωση του μήκους της χρονοσειράς παρουσιάζεται στην πρώτη μέθοδο και μπορούμε στη συνέχεια, μέσω της μεταβατικότητας, να χρησιμοποιήσουμε αυτή την ισχύ μείωσης μήκους της χρονοσειράς και στις υπόλοιπες συμβολικές αναπαραστάσεις.

Αναζήτηση ομοιότητας - Είδη Ερωτημάτων (Queries) :
Στην αναζήτηση ομοιότητας υπάρχουν δύο σημαντικά είδη ερωτημάτων (queries) που θα θέλαμε να αναπτύξουμε: *τα ερωτήματα εύρους* (π.χ., επιστρέψτε όλες τις χρονοσειρές εντός ενός εύρους ε της χρονοσειράς ερωτήματος) και *ο κοντινότερος γείτονας* (π.χ., επιστρέψτε τις *κ* πλησιέστερες χρονοσειρές στην χρονοσειρά ερωτήματος). Η πιο απλή απάντηση σε αυτά τα ερωτήματα είναι η διαδοχική σάρωση, η οποία όμως απαιτεί σύγκριση κάθε χρονοσειράς, πράγμα το οποίο είναι αρκετά χρονοβόρο. Οποιοδήποτε μέθοδος ευρετηρίασης που δεν εξετάζει ολόκληρο το σύνολο χρονοσειρών μπορεί να υποφέρει από δύο προβλήματα, ψευδείς συναγερμούς και ψευδείς απορρίψεις.

Ψευδείς Συναγερμοί : Ψευδείς συναγερμοί εμφανίζονται όταν αντικείμενα φαίνεται να είναι κοντά στο χώρο ευρετηρίου ενώ στην πραγματικότητα είναι απομακρυσμένα. Επειδή οι ψευδείς συναγερμοί μπορούν να αφαιρεθούν σε επόμενο στάδιο της επεξεργασίας, μπορούν να γίνουν ανεκτοί εφόσον εμφανίζονται σχετικά σπάνια.

Ψευδείς Απορρίψεις : Οι ψευδείς απορρίψεις, δηλαδή όταν αντικείμενα που πληρούν τις προϋποθέσεις παραλείπονται επειδή φαίνονται απομακρυσμένα στο χώρο ευρετηρίου, σε αντίθεση με τους ψευδούς συναγερμούς είναι συνήθως μη αποδεκτές.

Σε διάφορες ερευνητικές εργασίες έχουν γίνει προσπάθειες για τη διευκόλυνση της αναζήτησης ομοιότητας χρονοσειρών συμβολίζοντας πρώτα τα πρωτογενή (αρχικά) δεδομένα. Ωστόσο, σε αυτές τις περιπτώσεις, οι συγγραφείς εισήγαγαν ένα μέτρο απόστασης που καθορίστηκε στα νέα σύμβολα, επιτρέποντας με αυτόν τον τρόπο ψευδείς απορρίψεις σε σχέση με τα αρχικά δεδομένα. Οι αναπαραστάσεις που προτείνονται σε αυτήν την εργασία, σε αντίθεση, μετατρέπουν αρχικά τα δεδομένα (χρησιμοποιώντας την πρώτη μέθοδο και αποκτώντας έτσι την ιδιότητα του κατώτερου ορίου) και στην συνέχεια ορίζουν συμβολικές λέξεις για την εσωτερική οργάνωση και ευρετηρίαση των δεδομένων χρονοσειρών. Η ιδιότητα του κατώτερου ορίου (που διατηρείται και στις υπόλοιπες μεθόδους) επιτρέπει τη χρήση των αναπαραστάσεων για ευρετηρίαση των δεδομένων χρονοσειρών με εγγύηση χωρίς ψευδείς απορρίψεις [10].

Κεφάλαιο 2 – Τεχνικές της οικογένειας iSAX

2.1 Μια γενική μέθοδος ευρετηρίασης (μέθοδος GEMINI)

Μια χρονοσειρά *X* μήκους *n* μπορεί να θεωρηθεί ως ένα σημείο στο χώρο. Αυτό υποδηλώνει ότι οι χρονοσειρές θα μπορούσαν να ευρετηριαστούν με μεθόδους χωρικής πρόσβασης (Spatial Access Methods) όπως το R-tree [19]. Ωστόσο, η απόδοση τέτοιων μεθόδων αρχίζει να υποβαθμίζεται αρκετά γρήγορα καθώς αυξάνεται το μήκος της χρονοσειράς [20]. Προκειμένου να χρησιμοποιηθούν αυτοί οι μέθοδοι, είναι απαραίτητο να εκτελεστεί πρώτα η μείωση του μήκους της χρονοσειράς. Στην εργασία [10] περιγράφεται η μέθοδος Generic Multimedia Indexing (GEMINI) η οποία μπορεί να εκμεταλλευτεί οποιαδήποτε μέθοδο μείωσης μήκους χρονοσειράς και στη συνέχεια να ακολουθήσει η ευρετηρίαση.

Το μέτρο απόστασης στο χώρο ευρετηρίασης πρέπει να πληροί την ακόλουθη προϋπόθεση προκειμένου να διασφαλιστεί η μη εμφάνιση ψευδών απορρίψεων [10]:

$$D_{index\ space}(A,B) \leq D_{true}(A,B), \qquad (1)$$

Αυτό το θεώρημα μας εξασφαλίζει την ιδιότητα του κατώτερου ορίου [10]. Δεδομένου αυτής της ιδιότητας και της διαθεσιμότητας των μεθόδων χωρικής πρόσβασης, η μέθοδος GEMINI απαιτεί μόνο τα ακόλουθα βήματα [10]:

- Καθιέρωση ενός μέτρου απόστασης από ένα πεδίο.
- Δημιουργία μιας τεχνικής μείωσης του μήκους των δεδομένων, η οποία μειώνει το μήκος των δεδομένων από *n* σε *N*, όπου το *N* μπορεί να αντιμετωπιστεί από τη μέθοδο χωρικής πρόσβασης *(N << n).*
- Δημιουργία ενός μέτρου απόστασης το οποίο ορίζεται στην αναπαράσταση των δεδομένων με μήκος *N* και αποδεικνύεται ότι υπακούει :

$$D_{index\ space}(A,B) \leq D_{true}(A,B)$$

Όλες οι χρονοσειρές στο *Y* μετασχηματίζονται με κάποια τεχνική μείωσης μήκους χρονοσειράς και στη συνέχεια ευρετηριάζονται με μία μέθοδο χωρικής πρόσβασης. Το δέντρο ευρετηρίου αντιπροσωπεύει τις μετασχηματισμένες χρονοσειρές μήκους *N* ως σημεία στο χώρο που δημιουργήθηκε μετά το μετασχηματισμό των δεδομένων. Κάθε σημείο περιέχει ένα δείκτη στην αντίστοιχη αρχική χρονοσειρά στο δίσκο.

$\boldsymbol{Algorithm}\ BuildIndex\ (Y,n)$	// *Y* είναι τα δεδομένα, *n* είναι το μέγεθος
$\boldsymbol{for}\ i = 1\ to\ N$	// Για κάθε χρονοσειρά που πρόκειται να γίνει η ευρετηρίαση
$Y \leftarrow Y - Mean(Y)$	// (Προαιρετικό: αφαίρεσε το μέσο από κάθε Y)
$\bar{Y}_i \leftarrow SomeTransformation(Y_i)$	// Οποιαδήποτε τεχνική μείωσης μήκους
$Βάλε\ \bar{Y}_i\ μεσα\ στη\ μέθοδο\ χωρικής\ πρόσβασης\ με\ ένα\ δείκτη\ στο\ Y_i\ στο\ δίσκο$	
$\boldsymbol{end}$;	

Πίνακας 1.1: Ένα περίγραμμα του αλγορίθμου δημιουργίας ευρετηρίου GEMINI [10].

Σημειώστε ότι σε κάθε χρονοσειρά έχει αφαιρεθεί η μέση τιμή πριν από την ευρετηρίαση. Αυτό έχει ως αποτέλεσμα τη μετατόπιση της χρονοσειράς στον άξονα των y έτσι ώστε η μέση τιμή να είναι μηδέν, αφαιρώντας πληροφορίες σχετικά με την μετατόπιση του. Αυτό το βήμα περιλαμβάνεται επειδή για τις περισσότερες εφαρμογές η μετατόπιση δεν έχει σημασία κατά τον υπολογισμό της ομοιότητας. Ο Πίνακας 1.2 παρακάτω περιέχει ένα περίγραμμα του αλγορίθμου ερωτήματος εύρους GEMINI.

Algorithm $RangeQuery\ (Q, \varepsilon)$ *(το ε είναι το εύρος)*
$Προβάλλεται\ το\ ερώτημα\ \ Q\ \ στο\ χώρο\ που\ έχει\ δημιουργηθεί\ από\ την\ ευρετηρίαση.$
$Βρείτε\ όλα\ τα\ υποψήφια\ αντικείμενα\ στο\ ευρετήριο\ μεταξύ\ του\ ε\ του\ ερωτήματος$
$Ανακτήστε\ από\ το\ δίσκο\ τις\ χρονοσειρές\ που\ έδειξαν\ τα\ υποψήφια\ αντικείμενα.$
$Υπολογίστε\ τις\ πραγματικές\ αποστάσεις\ και\ αφαιρέστε\ τις\ ψευδείς\ απορρίψεις.$

Πίνακας 1.2: Ο αλγόριθμος ερωτήματος εύρους GEMINI [10].

Ο αλγόριθμος ερωτήματος εύρους «καλείται» σαν υπό-ρουτίνα στον αλγόριθμο K- κοντινότερου γείτονα που περιγράφεται στον Πίνακα 1.3.

Algorithm $K\_NearestNeigbor\ (Q, K)$
$Προβάλλεται\ το\ ερώτημα\ \ Q\ \ στο\ χώρο\ που\ έχει\ δημιουργηθεί\ από\ την\ ευρετηρίαση.$
$Βρείτε\ τα\ K-κοντινότερα\ αντικείμενα\ στο\ ευρετήριο.$
$Ανακτήστε\ από\ το\ δίσκο\ τις\ χρονοσειρές\ που\ έδειξαν\ \ τα\ υποψήφια\ αντικείμενα.$
$Υπολογίστε\ τις\ πραγματικές\ αποστάσεις\ και\ καταγράψτε\ το\ μέγιστο\ (emax).$
$Εκδόστε\ το\ ερώτημα\ εύρους, RangeQuery\ (Q, emax).$
$Υπολογίστε\ τις\ πραγματικές\ αποστάσεις\ και\ διαλέξτε\ τους\ K\ κοντινότερους\ γείτονες.$

Πίνακας 1.3: Ο αλγόριθμος κοντινότερου γείτονα GEMINI [10].

Η αποτελεσματικότητα των αλγορίθμων ερωτημάτων GEMINI εξαρτάται μόνο από την ποιότητα των μετασχηματισμών που χρησιμοποιούνται για την κατασκευή του ευρετηρίου. Όσο πιο στενά τα όρια $D_{index\ space}(A, B) \leq D_{true}(A, B)$ τόσο καλύτερα (ιδανικά θα θέλαμε $D_{index\ space}(A, B) = D_{true}(A, B)$). Το πλεονέκτημα των χρονοσειρών κατά τη μείωση του μήκους της χρονοσειράς, αποτελεί το γεγονός ότι είναι συνήθως καλοί υποψήφιοι επειδή τείνουν να περιέχουν ιδιαίτερα συσχετισμένα χαρακτηριστικά. Πιο συγκεκριμένα, τα σημεία δεδομένων τείνουν να συσχετίζονται με τους γείτονές τους (ένα φαινόμενο γνωστό ως αυτοσυσχέτιση). Επειδή τα σημεία δεδομένων συσχετίζονται με τους γείτονές τους, μπορούμε να αντιπροσωπεύσουμε αποτελεσματικά μια «γειτονιά» σημείων δεδομένων με τη μέση τιμή τους.

2.1.1 Η μέθοδος PAA (Piecewise Aggregate Approximation)

Η μέθοδος PAA (Piecewise Aggregate Approximation) [4] είναι μία τεχνική μετασχηματισμού των δεδομένων για να την επίτευξη μείωσης του μήκους της χρονοσειράς. Το πρώτο πράγμα που πρέπει να κάνουμε πριν ξεκινήσουμε τη δημιουργία του ευρετηρίου είναι να βρούμε ένα τρόπο να μειώσουμε το μήκος της χρονοσειράς. Η μέθοδος προήλθε από την απλή παρατήρηση ότι για τα

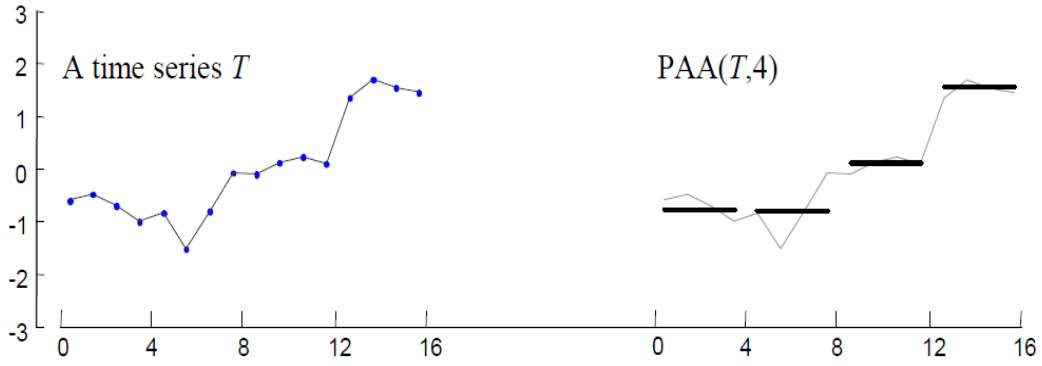
περισσότερα σύνολα δεδομένων χρονοσειρών μπορούν να προσεγγιστούν τα δεδομένα τμηματοποιώντας αυτές τις χρονοσειρές σε τμήματα ίσου μήκους και στη συνέχεια μπορεί να καταγραφεί η μέση τιμή από αυτά τα τμήματα. Χρησιμοποιώντας αυτές τις μέσες τιμές, μπορούμε στη συνέχεια να δημιουργήσουμε το ευρετήριο.

Η προσέγγιση αυτή έχει πολλά πλεονεκτήματα. Είναι απλή στην κατανόηση και στην εφαρμογή, επιτρέπει πιο ευέλικτα μέτρα απόστασης και το ευρετήριο μπορεί να κατασκευαστεί σε γραμμικό χρόνο. Επιπλέον, η μέθοδος αυτή επιτρέπει ερωτήματα (queries) που είναι μικρότερα από το μήκος για το οποίο δημιουργήθηκε το ευρετήριο.

Υποδηλώνουμε ένα ερώτημα χρονοσειρών ως $X = x_1, \ldots, x_n$. Έστω *N* είναι το μήκος της χρονοσειράς του μετασχηματισμένου χώρου που θέλουμε να ευρετηριάσουμε *(1 ≤ N ≤ n).* Μια χρονοσειρά *X* μήκους *n* αντιπροσωπεύεται στο χώρο *N* από ένα διάνυσμα $\bar{X} = \bar{x}_1, \ldots, \bar{x}_N$. Το *i* στοιχείο του $\bar{X}$ υπολογίζεται με την ακόλουθη εξίσωση [4]:

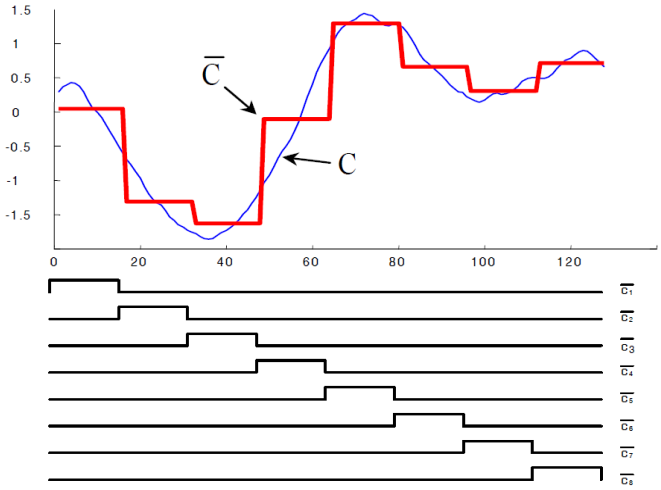
$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j \,, \qquad (2)$$

Με απλά λόγια, για τη μείωση του μήκους των δεδομένων από *n* σε *N*, τα δεδομένα χωρίζονται σε *N* "πλαίσια" ίσου μεγέθους. Υπολογίζεται η μέση τιμή των δεδομένων που εμπίπτουν σε ένα πλαίσιο και ένα διάνυσμα αυτών των τιμών γίνεται η αναπαράσταση των δεδομένων με μειωμένο μήκος (χρονοσειράς). Στο Σχήμα 1.1 απεικονίζεται αυτή η ιδέα.



Σχήμα 1.1: Μια απεικόνιση της τεχνικής μείωσης μήκους των δεδομένων που χρησιμοποιείται σε αυτή τη μέθοδο. Οι χρονοσειρές χωρίζονται σε 4 *(N)* πλαίσια και υπολογίζεται η μέση τιμή κάθε πλαισίου. Η αναπαράσταση PAA μειώνει το μήκος μιας χρονοσειράς, στην περίπτωση αυτή από 16 σε 4 [4].

Στην ειδική περίπτωση που *N = n* , η μετασχηματισμένη αναπαράσταση είναι ίδια με τη αρχική αναπαράσταση. Όταν *N = 1* η μετασχηματισμένη αναπαράσταση είναι απλά η μέση τιμή της αρχικής χρονοσειράς. Γενικότερα, ο μετασχηματισμός παράγει μια τμηματική σταθερή προσέγγιση (piecewise constant approximation) της αρχικής χρονοσειράς και η μέθοδος ονομάζεται *Piecewise Aggregate Approximation (PAA).*



Σχήμα 1.2: Η αναπαράσταση PAA μπορεί να απεικονιστεί ως μια προσπάθεια μοντελοποίησης μιας χρονοσειράς με έναν γραμμικό συνδυασμό λειτουργιών «κουτιών». Σε αυτήν την περίπτωση, μια χρονοσειρά μήκους 128 μειώνεται σε 8 [4].

Όσον αφορά τη χρονική πολυπλοκότητα για τη δημιουργία του ευρετηρίου Gemini με τη μέθοδο *Piecewise Aggregate Approximation (PAA)* για κατά προσέγγιση *m* «παράθυρα» πρέπει να υπολογίσουμε την εξίσωση 2 *N* φορές (η εξίσωση 2 έχει ένα άθροισμα μήκους *n/N)*. Ωστόσο, μπορούμε να μειώσουμε τη χρονική πολυπλοκότητα βασισμένοι στην ακόλουθη παρατήρηση. Υποθέστε ότι χρησιμοποιούμε την εξίσωση 2 για τον υπολογισμό των χαρακτηριστικών $\bar{x}_1, \ldots, \bar{x}_i, \ldots, \bar{x}_N$ για το πρώτο συρόμενο παράθυρο. Αντί να χρησιμοποιήσουμε την εξίσωση 2 για τα παράθυρα χρονοσειράς, μπορούμε απλά να προσαρμόσουμε τα χαρακτηριστικά, αφαιρώντας την επιρροή αυτού του σημείου που «σπρώχτηκε» έξω από το αριστερό του πλαισίου και λαμβάνοντας υπόψη την επιρροή του νέου σημείου που «έρχεται» στα δεξιά του πλαισίου. Πιο συγκεκριμένα το *i στο* χαρακτηριστικό στο *j στο* παράθυρο μπορεί να ενημερωθεί ως εξής [4]:

$$\bar{x}_{ij} = \bar{x}_{j-1i} - \frac{N}{n} x_{\frac{n}{N}(i-1)+1} + \frac{N}{n} x_{\frac{n}{N}(i)+1} \ , \qquad (3)$$

Με αυτό τον τρόπο κάθε φορά που αποκτάται μία νέα τιμή της χρονοσειράς μπορεί να χρησιμοποιηθεί η εξίσωση 3 για τον υπολογισμό του νέου χαρακτηριστικού, «σέρνοντας» το παράθυρο μία μονάδα προς τα δεξιά.

Όπως αναφέρθηκε προηγουμένως, για να εγγυηθούμε την μη εμφάνιση ψευδών απορρίψεων πρέπει να παράγουμε ένα μέτρο απόστασης, το οποίο να ορίζεται στο χώρο που έχει δημιουργηθεί από το ευρετήριο. Για μία χρονοσειρά X (με την οποία «χτίστηκε» το ευρετήριο) και μία χρονοσειρά Y (με την οποία υποβάλλουμε το ερώτημα), έχουμε την πραγματική τους απόσταση ως $D(X,Y)$. Μετά το μετασχηματισμό και των δύο χρονοσειρών (η X σε $\bar{X}$ και η Y σε $\bar{Y}$) η απόστασή τους (στο χώρο που έχει δημιουργηθεί από το ευρετήριο) είναι η $DR(\bar{X},\bar{Y})$. Η απόσταση αυτή πρέπει να έχει την ακόλουθη ιδιότητα: $DR(\bar{X},\bar{Y}) \leq D(X,Y)$. Το ακόλουθο μέτρο απόστασης έχει αυτήν την ιδιότητα [4]:

$$DR(\bar{X},\bar{Y}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\bar{x}_i - \bar{y}_i)^2} \ , \qquad (4)$$

2.1.2 Χειρισμός ερωτημάτων διαφόρων μηκών

Στην προηγούμενη ενότητα δείξαμε πώς να χειριστούμε ερωτήματα μήκους *n*, το μήκος για το οποίο δημιουργήθηκε η δομή ευρετηρίου. Ωστόσο, είναι πιθανό ο χρήστης να θέλει να υποβάλει ερώτημα στο ευρετήριο που να είναι μεγαλύτερα ή μικρότερα από το *n*. Για παράδειγμα, ένας χρήστης που συνήθως ενδιαφέρεται για μηνιαία μοτίβα στο χρηματιστήριο, επιθυμεί να αναζητήσει εβδομαδιαία μοτίβα. Παρακάτω δείχνεται πώς μπορούμε να εκτελέσουμε ερωτήματα διαφορετικών μηκών σε ένα μόνο σταθερού μήκους ευρετήριο. Για ευκολία, θα υποδηλώνουμε ερωτήματα μεγαλύτερα από *n* ως *XL* και ερωτήματα μικρότερα από *n* ως *XS*, με $|XL| = n_{XL}$ και $|XS| = n_{XS}$.

2.1.3 Διαχείριση σύντομων ερωτημάτων

Τα ερωτήματα που είναι μικρότερα από *n* μπορεί να αντιμετωπιστούν με τον ακόλουθο τρόπο. Ο υπολογισμός της απόστασης στην εξίσωση 4 μπορεί να έχει το ανώτερο όριο του αθροίσματός του τροποποιημένο σε [4]:

$$\sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{n_{XS}}(\bar{x}_i - \bar{y}_i)^2}\ , \qquad (5)$$

Η τροποποίηση αυτή δεν επηρεάζει την παραδοχή της μη ύπαρξης ψευδών απορρίψεων στην εξίσωση 1. Οι αλγόριθμοι αναζήτησης που δίνονται στους πίνακες 1.2 και 1.3 μπορούν να χρησιμοποιηθούν χωρίς τροποποιήσεις.

2.1.4 Διαχείριση μεγαλύτερων ερωτημάτων

Ο χειρισμός μεγαλύτερων ερωτημάτων είναι λίγο πιο δύσκολος από τη σύντομη περίπτωση ερωτημάτων. Το ευρετήριο περιέχει μόνο πληροφορίες σχετικά με τις χρονοσειρές μήκους *n*, όμως το ερώτημα *XL* έχει μήκος n_{XL} με $n_{XL} > n$. Σημειώνεται ότι η απόσταση στο χώρο ευρετηρίου μεταξύ του ερωτήματος και οποιασδήποτε πιθανής αντιστοίχισης είναι πάντα μικρότερη ή ίση με την ευκλείδεια απόσταση μεταξύ του ερωτήματος και της αντίστοιχης αρχικής χρονοσειράς. Δεδομένου αυτού του γεγονότος, οι αλγόριθμοι αναζήτησης που δίνονται στους πίνακες 1.2 και 1.3 μπορούν να χρησιμοποιηθούν με δύο μικρές τροποποιήσεις [4]:

1. Στην πρώτη γραμμή και των δύο αλγορίθμων, όταν το ερώτημα μετατρέπεται στην αναπαράσταση που χρησιμοποιείται στο ευρετήριο, πρέπει να αντικαταστήσουμε το *X* με το *XL[1:n].* Το υπόλοιπο της χρονοσειράς, *XL [n+1:n_{XL}],* αγνοείται κατά τη διάρκεια αυτής της λειτουργίας.
2. Στη δεύτερη γραμμή και των δύο αλγορίθμων, στην αρχική χρονοσειρά το πιο υποσχόμενο αντικείμενο στο ευρετήριο ανακτάται. Για μεγάλα ερωτήματα, η αρχική χρονοσειρά ανακτάται και στη συνέχεια συγκρίνεται με το XL και αυτό που προκύπτει πρέπει να έχει μήκος n_{XL} , όχι *n.*

2.2 Η μέθοδος SAX (Symbolic Aggregate Approximation)

Οι περισσότερες αναπαραστάσεις χρονοσειρών που έχουν προταθεί τις τελευταίες δεκαετίες υποφέρουν από δύο βασικά ελαττώματα. Πρώτον, το μήκος της χρονοσειράς της συμβολικής αναπαράστασης είναι η ίδια με τα αρχικά δεδομένα. Δεύτερον, τα μέτρα απόστασης που καθορίζονται στις συμβολικές προσεγγίσεις έχουν μικρή συσχέτιση με τα μέτρα απόστασης που ορίζονται στις αρχικές χρονοσειρές.

Η μέθοδος SAX (Symbolic Aggregate Approximation) [5] προσπαθεί να επιλύσει αυτά τα δύο προβλήματα. Η συμβολική αυτή αναπαράσταση επιτρέπει τη μείωση του μήκους της χρονοσειράς, μετατρέποντάς τη αρχικά σύμφωνα με τη μέθοδο PAA. Στη συνέχεια μετατρέπει επιπλέον τη χρονοσειρά σε σύμβολα (γράμματα ή και λέξεις), επιτρέποντας με αυτόν τον τρόπο οι μετρήσεις απόστασης να καθοριστούν σύμφωνα με τη συμβολική προσέγγιση (η οποία διατηρεί την ιδιότητα του κατώτερου ορίου).

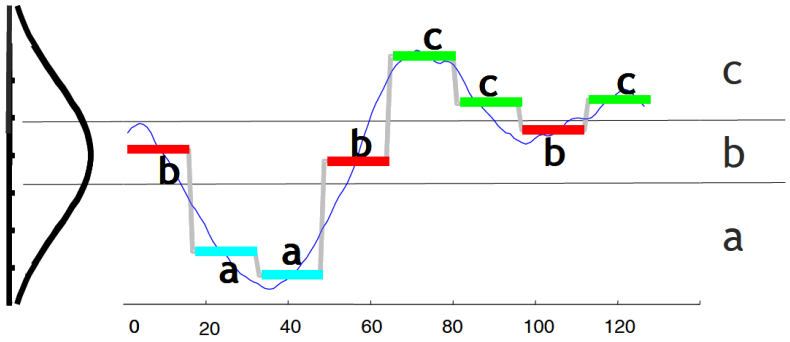
2.2.1 Παρουσίαση της SAX (Symbolic Aggregate Approximation)

Αρχικά κανονικοποιούμε τη χρονοσειρά με μέσο 0 και τυπική απόκλιση 1. Έχοντας μεταμορφώσει μια βάση δεδομένων χρονοσειρών σε PAA μπορούμε να εφαρμόσουμε έναν περαιτέρω μετασχηματισμό για να αποκτήσουμε μια διακριτή αναπαράσταση. Είναι επιθυμητό να έχουμε μια τεχνική διακριτοποίησης που θα παράγει σύμβολα με ίση πιθανότητα. Αυτό επιτυγχάνεται εύκολα αφού οι κανονικοποιημένες χρονοσειρές έχουν κατανομή Gauss. Αυτό μπορεί να διευκρινιστεί με τα διαγράμματα κανονικής πιθανότητας (normal probability plot) των δεδομένων. Μία κατά προσέγγιση ευθεία γραμμή δείχνει ότι τα δεδομένα κατανέμονται κανονικά.
Για μεγάλο μέγεθος δεδομένων χρονοσειρών η υπόθεση Gauss δεν παραβιάζεται. Για μικρότερο υποσύνολο δεδομένων όπου η υπόθεση δεν τηρείται, η απόδοση ελαττώνεται ελαφρώς. Ωστόσο, η ορθότητα του αλγορίθμου δεν επηρεάζεται. Η ορθότητα του αλγορίθμου διασφαλίζεται από την ιδιότητα κατώτερου ορίου του μέτρου απόστασης στο συμβολικό χώρο.
Δεδομένου ότι οι κανονικοποιημένες χρονοσειρές έχουν κατανομή Gauss με υψηλή πιθανότητα, μπορούμε απλά να προσδιορίσουμε τα «σημεία διακοπής» (breakpoints) που θα παράγουν *a* περιοχές ίσου μεγέθους κάτω από την καμπύλη Gauss.

βi \ *a*	2	3	4	5	6	7	8
β_1	0.00	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2		0.43	0.00	-0.25	-0.43	-0.57	-0.67
β_3			0.67	0.25	0.00	-0.18	-0.32
β_4				0.84	0.43	0.18	0.00
β_5					0.97	0.57	0.32
β_6						1.07	0.67
β_7							1.15

Πίνακας 2.1: Ένας πίνακας αναζήτησης που περιέχει τα «σημεία διακοπής» (breakpoints) που διαιρούν μια κατανομή Gauss σε έναν αυθαίρετο αριθμό (από 2 έως 8) ίσης πιθανότητας περιοχών [5].

«Σημεία διακοπής» (Breakpoints): Τα «σημεία διακοπής» είναι μια ταξινομημένη λίστα αριθμών $B = \beta_1, \ldots, \beta_{\alpha-1}$ έτσι ώστε η περιοχή κάτω από μια *N(0,1)* καμπύλη Gauss από το β_i στο β_{i+1} να είναι ίση με $1/a$ (β_0 και β_α ορίζονται ως - *∞* και *∞,* αντίστοιχα). Αυτά τα «σημεία διακοπής» μπορεί να προσδιοριστούν αναζητώντας τα σε έναν στατιστικό πίνακα. Για παράδειγμα, ο Πίνακας 2.1 παρέχει τα «σημεία διακοπής» για τιμές του *a* από 2 έως 8 [5].
Αρχικά δημιουργείται μια αναπαράσταση PAA της χρονοσειράς και στη συνέχεια βρίσκουμε τα «σημεία διακοπής». Όλοι οι συντελεστές PAA που είναι κάτω από το μικρότερο «σημείο διακοπής» απεικονίζονται στο σύμβολο "**a"**, όλοι οι συντελεστές μεγαλύτεροι ή ίσοι από το μικρότερο «σημείο διακοπής» και μικρότεροι από το δεύτερο μικρότερο «σημείο διακοπής» απεικονίζονται στο σύμβολο "**b"** , κλπ. Το Σχήμα 2.1 απεικονίζει την ιδέα.



Σχήμα 2.1: Μια χρονοσειρά διαχωρίζεται αποκτώντας πρώτα μια προσέγγιση από την αναπαράσταση PAA και στη συνέχεια χρησιμοποιώντας προκαθορισμένα «σημεία διακοπής» (breakpoints) για την απεικόνιση των συντελεστών PAA σε σύμβολα SAX. Στο παραπάνω παράδειγμα, με *n = 128*, *w = 8* και *a = 3*, οι χρονοσειρές απεικονίζονται στη λέξη **baabccbc** [5]**.**

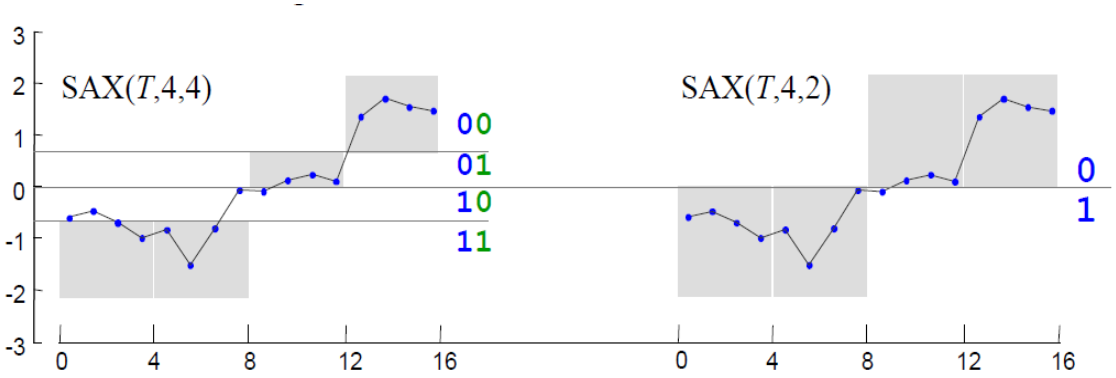
Σημειώστε ότι σε αυτό το παράδειγμα τα 3 σύμβολα **"a", "b" και "c" "** είναι σχεδόν κατανεμημένα με ίση πιθανότητα όπως θέλαμε.

Η SAX επιτρέπει τη μείωση μιας χρονοσειράς *T* αυθαίρετου μήκους *n* σε μια χρονοσειρά αυθαίρετου μήκους *w*, (*w << n*). Το μέγεθος του αλφαβήτου είναι επίσης ένας αυθαίρετος ακέραιος *a*, όπου *a > 2*. Αυτός ο ακέραιος *a* καλείται πληθικότητα (cardinality).

Λέξη: Μια χρονοσειρά *C* μήκους *n* μπορεί να αναπαρασταθεί ως μια *λέξη* $\hat{C} = \hat{c}_1, \ldots, \hat{c}_w$ ως εξής. Έστω $alpha_i$ δηλώνει το i στοιχείο του αλφαβήτου, δηλαδή, $alpha_1$ = **a** και $alpha_2$ = **b.** Στη συνέχεια, η απεικόνιση από μια προσέγγιση μέσω της PAA σε μια *λέξη* $\hat{C}$ λαμβάνεται ως εξής [5]:

$$\hat{c}_i = alpha_j, \qquad iif \ \ \beta_{j-1} \leq \bar{c}_i < \beta_j$$

Προηγουμένως, εκπροσωπήσαμε κάθε σύμβολο *SAX* ως ένα γράμμα ή έναν ακέραιο. Εδώ, ωστόσο, θα χρησιμοποιήσουμε δυαδικούς αριθμούς.



Σχήμα 2.2: Μια χρονοσειρά *T* μετατρέπεται σε *SAX* λέξεις με πληθικότητα 4 { 11 , 11 , 01 , 00 } (αριστερά), και με πληθικότητα 2 { 1 , 1 , 0 , 0 } (δεξιά) [5].

Μια λέξη *SAX* είναι απλά ένα διάνυσμα διακριτών συμβόλων. Χρησιμοποιούμε ένα **έντονο γράμμα** για να κάνουμε διάκριση μεταξύ μιας αρχικής χρονοσειράς και της *SAX* έκδοσής. Θα υποδηλώνουμε την πληθικότητα της λέξης *SAX* με έναν εκθέτη [5]:

$$SAX(T, w, a) = \boldsymbol{T^a} = \{t_1, t_2, \ldots, t_{w-1}, t_w\}$$

Στο Σχήμα 2.2 έχουμε μετατρέψει μια χρονοσειρά *T* μήκους 16 σε λέξεις *SAX*. Και τα δύο παραδείγματα έχουν μήκος λέξης 4, αλλά το ένα έχει πληθικότητα 4 και το άλλο έχει πληθικότητα 2. Επομένως έχουμε: *SAX (T,4,4) = T^4 = { 11 , 11 , 01 , 00 } και SAX (T,4,2) = T^2 = { 1 , 1 , 0 , 0 }.* Παρατηρούμε ότι μόλις έχουμε την T^4 μπορούμε να αντλήσουμε την T^2 απλώς αγνοώντας τα λιγότερο σημαντικά bits σε καθένα από τα τέσσερα σύμβολα στη λέξη *SAX*. Για παράδειγμα, εάν μετατρέψουμε τη χρονοσειρά *T* σε *SAX* με πληθικότητα του 8, έχουμε : *SAX (T,4,8) = T^8 = { 110 , 110 , 011 , 000 }.* Από αυτή, μπορούμε να τη μετατρέψουμε σε οποιαδήποτε χαμηλότερης ανάλυσης που διαφέρει με δύναμη δύο, απλά αγνοώντας το σωστό αριθμό bits. Ο Πίνακας 2.2 το καθιστά σαφέστερο.

SAX(*T*,4,16)	$\mathbf{T^{16}}$	{1100,1101,0110,0001}
SAX(*T*,4,8)	$\mathbf{T^{8}}$	{110 ,110 ,011 ,000 }
SAX(*T*,4,4)	$\mathbf{T^{4}}$	{11 ,11 ,01 ,00 }
SAX(*T*,4,2)	$\mathbf{T^{2}}$	{1 ,1 ,0 ,0 }

Πίνακας 2.2: Η δυνατότητα απόκτησης μειωμένης (κατά μισής) πληθικότητας λέξης SAX απλώς αγνοώντας τα υπόλοιπα bits [5].

2.2.2 Μέτρο απόστασης

Έχοντας εισαγάγει την αναπαράσταση χρονοσειρών, μπορούμε τώρα να καθορίσουμε ένα μέτρο απόστασης σε αυτό. Λαμβάνοντας υπόψη δύο χρονοσειρές *T* και *S* η ευκλείδεια απόσταση τους (το πιο σύνηθες μέτρο) είναι:

$$D(T, S) \equiv \sqrt{\sum_{i=1}^{n} (T_i - S_i)^2}$$

Εάν μετατρέψουμε τις αρχικές χρονοσειρές σε αναπαραστάσεις PAA, $\bar{T}$ και $\bar{S}$, μπορούμε στη συνέχεια να αποκτήσουμε μια κατώτερη οριακή προσέγγιση της ευκλείδειας απόστασης μεταξύ των αρχικών χρονοσειρών με:

$$DR(\bar{T},\bar{S}) \equiv \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(\bar{t}_i - \bar{s}_i)^2}$$

Εάν μετατρέψουμε περαιτέρω τα δεδομένα σε συμβολική αναπαράσταση, μπορούμε να ορίσουμε μια συνάρτηση MINDIST που επιστρέφει την ελάχιστη απόσταση μεταξύ της αρχικής χρονοσειράς δύο *λέξεων* [5]:

$$MINDIST(\boldsymbol{T^2},\boldsymbol{S^2}) \equiv \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(t_i - s_i))^2}$$

	00	01	10	11
00	0	0	0.67	1.34
01	0	0	0	0.67
10	0.67	0	0	0
11	1.34	0.67	0	0

Πίνακας 2.3: Ένας πίνακας αναζήτησης SAX απόστασης dist για α = 4 [5].

Η απόσταση μεταξύ δύο συμβόλων μπορεί να διαβαστεί εξετάζοντας την αντίστοιχη σειρά και στήλη. Για παράδειγμα, *dist (00 , 01) = 0* και *dist (00 , 10) = 0,67*.

Για λόγους σαφήνειας, θα δώσουμε ένα συγκεκριμένο παράδειγμα για τον τρόπο υπολογισμού αυτού του μέτρου. Εάν δημιουργήσουμε μια χρονοσειρά *S* που είναι απλώς εικόνα της *T*, τότε η ευκλείδεια απόσταση μεταξύ τους είναι *D(T,S) = 46.06*.

Όπως έχουμε ήδη δει, *SAX (T,4,4) = T* ^{4} *= { 11 , 11 , 01 , 00 }*, και επομένως *SAX (S,4,4) = S* ^{4} *= { 00 , 01 , 11 , 11 }*

Η επίκληση της συνάρτησης MINDIST θα χρησιμοποιεί στον πίνακα αναζήτησης που φαίνεται στον Πίνακα 2.3 για να βρει:

dist (t_1, s_1) *= dist (11, 00) = 1.34*
dist (t_2, s_2) *= dist (11, 01) = 0.67*
dist (t_3, s_3) *= dist (01, 11) = 0.67*
dist (t_4, s_4) *= dist (00, 11) = 1.34*

Το οποίο, όταν συνδέεται στη συνάρτηση MINDIST, μας δίνει:

$$MINDIST(\boldsymbol{T^2},\boldsymbol{S^2}) = \sqrt{\frac{16}{4}}\sqrt{(1.34)^2 + (0.67)^2 + (0.67)^2 + (1.34)^2} = 4.237$$

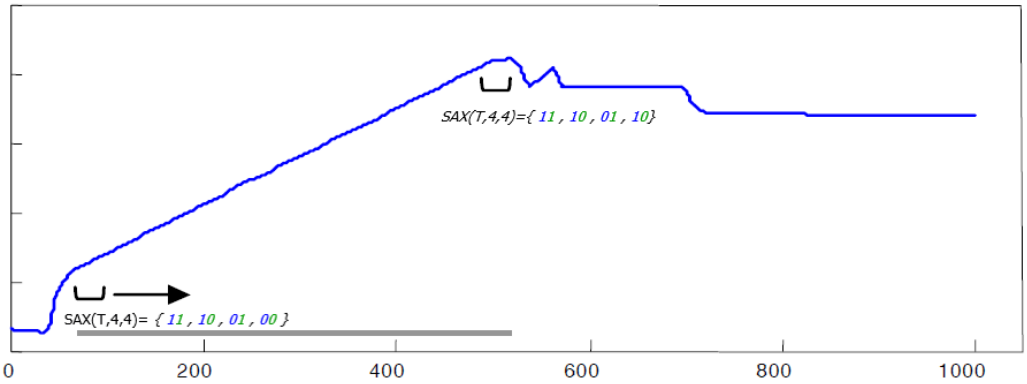
Οπότε παράγεται μία τιμή κατώτερου ορίου *4,237*. Σε αυτήν την περίπτωση, το κάτω όριο είναι αρκετά χαλαρό. Ωστόσο, η ύπαρξη είτε περισσότερων συμβόλων *SAX* είτε υψηλότερης πληθικότητας *(a)* θα παράγει ένα πιο αυστηρό κατώτερο όριο. Εάν θέλουμε να προσεγγίσουμε ένα τεράστιο σύνολο δεδομένων στην κύρια μνήμη, οι παράμετροι *w* και *a* πρέπει να επιλεγούν με τέτοιο τρόπο ώστε η προσέγγιση να κάνει την καλύτερη δυνατή χρήση της κύριας μνήμης που είναι διαθέσιμη. Η καλύτερη επιλογή εξαρτάται σε μεγάλο βαθμό από τα δεδομένα και την πληθικότητα των λέξεων SAX. Η στενότητα των κάτω ορίων μπορεί να μετρηθεί, η οποία ορίζεται ως η κατώτερη οριακή απόσταση σε σχέση με την πραγματική απόσταση [4]:

$$Tightness\ of\ Lower\ Bound = \frac{MINDIST(\hat{T},\hat{S})}{D(T,S)}$$

Προηγούμενες μελέτες δείχνουν ότι μεγαλύτερα μεγέθη αλφαβήτου αποδίδουν καλύτερα αποτελέσματα, η απόδοση ωστόσο μειώνεται καθώς το *a* αυξάνεται από ένα σημείο και μετά. Καθώς ο χώρος αποθήκευσης αποτελεί ζήτημα, η καλύτερη επιλογή είναι μια λογική ισορροπία μεταξύ του χώρου αποθήκευσης και της στενότητας του κατώτερου ορίου. Η υπερβολική αύξηση του μεγέθους του αλφαβήτου απαιτεί περισσότερα bits για την αναπαράσταση κάθε αλφαβήτου.

2.2.3 Μείωση αριθμητικότητας

Με τη συμβολική προσέγγιση SAX μια αρχική χρονοσειρά *T* μετατρέπεται σε σύμβολα. Η αριθμητικότητα έχει την έννοια του αριθμού των εμφανίσεων μίας λέξης SAX. Χρονοσειρές μήκους *n* εξάγονται με χρήση ενός συρόμενου παραθύρου και στη συνέχεια αποθηκεύονται σε ένα πίνακα για περαιτέρω χειρισμό [10], [12].



Σχήμα 2.4: Έστω μια χρονοσειρά T και σε κάποιο σημείο η λέξη που εξάγεται είναι η SAX (T,4,4) = $\mathbf{T^4}$ = { *11 , 10, 01 , 00* }. Η ίδια λέξη συνεχίζει να εξάγεται οπότε χρειάζεται η καταγραφή ενός μόνου δείκτη μόνο στην πρώτη εμφάνιση [5].

Ένα επιπλέον χαρακτηριστικό της προτεινόμενης συμβολικής προσέγγισης SAX είναι ότι μας δίνει την επιλογή της μείωσης της αριθμητικότητας. Εάν η πρώτη λέξη που εξάγεται (κατά τη μετατροπή της χρονοσειράς σε λέξη SAX) είναι *SAX (T,4,4) =* $\mathbf{T^4}$ *= { 11 , 10 , 01 , 00 }* και το συρόμενο παράθυρο μετατοπίζεται για να ανακαλύψει ότι και η δεύτερη λέξη είναι *(T,4,4)={ 11 , 10 , 01 , 00 }*, μπορούμε να αποφασίσουμε να μην συμπεριλάβουμε τη δεύτερη εμφάνιση της λέξης στον πίνακα. Εάν χρειαστεί ποτέ να ανακτήσουμε όλες τις εμφανίσεις του *(T,4,4)={ 11 , 10 , 01 , 00 }*, μπορούμε να μεταβούμε στην τοποθεσία που

επισημαίνεται από τα πρώτα περιστατικά και να θυμηθούμε να εμφανίσουμε το συρόμενο παράθυρο προς τα δεξιά, ελέγχοντας αν το επόμενο παράθυρο έχει επίσης απεικονιστεί στην ίδια λέξη. Μπορούμε να σταματήσουμε τις δοκιμές μόλις αλλάξει η λέξη.
Σε σύνολα δεδομένων που χαρακτηρίζονται από μεγάλες περιόδους μικρής ή καθόλου κίνησης, ακολουθούμενες από εκρήξεις δραστηριότητας (τα σεισμικά δεδομένα είναι ένα παράδειγμα) η μείωση της αριθμητικότητας μπορεί να είναι τεράστια.

2.3. Η μέθοδος iSAX (Indexable Symbolic Aggregate ApproXimation)

Τα τελευταία χρόνια, το αυξανόμενο επίπεδο ενδιαφέροντος στην ευρετηρίαση και την εξόρυξη δεδομένων χρονοσειρών γενικότερα, έχει δημιουργήσει πολλούς αλγόριθμους και αναπαραστάσεις. Ωστόσο, η ταχεία ανάπτυξη σε πολλούς τομείς, έχει οδηγήσει σε ολοένα και μεγαλύτερη βελτίωση των τεχνικών σύλληψης και αποθήκευσης δεδομένων, με αποτέλεσμα να αυξάνονται μέρα με τη μέρα τα δεδομένα που αποθηκεύονται, δημιουργώντας με αυτό τον τρόπο τεράστια σε μέγεθος σύνολα δεδομένων χρονοσειρών. Τέτοια σύνολα δεδομένων είναι αρκετά δύσκολο να ευρετηριαστούν καθώς ο χρόνος δημιουργίας του ευρετηρίου είναι αρκετά μεγάλος.

Η μέθοδος iSAX (Indexable Symbolic Aggregate ApproXimation) [6] είναι μία συμβολική αναπαράσταση η οποία προσπαθεί να επιλύσει αυτό το πρόβλημα. Πέρα από το γεγονός ότι η μέθοδος iSAX καταφέρνει και δημιουργεί το ευρετήριο σε σύντομο χρονικό διάστημα (σε σχέση με άλλους μεθόδους), επιτρέπει επίσης την πολύ γρήγορη αναζήτηση ομοιότητας σε μεγάλα σύνολα δεδομένων χρονοσειρών.

2.3.1 Παρουσίαση της *iSAX*

Η μέθοδος *iSAX* αποτελεί μία επέκταση της κλασικής *SAX*. Η μέθοδος *SAX* έκανε χρήση ακεραίων ή αλφαριθμητικών χαρακτήρων που αντιπροσώπευαν τα σύμβολα *SAX* . Ένα παράδειγμα αποτύπωσης μίας λέξης *SAX* είναι:

$SAX\ (T,4,8) = \boldsymbol{T^8} = \{ 110 , 110 , 011 , 000 \} = \{6, 6, 3, 0\}$

Η συνεισφορά της μεθόδου *iSAX* είναι ότι μας επιτρέπει να χρησιμοποιήσουμε την αναπαράσταση *SAX* ώστε να συγκρίνουμε λέξεις με διαφορετική πληθικότητα, ή ακόμη και διαφορετικές πληθικότητες μέσα στην ίδια την λέξη. Ο τρόπος αποτύπωσης μίας λέξης *iSAX* είναι γράφοντας τις πληθικότητες ως εκθέτη. Οπότε και το από πάνω παράδειγμα μπορεί να γραφτεί ως:

$iSAX\ (T,4,8) = \boldsymbol{T^8} = \{ 6^8, 6^8, 3^8, 0^8 \}$

2.3.2 Σύγκριση διαφορετικών λέξεων *iSAX*

Είναι δυνατόν να συγκρίνουμε δύο λέξεις *iSAX* διαφορετικής πληθικότητας. Ας υποθέσουμε ότι έχουμε δύο χρονοσειρές *T* και *S*, οι οποίες έχουν μετατραπεί σε λέξεις *iSAX* :

$iSAX\ (T,4,8) = \boldsymbol{T^8} = \{110 , 110 , 011 , 000 \} = \{6^8, 6^8, 3^8, 0^8\}$

$iSAX\ (S,4,2) = \boldsymbol{S^2} = \{0 , 0 , 1 , 1 \} = \{0^2, 0^2, 1^2, 1^2\}$

Μπορούμε να βρούμε το κατώτερο όριο μεταξύ της *T* και *S*, παρόλο που οι λέξεις *iSAX* που τις αντιπροσωπεύουν είναι διαφορετικής πληθικότητας. Το «κόλπο» είναι να προάγουμε την πληθικότητα της χαμηλότερης αναπαράστασης σε μεγαλύτερη πληθικότητα προτού την βάλουμε στη συνάρτηση MINDIST.

Μπορούμε να σκεφτούμε ότι μπορούμε να προάγουμε προσωρινά την S^2 λέξη ως $S^8 = \{ 0^{**}_1, 0^{**}_2, 1^{**}_3, 1^{**}_4\}$. Τότε το ερώτημα απλά γίνεται : «ποιες είναι σωστές τιμές που λείπουν στα $**_i$ bits»; Σημειώστε ότι και οι δύο πληθικότητες μπορούν να εκφραστούν ως η δύναμη κάποιου ακέραιου. Αυτό εγγυάται μία επικάλυψη στα σημεία διακοπής που χρησιμοποιούνται κατά τον υπολογισμό της *SAX*.

Χρησιμοποιώντας αυτήν την πληροφορία, μπορούμε να λάβουμε τις τιμές bit που λείπουν εξετάζοντας για κάθε θέση i γνωστά bits της S^8, τα οποία ορίζονται ως S_i^k (k είναι η πληθικότητα μετά την προαγωγή της) και τα αντίστοιχα bits στην T_i^8 [6]:

IF S_i^k *σχηματίζει ένα πρόθεμα για την* T_i^8, **THEN**,
$*_i = T_i^8$ *για όλα τα άγνωστα bits.*
ELSE IF S_i^k *είναι λεξικογραφικά μικρότερο από τις αντίστοιχες τιμές στην* T_i^8, **THEN**,
$*_i = 1$ *για όλα τα άγνωστα bits.*
ELSE,
$*_i = 0$ *για όλα τα άγνωστα bits.*

Στο δικό μας παράδειγμα για την $S^8 = \{ 0^{**}_1, 0^{**}_2, 1^{**}_3, 1^{**}_4\}$:

Για i=1 , το 0 δεν σχηματίζει πρόθεμα για την $T_1^8(1)$*, αλλά το 0 είναι λεξικογραφικά μικρότερο από το* $T_1^8(1)$*, οπότε και τα bits που λείπουν για το θα πάρουν όλα την τιμή 1. Το ίδιο συμβαίνει και για i=2.*

Για i=3 , το 1 δεν σχηματίζει πρόθεμα για την $T_3^8(0)$*, ούτε το 1 είναι λεξικογραφικά μικρότερο από το* $T_3^8(0)$*, οπότε και τα bits που λείπουν για το θα πάρουν όλα την τιμή 0. Το ίδιο συμβαίνει και για i=4.*

Οπότε και η $S^8 = \{ 0^{**}_1, 0^{**}_2, 1^{**}_3, 1^{**}_4\}$ γίνεται:

$S^8 = \{ 011 , 011 , 100 , 100 \}$

Αυτή η μέθοδος καθιστά την αναπαράσταση του S^8 χρησιμοποιήσιμη για τον υπολογισμό της *MINDIST*. Είναι σημαντικό να σημειωθεί ότι αυτή δεν είναι απαραίτητα η ίδια λέξη *iSAX* που θα είχαμε πάρει αν είχαμε μετατρέψει την αρχική χρονοσειρά *S*. Δεν μπορούμε να αναιρέσουμε αυτήν την απώλεια. Ωστόσο, η χρήση αυτής της λέξης *iSAX* μας δίνει ένα αποδεκτό κατώτερο όριο.

Τέλος, σημειώστε ότι, εκτός από τη σύγκριση λέξεων *iSAX* διαφορετικής πληθικότητας, το τέχνασμα «προαγωγής» που περιγράφεται παραπάνω μπορεί να χρησιμοποιηθεί για τη σύγκριση λέξεων *iSAX* όπου κάθε λέξη έχει μικτές πληθικότητες. Για παράδειγμα, μπορούμε να επιτρέψουμε λέξεις *iSAX* όπως $\{ 111 , 11 , 101 , 0 \} = \{ 7^8, 3^4, 5^8, 0^2\}$. Εάν υπάρχουν τέτοιες λέξεις, μπορούμε απλώς να ευθυγραμμίσουμε τις δύο εν λόγω λέξεις, να σαρώσουμε κάθε ζεύγος αντίστοιχων συμβόλων και να προάγουμε το σύμβολο με χαμηλότερη πληθικότητα στην ίδια πληθικότητα με το σύμβολο με μεγαλύτερη πληθικότητα.

2.3.3 Η διαίσθηση πίσω από την ευρετηρίαση (indexing) *iSAX*

Η κλασική αναπαράσταση *SAX* προσφέρει τη δυνατότητα της ευρετηρίασης επιλέγοντας μια σταθερή πληθικότητα. Ας υποθέσουμε για παράδειγμα ότι έχουμε ένα εκατομμύριο χρονοσειρές προς ευρετηρίαση. Έστω ότι επιλέγουμε μία πληθικότητα *(a) 8* και μήκος λέξης *(w) 4* οπότε θα έχουμε $8^4=4096$ τμήματα (αρχεία). Το κάθε αρχείο θα έχει κατά μέσο όρο 244 χρονοσειρές σε αυτό (1.000.000/4.096). Όπως θα παρατηρούσαμε αν δημιουργούσαμε το ευρετήριο αυτό δε συμβαίνει, καθώς θα βρίσκαμε μία τεράστια λοξότητα στην κατανομή, με τα περισσότερα από τα μισά αρχεία να είναι κενά, ενώ το μεγαλύτερο αρχείο ίσως να περιέχει μεγάλο μέρος του συνόλου των δεδομένων. Τα κενά αρχεία όσο και το πολύ μεγάλο σε μέγεθος αρχείο αποτελούν περιπτώσεις που είναι ανεπιθύμητες για ευρετηρίαση. Εάν υποθέσουμε ότι το μεγάλο σε μέγεθος αρχείο έχει το 20% των δεδομένων, τότε κατά την διαδικασία ανάκτησης πληροφορίας από το δίσκο, μπορούμε να είμαστε το πολύ πέντε φορές πιο γρήγοροι από τη διαδοχική σάρωση.

Η μέθοδος iSAX προσπαθεί να επιλύσει αυτό το πρόβλημα, καθώς δίνει τη δυνατότητα στο χρήστη να ορίσει ένα όριο *th* (ο μέγιστος αριθμός χρονοσειρών σε ένα αρχείο), διασφαλίζοντας ότι κάθε αρχείο έχει τουλάχιστον μία και το πολύ *th* χρονοσειρές σε αυτό.

Έστω για παράδειγμα ότι βρισκόμαστε στη διαδικασία δημιουργίας του ευρετηρίου και έχουμε επιλέξει *th = 100*. Σε κάποιο σημείο της διαδικασίας μπορεί να υπάρχουν ακριβώς 100 χρονοσειρές που απεικονίζονται στη λέξη *iSAX* $\{2^4, 3^4, 3^4, 2^4\}$. Εάν, καθώς συνεχίζουμε να χτίζουμε το ευρετήριο, βρίσκουμε και άλλες απεικονίσεις χρονοσειρών, έχουμε μια «υπερχείλιση», οπότε χωρίζουμε το αρχείο. Η ιδέα είναι να επιλέξουμε ένα σύμβολο *iSAX*, να εξετάσουμε ένα επιπλέον bit και να χρησιμοποιήσουμε την αξία του για να δημιουργήσουμε δύο νέα αρχεία. Σε αυτήν την περίπτωση [6]:

Το Original file: $\{2^4, 3^4, 3^4, 2^4\}$ *χωρίζεται σε…*

"Child" file 1: $\{4^{\mathbf{8}}, 3^4, 3^4, 2^4\}$

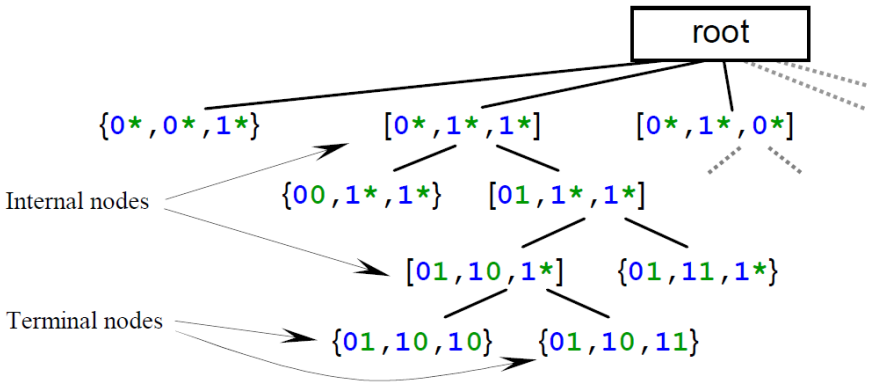
"Child" file 2: $\{5^{\mathbf{8}}, 3^4, 3^4, 2^4\}$

Σημειώστε ότι στο από πάνω παράδειγμα χωρίσαμε το πρώτο σύμβολο, προάγοντας την πληθικότητα, από 4 σε 8. Για κάποιες χρονοσειρές στο αρχείο, η επιπλέον τιμή στο πρώτο τους σύμβολο *iSAX* ήταν *1* (απεικόνιση εκ νέου στο *Child 1),* και για άλλες ήταν *0* (απεικόνιση εκ νέου στο *Child 2).* Τα «*Child*» αρχεία μπορούν να ονομαστούν με κάποιο τρόπο που να δείχνει τη μεταβλητή πληθικότητα τους, για παράδειγμα ***5.8***_3.4_3.4_2.4.txt και ***4.8***_3.4_3.4_2.4.txt.

2.3.4 Κατασκευή ευρετηρίου (index) *iSAX*

Η χρήση της μεθόδου *iSAX* επιτρέπει τη δημιουργία δομών ευρετηρίου που είναι ιεραρχικές και περιέχουν μη επικαλυπτόμενες περιοχές [21]. Για μία πιο

πλήρη εικόνα, απεικονίζεται στο Σχήμα 3.1 μια απλή δομή ευρετηρίου της μεθόδου *iSAX* που βασίζεται σε δέντρο απόφασης.



Σχήμα 3.1: Μια απεικόνιση ενός ευρετηρίου (index) *iSAX* [6].

Το ευρετήριο κατασκευάζεται δεδομένης βασικής πληθικότητας *b*, μήκους λέξης *w* και ορίου *th*. Η δομή ευρετηρίου υποδιαιρεί ιεραρχικά το χώρο *iSAX* (το χώρο δηλαδή που δημιουργείται μετά την μετατροπή των αρχικών χρονοσειρών σε λέξεις *iSAX* [σύμβολα]), έως ότου ο αριθμός καταχωρήσεων χρονοσειρών σε κάθε υπό-περιοχή να πέσει κάτω από το όριο *th*. Η αξιολόγηση μεταξύ κόμβων ή χρονοσειρών γίνεται μέσω σύγκρισης λέξεων *iSAX* και οι τρεις κλάσεις κόμβων που βρίσκονται σε ένα δέντρο και η αντίστοιχη λειτουργικότητά τους περιγράφονται παρακάτω:

Τερματικός κόμβος (Terminal Node): Ένας τερματικός κόμβος είναι ένας κόμβος «φύλλων» που περιέχει ένα δείκτη (ο οποίος μας δείχνει το αρχείο ευρετηρίου στο οποίο είναι καταχωρημένες οι αρχικές χρονοσειρές). Ο κόμβος αποθηκεύει τη λέξη *iSAX* με την υψηλότερη πληθικότητα για κάθε χρονοσειρά.

Εσωτερικός κόμβος (Internal Node): Ένας εσωτερικός κόμβος προσδιορίζει μια διαίρεση στο χώρο *iSAX* και δημιουργείται όταν ο αριθμός των χρονοσειρών που περιέχονται σε έναν τερματικό κόμβο υπερβαίνει το όριο *th*. Ο εσωτερικός κόμβος χωρίζει τον χώρο *iSAX* με την προαγωγή των τιμών πληθικότητας των λέξεων. Οι χρονοσειρές από τον τερματικό κόμβο που ενεργοποίησαν το διαχωρισμό εισάγονται στον νέο δημιουργημένο εσωτερικό κόμβο και κατακερματίζονται στις αντίστοιχες θέσεις τους. Εάν ο κατακερματισμός δεν περιέχει αντιστοίχιση καταχώρισης *iSAX*, δημιουργείται ένας νέος τερματικός κόμβος (πριν την εισαγωγή των χρονοσειρών) και ο κατακερματισμός ενημερώνεται ανάλογα. Έτσι, οι εσωτερικοί κόμβοι αποθηκεύουν μια αναπαράσταση SAX και δείκτες των δύο παιδιών (child) τους.

Κόμβος Ρίζας (Root Node): Ο κόμβος ρίζας αποτελεί έναν αντιπρόσωπο του πλήρους δημιουργημένου χώρου *iSAX* και έχει παρόμοια λειτουργικότητα με έναν εσωτερικό κόμβο. Ο κόμβος ρίζας αξιολογεί τις χρονοσειρές στη βασική πληθικότητα, δηλαδή *b*. Ο κόμβος ρίζας δεν περιέχει αναπαράσταση *SAX*, αλλά μόνο δείκτες για τους θυγατρικούς (child) κόμβους (στη χειρότερη περίπτωση, ο αριθμός τους είναι 2^w).

```
1   Function Insert(ts)
2   iSAX_word = iSAX(ts, this.parameters)
3
4   if Hash.ContainsKey(iSAX_word)
5       node = Hash.ReturnNode(iSAX_word)
6       if node is terminal
7           if SplitNode() == false
8               node.Insert(ts)
9           else
10              newnode = new internal
11              newnode.Insert(ts)
12              foreach ts in node
13                  newnode.Insert(ts)
14              end
15              Hash.Remove(iSAX_word, node)
16              Hash.Add(iSAX_word, newnode)
17          endif
18      elseif node is internal
19          node.Insert(ts)
20      endif
21  else
22      newnode = new terminal
23      newnode.Insert(ts)
24      Hash.Add(iSAX_word, newnode)
25  endif
```

Πίνακας 3.1: Ψευδοκώδικας συνάρτησης εισαγωγής ευρετηρίου iSAX [6].

Ο ψευδοκώδικας της συνάρτησης εισαγωγής που χρησιμοποιείται για την κατασκευή ευρετηρίου φαίνεται στον Πίνακα 3.1. Δεδομένης μιας χρονοσειράς για εισαγωγή, λαμβάνουμε πρώτα την αναπαράσταση της λέξης *iSAX* χρησιμοποιώντας τις αντίστοιχες παραμέτρους *iSAX* στον τρέχοντα κόμβο (γραμμή 2). Εάν ο πίνακας κατακερματισμού δεν περιέχει ακόμη μια καταχώριση για τη λέξη *iSAX*, δημιουργείται ένας τερματικός κόμβος για την αναπαράσταση *SAX* και η χρονοσειρά εισάγεται ανάλογα (γραμμές 22-24). Διαφορετικά, υπάρχει μια καταχώριση στον πίνακα κατακερματισμού και λαμβάνεται ο αντίστοιχος κόμβος. Εάν αυτός ο κόμβος είναι εσωτερικός κόμβος, ζητούμε τη λειτουργία εισαγωγής του αναδρομικά (γραμμή 19). Εάν ο κόμβος είναι τερματικός κόμβος, η πληρότητα αξιολογείται για να προσδιοριστεί εάν μία πρόσθετη εισαγωγή απαιτεί διαχωρισμό (γραμμή 7). Αν ναι, δημιουργείται ένας νέος εσωτερικός κόμβος και εισάγονται όλες οι εγγραφές που περικλείονται από τον υπερπληρωμένο τερματικό κόμβο (γραμμές 10-16). Σε διαφορετική περίπτωση, υπάρχει αρκετός χώρος και η καταχώρηση προστίθεται απλά στον τερματικό κόμβο (γραμμή 8).

2.3.5 Κατά προσέγγιση Αναζήτηση

Ένα ευρετήριο *iSAX* μπορεί να υποστηρίξει πολύ γρήγορες κατά προσέγγιση αναζητήσεις. Συγκεκριμένα, απαιτούν μία μόνο πρόσβαση στο δίσκο. Η μέθοδος της αναζήτησης προέρχεται από τη διαίσθηση ότι δύο παρόμοιες χρονοσειρές αντιπροσωπεύονται συχνά από την ίδια λέξη *iSAX*. Δεδομένης αυτής της υπόθεσης, το κατά προσέγγιση αποτέλεσμα επιτυγχάνεται προσπαθώντας να βρούμε έναν τερματικό κόμβο στο ευρετήριο ίδιο με το ερώτημα (αυτό διότι όπως έχουμε πει ο τερματικός κόμβος αποθηκεύει λέξεις *iSAX* με την υψηλότερη

πληθικότητα). Αυτό γίνεται διασχίζοντας το ευρετήριο σύμφωνα με τις πολιτικές διαχωρισμού και ταιριάζοντας τις αναπαραστάσεις *iSAX* σε κάθε εσωτερικό κόμβο. Επειδή το ευρετήριο είναι ιεραρχικό και χωρίς επικάλυψη, εάν υπάρχει τέτοιος τερματικός κόμβος εντοπίζεται αμέσως. Όταν φτάσουμε σε αυτόν τον τερματικό κόμβο, το αρχείο ευρετηρίου που δείχνει ο κόμβος ανακτάται και επιστρέφεται. Αυτό το αρχείο θα περιέχει τουλάχιστον *1* και το πολύ *th* χρονοσειρές σε αυτό. Μια κύρια διαδοχική σάρωση μνήμης σε αυτές τις χρονοσειρές δίνει το αποτέλεσμα της κατά προσέγγιση αναζήτησης.

2.3.6 Ακριβής αναζήτηση

Για να βελτιώσουμε την ταχύτητα αναζήτησης, χρησιμοποιούμε έναν συνδυασμό κατά προσέγγιση αναζήτησης και κατώτερης οριακής απόστασης για τη μείωση του χώρου αναζήτησης. Ο αλγόριθμος ξεκινά με τη λήψη μιας κατά προσέγγιση καλύτερης μέχρι στιγμής (BSF- Best so far) απάντησης, χρησιμοποιώντας την κατά προσέγγιση αναζήτηση όπως περιγράφεται προηγουμένως (γραμμές 2-3). Μόλις ληφθεί μια βασική BSF, δημιουργείται μια ουρά προτεραιότητας για την εξέταση κόμβων των οποίων η απόσταση είναι δυνητικά μικρότερη από την BSF. Αυτή η ουρά προτεραιότητας αρχικοποιείται πρώτα με τον κόμβο ρίζας (γραμμή 6). Η απόσταση που χρησιμοποιείται για την ταξινόμηση της προτεραιότητας των ουρών των κόμβων υπολογίζεται χρησιμοποιώντας την *MINDIST_PAA_iSAX*.

Δεδομένης της αναπαράστασης *PAA (T_{PAA}* μιας χρονοσειράς *T)* και της αναπαράστασης *iSAX (S_{iSAX}* μιας χρονοσειράς *S)*, τέτοια έτσι ώστε *$|T_{PAA}| = |S_{iSAX}| = w$, $|T| = |S| = n$*, με δύο άνω και κάτω σημεία διακοπής κάθε φορά $\beta_L, \beta_U, (\beta_L < \beta_U)$ ορίζουμε την απόσταση κατώτερου ορίου ως [6]:

$$MINDIST\_PAA\_iSAX(T_{PAA}, S_{iSAX}) = DS\sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}\begin{cases}(\beta_{L_i} - T_{PAA_i})^2 & if \beta_{L_i} > T_{PAA_i} \\ (\beta_{U_i} - T_{PAA_i})^2 & if \beta_{U_i} < T_{PAA_i} \\ 0\,, διαφορετικά\end{cases}},$$

Μια σημαντική ιδιότητα της συνάρτησης απόστασης που χρησιμοποιούμε είναι η ιδιότητα του κατώτερου ορίου. Εάν η BSF απόσταση είναι μικρότερη ή ίση από την ελάχιστη απόσταση από το ερώτημα σε έναν κόμβο, μπορούμε να απορρίψουμε τον κόμβο και όλους τους απογόνους χωρίς να εξετάσουμε το περιεχόμενό τους ή να εισαγάγουμε ψευδείς απορρίψεις. Στη συνέχεια, ο αλγόριθμος εξάγει επανειλημμένα τον κόμβο με τη μικρότερη τιμή απόστασης από την ουρά προτεραιότητας, τερματίζοντας όταν είτε η ουρά προτεραιότητας αδειάσει είτε πληρείται κάποια συνθήκη πρώιμου τερματισμού. Ο πρόωρος τερματισμός συμβαίνει όταν η απόσταση κατώτερου ορίου που υπολογίζουμε ισούται ή υπερβαίνει την BSF απόσταση. Αυτό σημαίνει ότι οι υπόλοιπες καταχωρήσεις στην ουρά δεν μπορούν να χαρακτηριστούν ως ο κοντινότερος γείτονας και μπορούν να απορριφθούν.

Εάν δεν πληρείται η συνθήκη πρώιμου τερματισμού (γραμμή 10), ο κόμβος αξιολογείται περαιτέρω. Στην περίπτωση που ο κόμβος είναι τερματικός κόμβος,

παίρνουμε το αρχείο ευρετηρίου από το δίσκο και υπολογίζουμε την απόσταση από το ερώτημα σε κάθε εισαγωγή στο αρχείο ευρετηρίου, καταγράφοντας την ελάχιστη απόσταση (γραμμή 14). Εάν αυτή η απόσταση είναι μικρότερη από την BSF, ενημερώνουμε την BSF (γραμμές 16-17).

Στην περίπτωση που ο κόμβος είναι ένας εσωτερικός κόμβος ή κόμβος ρίζας, οι άμεσοι απόγονοί του εισάγονται στην ουρά προτεραιότητας (γραμμές 20-23). Ο αλγόριθμος στη συνέχεια επαναλαμβάνεται εξάγοντας τον επόμενο ελάχιστο κόμβο από την ουρά προτεραιότητας.

```
1   Function [IndexFile] = ExactSearch(ts)
2   BSF.IndexFile = ApproximateSearch(ts)
3   BSF.dist = IndexFileDist(ts, BSF.IndexFile)
4
5   PriorityQueue pq
6   pq.Add(root)
7
8   while !pq.IsEmpty
9       min = pq.ExtractMin()
10      if min.dist >= BSF.dist
11          break
12      endif
13      if min is terminal
14          tmp = IndexFileDist(ts, min.IndexFile)
15          if BSF.dist > tmp
16              BSF.dist = tmp
17              BSF.IndexFile = min.IndexFile
18          endif
19      elseif min is internal or root
20          foreach node in min.children
21              node.dist = MINDIST_PAA_iSAX(ts,node.iSAX)
22              pq.Add(node)
23          end
24      endif
25  end
26  return BSF.IndexFile
```

Πίνακας 3.2: Ο ψευδοκώδικας για την επιτάχυνση της ακριβής αναζήτησης χρησιμοποιώντας την κατά προσέγγιση αναζήτηση και συναρτήσεις κατώτερης οριακής απόστασης [6].

2.4 Η μέθοδος iSAX 2.0 (Indexable Symbolic Aggregate ApproXimation 2.0)

Το πρόβλημα της ευρετηρίασης και εξόρυξης χρονοσειρών βρίσκεται στο επίκεντρο της προσοχής τα τελευταία χρόνια, καθώς οι βάσεις δεδομένων χρονοσειρών αυξάνονται καθημερινά σε μέγεθος εξαιτίας της συνεχής ανανέωσής τους. Για όλες τις προσπάθειες ευρετηρίασης και εξόρυξης χρονοσειρών μεγάλης κλίμακας, είναι η χρονική πολυπλοκότητα δημιουργίας του ευρετηρίου που παραμένει το πιο σημαντικό σημείο συμφόρησης. Έχουν προταθεί διάφορες μέθοδοι αναπαραστάσεων οι οποίες προσπαθούν να λύσουν το πρόβλημα, ωστόσο η χρονική διάρκεια δημιουργίας του ευρετηρίου παραμένει αρκετά υψηλή.

Η μέθοδος iSAX 2.0 [7] (Indexable Symbolic Aggregate ApproXimation 2.0) προσπαθεί να επιλύσει αυτό το πρόβλημα προτείνοντας μία τεχνική μαζικής φόρτωσης για ένα ευρετήριο χρονοσειρών. Αυτό επιτυγχάνεται επεκτείνοντας τη συμβολική αναπαράσταση *iSAX (indexable Symbolic Aggregate ApproXimation)*. Προτείνονται οι ακόλουθες δύο συμπληρωματικές τεχνικές για τη βελτίωση της μεθόδου *iSAX* .

- Ένας νέος αλγόριθμος για μαζική φόρτωση χρονοσειρών που μειώνει σημαντικά το συνολικό αριθμό προσβάσεων στο δίσκο.
- Μια νέα πολιτική διαχωρισμού για τους εσωτερικούς κόμβους του ευρετηρίου μειώνοντας έτσι περαιτέρω το χρόνο δημιουργίας του ευρετηρίου.

Η βασική ιδέα του αλγορίθμου είναι ότι αντί να αναπτυχθεί ολόκληρο το ευρετήριο ταυτόχρονα, εστιάζονται οι προσπάθειες στην οικοδόμηση των ξεχωριστών υπό-δέντρων του ευρετηρίου, ένα κάθε φορά. Με αυτόν τον τρόπο, ελαχιστοποιείται αποτελεσματικά ο αριθμός των λειτουργιών διαχωρισμού κόμβων και γίνονται πιο αποδοτικές όλες οι προσβάσεις στο δίσκο. Αυτό επιτυγχάνεται ομαδοποιώντας αποτελεσματικά τις χρονοσειρές που θα καταλήξουν σε ένα συγκεκριμένο υπό-δέντρο του ευρετηρίου και στη συνέχεια επεξεργάζονται όλες μαζί.

Ο αλγόριθμος χρησιμοποιεί δύο κύρια επίπεδα μνήμης (buffer), δηλαδή το First Buffer Layer (FBL) και το Leaf Buffer Layer (LBL). Το FBL αντιστοιχεί στο πρώτο επίπεδο κόμβων της *iSAX 2.0* που παραμένει σταθερό καθ' όλη τη διάρκεια της δημιουργίας του ευρετηρίου. Το LBL αντιστοιχεί σε τερματικούς κόμβους. Ο ρόλος των buffer στο FBL είναι να δημιουργήσουν (συγκεντρωτικά) κλάσεις χρονοσειρών που θα καταλήξουν στο ίδιο υπό-δέντρο *iSAX 2.0*, «ριζωμένο» σε ένα από τα άμεσα παιδιά (child) της ρίζας. Οι buffer στο FBL δεν έχουν περιορισμό στο μέγεθός τους και μεγαλώνουν μέχρι να καταλάβουν όλη τη διαθέσιμη κύρια μνήμη. Αντίθετα, τα buffer στο LBL χρησιμοποιούνται για τη συλλογή όλων των χρονοσειρών των τερματικών κόμβων και τη μεταφορά τους στο δίσκο. Αυτά τα buffer έχουν το ίδιο μέγεθος με το αυτό των τερματικών κόμβων.

2.4.1 Παρουσίαση της iSAX 2.0

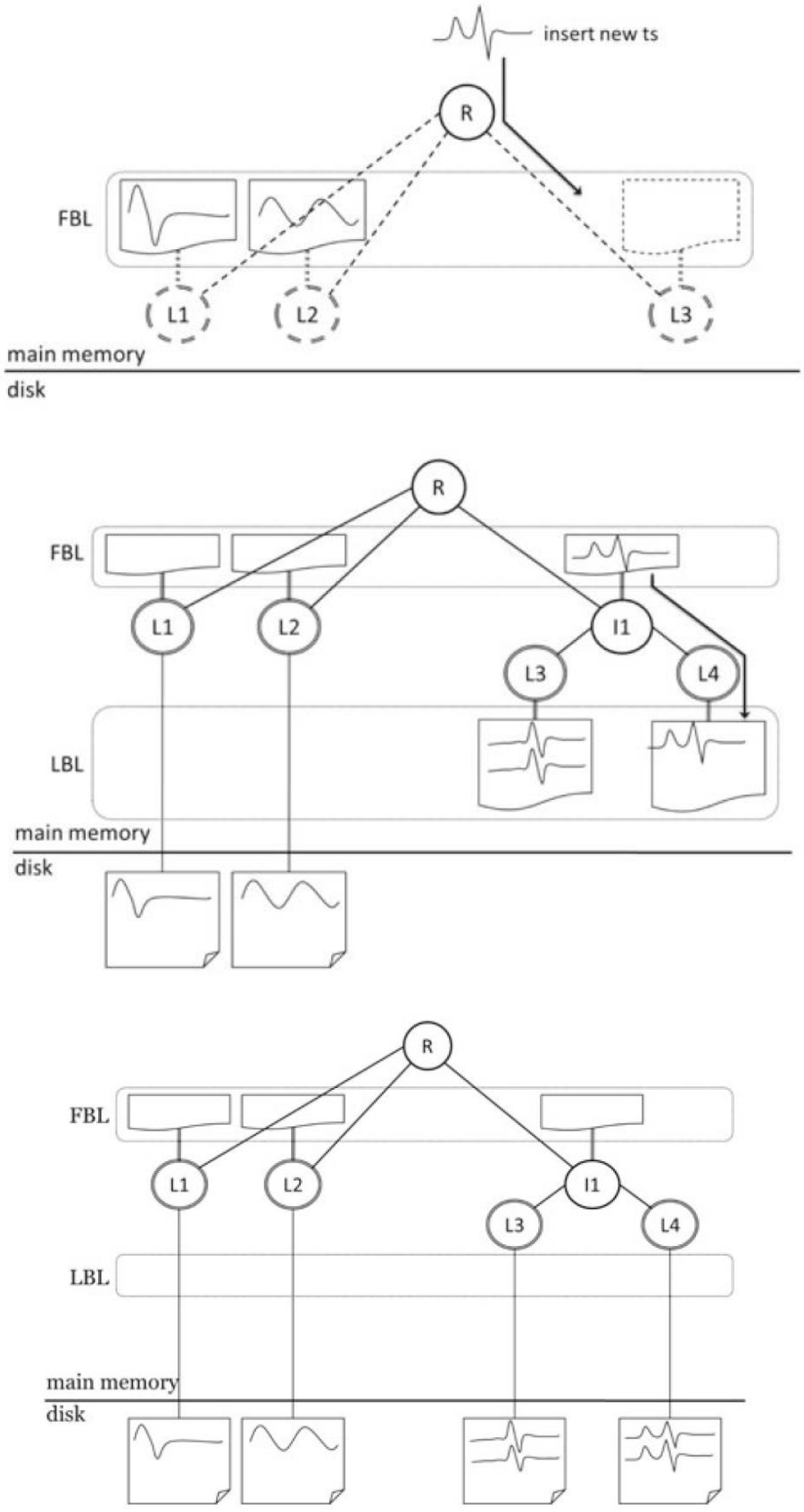
Ο αλγόριθμος για μαζική φόρτωση χρονοσειρών λειτουργεί σε δύο φάσεις, οι οποίες εναλλάσσονται έως ότου ολοκληρωθεί η ευρετηρίαση ολόκληρου του συνόλου δεδομένων. Ο ψευδοκώδικας του αλγορίθμου απεικονίζεται στο Σχήμα 4.2.

Φάση 1: Ο αλγόριθμος διαβάζει τις χρονοσειρές και τις εισάγει στο αντίστοιχο buffer στο FBL (γραμμές 4-16). Αυτή η φάση συνεχίζεται έως ότου η κύρια μνήμη είναι σχεδόν πλήρης. Χρειάζεται μια μικρή ποσότητα επιπλέον μνήμης για εκχώρηση νέων κόμβων κατά τη Φάση 2. Ωστόσο, αυτό απαιτείται μόνο για την έναρξη της πρώτης επανάληψης του βρόχου στις γραμμές 12–16, καθώς κάθε επανάληψη απελευθερώνει μνήμη.

Στο τέλος της Φάσης 1, έχουν συγκεντρωθεί οι χρονοσειρές στις προσωρινές μνήμες FBL. Αυτή η κατάσταση απεικονίζεται στο Σχήμα 4.1 (πάνω γράφημα). Σημειώστε ότι παρόλο που έχουν δημιουργηθεί ορισμένα buffer FBL (σύμφωνα με τις χρονοσειρές που έχουν υποβληθεί σε επεξεργασία μέχρι τώρα), οι αντίστοιχοι τερματικοί κόμβοι (φύλλων) *L1*, *L2*, και *L3*, του ευρετηρίου δεν έχουν δημιουργηθεί ακόμη.

Φάση 2: Ο αλγόριθμος προχωρά μετακινώντας τις χρονοσειρές που περιέχονται σε κάθε buffer FBL στα κατάλληλα buffer LBL. Για κάθε buffer FBL, ο αλγόριθμος διαβάζει τις χρονοσειρές και δημιουργεί όλους τους απαραίτητους εσωτερικούς (γραμμές 25-32) και τερματικούς (γραμμές 35-38) κόμβους *iSAX 2.0*. Δημιουργεί βασικά ολόκληρο το υπό-δέντρο με ρίζες στον κόμβο που αντιστοιχεί σε αυτό το buffer FBL. Για παράδειγμα, στο Σχήμα 4.1 (μεσαίο γράφημα), αδειάζοντας το δεξί FBL buffer, δημιουργείται το δευτερεύον δέντρο με ρίζες στον εσωτερικό κόμβο *I1*. Ο αλγόριθμος δημιουργεί επίσης για κάθε τερματικό κόμβο ένα αντίστοιχο buffer LBL (γραμμή 37). Όταν όλες οι χρονοσειρές ενός συγκεκριμένου buffer FBL έχουν μετακινηθεί προς τα κάτω στα αντίστοιχα buffer LBL, ο αλγόριθμος μεταφέρει τις χρονοσειρές στο δίσκο ελευθερώνοντας μνήμη από τα buffer LBL (γραμμή 15). Παρατηρήστε ότι στο Σχήμα 4.1 (κάτω γράφημα), τα buffer LBL για τα υπό-δέντρα που έχουν τις ρίζες τους σε κόμβους *L1* και *L2* έχουν ήδη ελευθερωθεί από τη μνήμη και όλη η διαθέσιμη μνήμη μπορεί να αφιερωθεί στα buffer LBL του *I1* υπόδεντρου.

Στο τέλος της Φάσης 2 του αλγορίθμου, όλες οι χρονοσειρές από τα buffer FBL έχουν μετακινηθεί κάτω από το δέντρο στους κατάλληλους τερματικούς κόμβους (δημιουργώντας νέους εάν είναι απαραίτητο) και στα buffer LBL και στη συνέχεια από τα buffer LBL στο δίσκο (Σχήμα 4.1 - κάτω γράφημα). Αυτό σημαίνει ότι όλα τα buffer (τόσο FBL όσο και LBL) είναι κενά και μπορεί να συνεχιστεί η επεξεργασία του συνόλου δεδομένων, επιστρέφοντας στη Φάση 1 του αλγορίθμου. Αυτή η διαδικασία συνεχίζεται έως ότου ευρετηριαστεί ολόκληρο το σύνολο δεδομένων.



Σχήμα 4.1: Ο αλγόριθμος μαζικής φόρτωσης. Η Φάση 1 συμπληρώνει τα buffer FBL με χρονοσειρές έως ότου η κύρια μνήμη είναι πλήρης (πάνω γράφημα). Φάση 2, επεξεργασία υποδέντρου με ρίζα στον κόμβο *I1*, τα υπό-δέντρα που έχουν ρίζες στους κόμβους *L1* και *L2* έχουν ήδη μεταφερθεί στο δίσκο (μεσαίο γράφημα). Μετά τη φάση 2 (κάτω γράφημα) [7].

```
1.  FBL[]        // πίνακας (array) με FBL buffers
2.  LBL[]        // πίνακας (array) με LBL buffers
3.  Function Bulk_Insert()
4.  while (περισσότερες χρονοσειρές στο ευρετήριο)
5.    ts_new = επόμενη χρονοσειρά προς ευρετηρίαση
6.    iSAX_word = αναπαράσταση iSAX της ts_new
7.    if (η κύρια μνήμη είναι ακόμη διαθέσιμη)
8.      if (κανένα FBL buffer δεν περιέχει iSAX_word)
9.        δημιούργησε καινούργιο FBL buffer που να αντιστοιχίζεται στην iSAX_word
10.     πρόσθεσε ts_new στο FBL[]
11.   else if (η κύρια μνήμη είναι πλήρης)
12.     for each buf στο FBL[]
13.       for each ts σε buf
14.         «κάλεσε» τη συνάρτηση Insert(ts)
15.       μετέφερε τα LBL buffers που δημιουργήθηκαν κατά τη διάρκεια αυτής της εισόδου
16.       διέγραψε από τη μνήμη αυτά τα LBL buffers
```

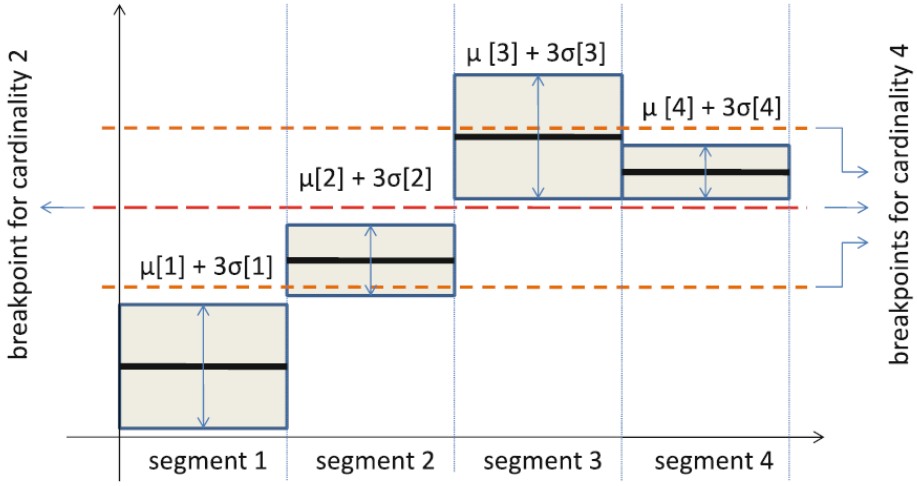
```
17. Function Insert(ts_new)
18. iSAX_word = αναπαράσταση iSAX της ts_new
19. if (το υπόδεντρο που αντιστοιχίζεται στη iSAX_word υπάρχει)
      // ο τρέχων κόμβος έχει ένα κόμβο child για τη λήψη της ts_new
20.   n = κόμβος προορισμού της ts_new
21. // η διαδρομή της ts_new καθοδικά στο δέντρο
22.   if (n είναι τερματικός κόμβος)
23.     if (n δεν είναι πλήρης)   // ο κόμβος δε χρειάζεται να διαχωριστεί
24.       πρόσθεσε ts_new μέσα στο LBL[n]    // το buffer που αντιστοιχίζεται στο n
25.     else    // ο κόμβος χρειάζεται να διαχωριστεί
26.       for each ts σε n   // διάβασε όλες τις χρονοσειρές στο n (από το δίσκο)
27.         πρόσθεσε ts στο LBL[n]
28.       n_new = νέος εσωτερικός κόμβος
29.       for each ts σε LBL[n]
30.         n_new.Instert(ts)
31.       n_new.Instert(ts_new)
32.       διέγραψε (remove) το n   // όλες οι χρονοσειρές μεταφέρθηκαν κάτω από το n_new
33.   else if (n είναι εσωτερικός κόμβος)
34.     n.Instert(ts_new)
35. else   // ο τρέχων κόμβος δεν έχει ένα κόμβο child για τη λήψη της ts_new
36.   n_new_leaf = new leaf node
37.   δημιούργησε νέο LBL buffer που να αντιστοιχίζεται σε n_new_leaf
38.   πρόσθεσε ts_new σε αυτό το νέο LBL buffer
```

Σχήμα 4.2: Ψευδοκώδικας για τον αλγόριθμο μαζικής φόρτωσης [7].

2.4.2 Πολιτική διαχωρισμού κόμβων

Είναι προφανές ότι το μέγεθος μιας δομής ευρετηρίου επηρεάζει το χρόνο δημιουργίας του ευρετηρίου. Η πολιτική διαχωρισμού κόμβων που προτείνεται λαμβάνει ενημερωμένες αποφάσεις με βάση τη γνώση της κατανομής των δεδομένων που είναι αποθηκευμένα σε κάθε κόμβο. Κατά τον διαχωρισμό ενός κόμβου, πρέπει να διανέμονται οι χρονοσειρές από αυτόν τον κόμβο εξίσου στους δύο νέους κόμβους. Ο αλγόριθμός που αναφέρεται, εξετάζει για κάθε τμήμα τις κατανομές των συμβόλων με την υψηλότερη πληθικότητα στις σχετικές χρονοσειρές. Στη συνέχεια, διαχωρίζει τον κόμβο στο τμήμα για το οποίο η κατανομή των συμβόλων υποδεικνύει ότι υπάρχει μεγάλη πιθανότητα να χωριστεί η χρονοσειρά και στους δύο νέους κόμβους.

Εξετάστε το παράδειγμα που απεικονίζεται στο Σχήμα 4.3. Έστω μια *iSAX* λέξη μήκους (w) τέσσερα (δηλαδή αριθμός τμημάτων) και θα θέλαμε να χωριστεί ένας κόμβος του οποίου η πληθικότητα (α) είναι 2 για όλα τα τμήματα.
Με άλλα λόγια ψάχνουμε εκείνο το τμήμα (δηλαδή εκείνο το γράμμα) στο οποίο θα αυξήσουμε τη πληθικότητα και θα γίνει ο διαχωρισμός σε δύο νέα θυγατρικά αρχεία (child αρχεία).
Για κάθε τμήμα, υπολογίζεται το εύρος $\mu \pm 3\sigma$ των αντίστοιχων συμβόλων.



Σχήμα 4.3: Παράδειγμα πολιτικής διαχωρισμού κόμβων [7].

Παρατηρούμε ότι αυτό το εύρος για το τμήμα 1 βρίσκεται εντελώς κάτω από το χαμηλότερο σημείο διακοπής της πληθικότητας 4. Μόνο οι περιοχές των τμημάτων 2 και 3 διασχίζουν κάποιο σημείο διακοπής της πληθικότητας 4. Μεταξύ αυτών των δύο, ο αλγόριθμος θα επιλέξει να διαχωριστεί στο τμήμα 3, επειδή η τιμή μ βρίσκεται πιο κοντά σε ένα σημείο διακοπής από εκείνη του τμήματος 2. Αυτή είναι μια ένδειξη ότι με μεγάλη πιθανότητα, μερικές από τις χρονοσειρές στον κόμβο που θα διαχωριστεί θα καταλήξουν στον νέο κόμβο που αντιπροσωπεύει την περιοχή πάνω από το σημείο διακοπής, ενώ οι υπόλοιπες θα μεταβούν στον δεύτερο νέο κόμβο, επιτυγχάνοντας έτσι μια ισορροπημένη διαίρεση.

Ο ψευδοκώδικας για τον αλγόριθμο διαχωρισμού κόμβων φαίνεται στο Σχήμα 4.4 (καλείται κάθε φορά που πρέπει να δημιουργηθεί ένας νέος εσωτερικός κόμβος: γραμμές 29 και 37 στο Σχήμα 4.2). Ο αλγόριθμος ξεκινά υπολογίζοντας για κάθε τμήμα τη μέση τιμή μ και τυπική απόκλιση σ της κατανομής των συμβόλων σε όλες τις χρονοσειρές στον κόμβο που θα διαχωριστεί (γραμμές 2–3). Στη συνέχεια, ο αλγόριθμος πρέπει να επιλέξει ένα από τα τμήματα για τον διαχωρισμό του κόμβου. Για κάθε τμήμα, ο αλγόριθμος εξετάζει εάν το εύρος που δημιουργείται από το $\mu \pm 3\sigma$ διασχίζει οποιοδήποτε από τα σημεία διακοπής *iSAX* της αμέσως υψηλότερης πληθικότητας (γραμμές 6-10). Μεταξύ των τμημάτων για τα οποία ισχύει αυτό, ο αλγόριθμος επιλέγει εκείνο για το οποίο η τιμή μ βρίσκεται πιο κοντά σε ένα σημείο διακοπής (γραμμές 9–10).

```
1.  Function Split()
2.  mean[] = ComputeSympolMean()    // χρησιμοποιώντας την υψηλότερη αναπαράσταση iSAX
3.  stdev[] = ComputeSympolStDev()    // ήδη υπολογισμένα κατά τη διάρκεια των εισαγωγών
4.  segmentToSplit = κανένα
5.  for each τμήμα s στη λέξη iSAX
6.     b = getBreakPoints(s)    // σημείο διακοπής s με αυξημένη πληθικότητα
7.     if (b μεταξύ mean[s] ± 3stdev[s] )    // το τμήμα s είναι υποψήφιο για διαχωρισμό
8.        if (mean[s] είναι πιο κοντά στο b από ότι στο segmentToSplit)
9.           segmentToSplit = s
10. segmentToSplit.IncreaseCardinality()
```

Σχήμα 4.4: Ψευδοκώδικας για τον αλγόριθμο διαχωρισμού κόμβων [7].

2.4.3 Η μέθοδος iSAX 2.0 clustered

Το πρόβλημα της μείωσης του χρόνου κατασκευής του ευρετηρίου αντιμετωπίζεται ως ένα βαθμό με τη μέθοδο iSAX 2.0. Ωστόσο, η μέθοδος iSAX 2.0 clustered συνεισφέρει σε αυτήν την κατεύθυνση, μειώνοντας ακόμη περισσότερο το χρόνο κατασκευής του ευρετηρίου. Η μέθοδος αυτή προσπαθεί να ελαχιστοποιήσει τις τυχαίες προσβάσεις δίσκου, διασφαλίζοντας ότι οι χρονοσειρές που καταλήγουν στον ίδιο τερματικό κόμβο του ευρετηρίου αποθηκεύονται προσωρινά στις ίδιες (ή παρακείμενες) σελίδες δίσκου.

Η μέθοδος *iSAX 2.0 clustered* [8] είναι μια απλή επέκταση του βασικού αλγορίθμου, ο οποίος χρησιμοποιεί ένα επιπλέον σύνολο σελίδων δίσκου για την ομαδοποίηση (clustering) των αρχικών χρονοσειρών που πιθανώς θα καταλήξουν στον ίδιο τερματικό κόμβο. Αυτές οι σελίδες δίσκου καλούνται FBL Clusters.

Σε κάθε επανάληψη, η πρώτη φάση είναι η ίδια όπως στην *iSAX 2.0* , οι αρχικές χρονοσειρές και οι προσεγγίσεις τους αποθηκεύονται στα buffer FBL. Κατά τη δεύτερη φάση, δημιουργείται το ευρετήριο *iSAX* όπως είδαμε, ωστόσο στις προσωρινές μνήμες LBL και στις σελίδες δίσκου τερματικών κόμβων μεταφέρονται **μόνο** οι προσεγγίσεις των χρονοσειρών (και όχι οι χρονοσειρές όπως προηγουμένως).Οι αρχικές χρονοσειρές μεταφέρονται στο δίσκο απευθείας από τα buffer FBL έως τα FBL Clusters. Αυτή η κατάσταση απεικονίζεται στο Σχήμα 4.5.

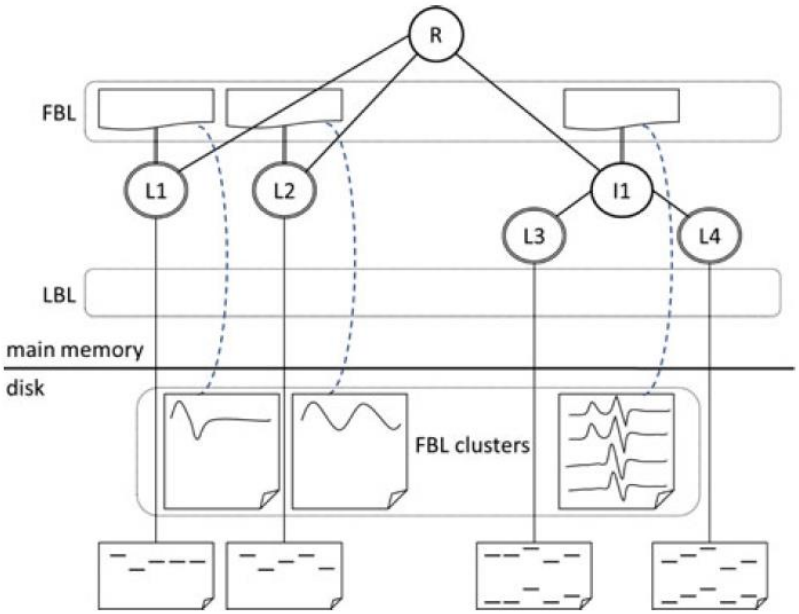
Σημειώνουμε ότι υπάρχει μία-προς-μία απεικόνιση μεταξύ των buffer FBL και των FBL Clusters. Αυτή η απεικόνιση παραμένει σταθερή καθ' όλη τη διαδικασία μαζικής φόρτωσης. Επομένως, στις επόμενες επαναλήψεις, οι αρχικές χρονοσειρές των ίδιων FBL buffers θα μεταφερθούν στα ίδια FBL Clusters.

Στο τέλος της διαδικασίας, όταν έχουν υποβληθεί σε επεξεργασία όλες οι χρονοσειρές, πρέπει να μεταφερθούν τα δεδομένα των αρχικών χρονοσειρών από τα FBL Clusters στους σωστούς τερματικούς κόμβους. Αυτό γίνεται διαβάζοντας τα FBL Clusters ένα-προς-ένα και στη συνέχεια μετακινούνται οι αρχικές χρονοσειρές στους αντίστοιχους τερματικούς κόμβους. Όλοι οι απαραίτητοι τερματικοί κόμβοι είναι ήδη στη θέση τους και δεν χρειάζονται διαχωρισμοί. Το δέντρο *iSAX* έχει φτάσει στην τελική του μορφή (βάσει των προσεγγίσεων).

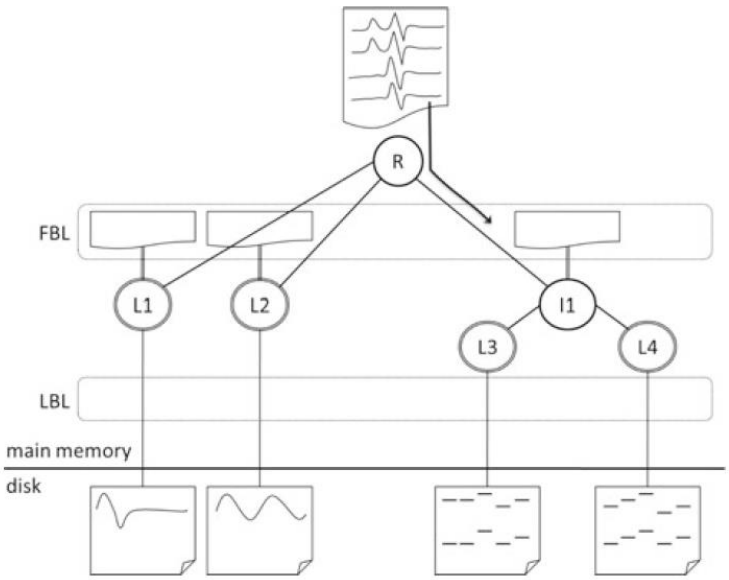
Κατά τη διάρκεια αυτής της διαδικασίας, χρησιμοποιούνται για άλλη μια φορά τα LBL buffers προκειμένου να διασφαλιστεί ότι το μεγαλύτερο μέρος του δίσκου περιλαμβάνει διαδοχικές προσβάσεις. Το σχήμα 4.6 δείχνει την κατάσταση όπου έχουν ήδη επεξεργαστεί τα FBL Clusters που αντιστοιχούν στα δύο πρώτα FBL buffers, *L1* και *L2*, και είμαστε έτοιμοι να επεξεργαστούμε το τελευταίο FBL Cluster, που αντιστοιχεί στον κόμβο *I1*.

Έτσι με αυτόν τον τρόπο μειώνεται ο χρόνος κατασκευής του ευρετηρίου επειδή έχουμε περισσότερη ελεύθερη μνήμη, καθώς μεταφέρονται στις προσωρινές μνήμες LBL και στις σελίδες δίσκου τερματικών κόμβων μόνο οι προσεγγίσεις

των χρονοσειρών. Όσον αφορά τις χρονοσειρές που κατέληξαν στον ίδιο τερματικό κόμβο του ευρετηρίου αποθηκεύονται προσωρινά στις ίδιες (ή παρακείμενες) σελίδες δίσκου (FBL Clusters) μειώνοντας έτσι και άλλο το χρόνο κατασκευής του ευρετηρίου.



Σχήμα 4.5: Το iSAX 2.0 clustered ευρετήριο στο τέλος της πρώτης επανάληψης. Τα FBL Clusters περιέχουν τα δεδομένα αρχικών χρονοσειρών που αντιστοιχούν σε καθένα από τα buffer FBL, ενώ οι σελίδες δίσκου τερματικών κόμβων περιέχουν μόνο τις προσεγγίσεις των χρονοσειρών σε κάθε τερματικό κόμβο [8].



Σχήμα 4.6: Μετακίνηση των αρχικών χρονοσειρών στην τελική τους θέση για την iSAX 2.0 Clustered: τα τελευταία FBL Clusters (που αντιστοιχούν στον κόμβο I1) βρίσκονται υπό επεξεργασία. Οι δύο σελίδες δίσκων τερματικών κόμβων στα αριστερά περιέχουν ήδη τα δεδομένα των αρχικών χρονοσειρών, καθώς και τις προσεγγίσεις τους. Οι δύο σελίδες δίσκου στα δεξιά περιέχουν μόνο τις προσεγγίσεις [8].

Κεφάλαιο 3 – hyperSAX, μία Πολυδιάστατη Επέκταση της iSAX

Οι συμβολικές αναπαραστάσεις χρονοσειρών που αναφέρθηκαν στο προηγούμενο Κεφάλαιο, καθώς και αρκετές άλλες αναπαραστάσεις, χρησιμοποιούνται για την εξόρυξη και ευρετηρίαση δεδομένων που αφορούν μονοδιάστατες χρονοσειρές. Η πολύ μεγάλη ανάπτυξη στον επιστημονικό αλλά και στον επιχειρηματικό κλάδο τις τελευταίες δεκαετίες έχει οδηγήσει στη δημιουργία και αποθήκευση χρονοσειρών πολλών διαστάσεων. Πολυδιάστατες χρονοσειρές είναι χρονοσειρές για τις οποίες σε κάθε χρονικό σημείο συλλέγονται μετρήσεις, όχι για ένα χαρακτηριστικό της, αλλά για πολλά. Ένα παράδειγμα πολυδιάστατης χρονοσειράς είναι τα μετεωρολογικά δεδομένα, καθώς σε κάθε χρονική στιγμή πραγματοποιείται συλλογή πολλών χαρακτηριστικών που μας ενδιαφέρει (όπως θερμοκρασία, υγρασία και ταχύτητα του ανέμου).

Η ευρετηρίαση και η αναζήτηση ομοιότητας σε πολυδιάστατες χρονοσειρές αποτελεί ένα πρόβλημα, καθώς δεν υπάρχουν πολλοί αλγόριθμοι εξόρυξης πολυδιάστατων χρονοσειρών. Η μέθοδος hyperSAX [9] αποτελεί μία συμβολική αναπαράσταση η οποία προσπαθεί να επιλύσει αυτό το πρόβλημα. Ωστόσο η μέθοδος hyperSAX έχει προταθεί για ευρετηρίαση πολυδιάστατων δεδομένων και όχι απαραίτητα χρονοσειρών. Σε αυτήν την εργασία προσπαθούμε να εφαρμόσουμε τον αλγόριθμο της μεθόδου hyperSAX, επεκτείνοντας τη χρησιμότητά της για πολυδιάστατες χρονοσειρές.

Η αναπαράστασης hyperSAX χρησιμοποιεί δυναμικά μήκη λέξεων σε αντίθεση με τη μέθοδο iSAX η οποία χρησιμοποιεί σταθερό μήκος λέξης. Ενώ μία λέξη iSAX είναι ένα **απλό διάνυσμα** γραμμάτων iSAX , μια λέξη hyperSAX είναι **μια δομή δέντρου** με τα γράμματα iSAX ως τερματικούς κόμβους.

3.1 Παρουσίαση της hyperSAX

Αρχικά θα ορίσουμε το γράμμα hyperSAX : Τα γράμματα μιας λέξης hyperSAX είναι τα ίδια με τα γράμματα μιας λέξης *iSAX* και αποτελούνται από ένα σύμβολο s και μια πληθικότητα α, όπου 0 ≤ s <α. Θα γράφουμε το σύμβολο με την πληθικότητα ως εκθέτη s^{α}, για παράδειγμα 4^{8}, 2^{4}.
Στη συνέχεια θα ορίσουμε τη λέξη hyperSAX, η οποία αποτελείται από γράμματα. Η λέξη hyperSAX έχει την εξής μορφή :

$$\{h_1, \dots, h_w\}_d$$

Με *w* είναι το μήκος της λέξης hyperSAX, δηλαδή το μέγεθος της διάσπασης και *d* η διάσταση. Κάθε h_i στην λέξη hyperSAX αντιπροσωπεύει ένα θυγατρικό κόμβο (child node) και μπορεί να είναι ένα γράμμα s^{α}. Ένα παράδειγμα μίας λέξης είναι :

$$\{0^2, 1^2\}_2$$

Ωστόσο με τη μέθοδο hyperSAX, μία λέξη hyperSAX μπορεί να περιέχει μία άλλη λέξη hyperSAX (έναν εσωτερικό κόμβο). Σε αυτήν την περίπτωση το h_i μπορεί να αντιπροσωπεύει μία λέξη hyperSAX. Για παράδειγμα :

$$\{0^2\{1^2 0^2\}_1\}_2$$

Χρήσιμος είναι επίσης και ο τύπος λέξης hyperSAX, ο οποίος περιλαμβάνει τη δομή του δέντρου καθώς και την πληθικότητα των χαρακτήρων των γραμμάτων στους τερματικούς κόμβους. Στο προηγούμενο παράδειγμα ο τύπος λέξης hyperSAX είναι :

$$\langle 2, \langle 2,2 \rangle_1 \rangle_2$$

3.2 Δημιουργία λέξης hyperSAX

Αρχικά οι χρονοσειρές κανονικοποιούνται (με μέσο όρο 0 και τυπική απόκλιση 1) και στη συνέχεια διαχωρίζονται σύμφωνα με τον αρχικό διαχωρισμό που προκαλεί ο βασικός τύπος λέξης hyperSAX (Βλέπε Ενότητα 3.4). Υπολογίζεται η μέση τιμή σε κάθε φύλλο (τερματικό κόμβο) και στη συνέχεια, οι μέσες τιμές μετατρέπονται σε σύμβολα. Γενικότερα, προσδιορίζεται η συνάρτηση διαχωρισμού ***P (A, w, d),*** που χωρίζει τον πολυδιάστατο πίνακα ***A*** σε «τμήματα» **w** ίσου μεγέθους στην *d* διάσταση, δηλαδή η συνάρτηση ***P (A, w, d)*** επιστρέφει μια λίστα με πολυδιάστατους πίνακες μήκους ***w***. Οι μέσες τιμές μετατρέπονται σε γράμματα χρησιμοποιώντας τη λίστα των σημείων διακοπής. Είναι σημαντικό να σημειώσουμε ότι ο ***A*** μπορεί να είναι ένας υπό-πίνακας, δηλαδή μπορεί να είναι το αποτέλεσμα μιας κατάτμησης.

Η μέση τιμή $\bar{p}$ ενός τμήματος p είναι [9]:

$$\bar{p} = \sum_{i_1=1}^{m_1} \dots \sum_{i_n=1}^{m_n} \frac{p_{i_1,\dots i_n}}{\prod_{d=1}^{n} m_d}$$

όπου *n* είναι ο αριθμός των διαστάσεων και m_d είναι το μέγεθος του p στην διάσταση *d*. Για τη δημιουργία ενός γράμματος πληθικότητας α από μία μέση τιμή $\bar{p}$, χρειαζόμαστε μία λίστα αποτελούμενη από $\alpha - 1$ ταξινομημένα σημεία διακοπής $(\beta_1, \dots, \beta_{\alpha-1})$, όπου $\beta_1 < \beta_2 < \dots < \beta_{\alpha-1}$. Ψάχνουμε το χαμηλότερο σημείο διακοπής μεγαλύτερο από το $\bar{p}$ (δηλαδή $\bar{p} < \beta_i$) και επιστρέφουμε το σύμβολο $i - 1$. Εάν κανένα από τα σημεία διακοπής δεν είναι μεγαλύτερο από το $\bar{p}$ ($\bar{p} \geq \beta_{\alpha-1}$) επιστρέφουμε το σύμβολο $\alpha - 1$. Ο τρόπος υπολογισμού της λίστας των σημείων διακοπής μπορεί να γίνει με τη χρήση της αντίστροφης αθροιστικής συνάρτησης κατανομή έτσι ώστε $\beta_i = CDF^{-1}(N(\mu, \sigma), \frac{i}{\alpha})$, όπου $N(\mu, \sigma), \frac{i}{\alpha}$ είναι μία κανονική κατανομή με μέσος μ και τυπική απόκλιση σ.

βi \ *a*	2	3	4	5	6	7	8
β_1	0.00	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2		0.43	0.00	-0.25	-0.43	-0.57	-0.67
β_3			0.67	0.25	0.00	-0.18	-0.32
β_4				0.84	0.43	0.18	0.00
β_5					0.97	0.57	0.32
β_6						1.07	0.67
β_7							1.15

Πίνακας 3.1: Ένας πίνακας αναζήτησης που περιέχει τα «σημεία διακοπής» που διαιρούν μια κατανομή Gauss σε έναν αυθαίρετο αριθμό (από 2 έως 8) ίσης πιθανότητας περιοχών [5].

3.3 Μέτρο Απόστασης

Για δύο n-διάστατους πίνακες A και B, μεγέθους m_d στη διάσταση *d*, θα χρησιμοποιήσουμε την εξής απόσταση [9]:

$$Dist(A,B) = \sqrt{\sum_{i_1=1}^{m_1} \dots \sum_{i_n=1}^{m_n} (A_{i_1,\dots,i_n} - B_{i_1,\dots,i_n})^2}$$

Σημειώνεται ότι ισχύει η ιδιότητα του κατώτερου ορίου στην συνάρτηση για τον υπολογισμό των αποστάσεων μεταξύ λέξεων hyperSAX και πολυδιάστατων πινάκων.

Το μικρότερο μέρος μιας λέξη hyperSAX είναι ένα γράμμα, οπότε παρόμοια με τη μέθοδο *iSAX*, καθορίζουμε άνω και κάτω σημεία διακοπής, b_L και b_U για ένα γράμμα s^α (σύμβολο s και πληθικότητας α) [9]:

$$\beta_L = \begin{cases} \beta_s\,, \alpha\nu\ s > 0 \\ -\infty\,, \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \quad , \quad \beta_U = \begin{cases} \beta_{s+1}\,, \alpha\nu\ s < a-1 \\ \infty\,, \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Τα σημεία διακοπής b_L και b_U χρησιμοποιούνται τώρα για τον υπολογισμό του *DistLetter,* που είναι μια χαμηλότερη οριακή απόσταση μεταξύ ενός γράμματος $\boldsymbol{s^\alpha}$ και ενός πολυδιάστατου πίνακα $\boldsymbol{A}$, με μέση τιμή $\bar{\boldsymbol{A}}$ [9]:

$$DistLetter(A, s^\alpha) = \begin{cases} \beta_L - \bar{A}\ , \varepsilon\acute{\alpha}\nu\ \ \beta_L > \bar{A} \\ \bar{A} - \beta_U\ , \varepsilon\acute{\alpha}\nu\ \beta_U < \bar{A} \\ 0\ , \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Στη συνέχεια, ορίζουμε μια αναδρομική συνάρτηση *D* που διασχίζει τη δομή του δέντρου της λέξης hyperSAX και αθροίζει τις αποστάσεις μεταξύ των γραμμάτων και των τμημάτων του πολυδιάστατου πίνακα που αντιπροσωπεύουν [9]:

$$D(A,h) = \frac{1}{w}\sum_{i=1}^{w} \begin{cases} D(p_i, h_i), & \alpha\nu\ h_i\ \varepsilon\acute{\iota}\nu\alpha\iota\ \acute{\varepsilon}\nu\alpha\ split \\ (DistLetter(p_i, h_i))^2, & \alpha\nu\ h_i\ \varepsilon\acute{\iota}\nu\alpha\iota\ \acute{\varepsilon}\nu\alpha\ \gamma\rho\acute{\alpha}\mu\mu\alpha \end{cases}$$

Σε αυτή τη συνάρτηση, η $\boldsymbol{h}$ είναι μια λέξη hyperSAX μήκους *w*, διάστασης *d* και αποτελείται από λέξεις hyperSAX h_1 μέχρι h_w .Το p_i είναι το *i* τμήμα που προκύπτει από τον κατακερματισμό του πολυδιάστατου πίνακα *A* με τη συνάρτηση *P (A, w, d).* Η κατώτερη οριακή απόσταση μεταξύ ενός *n-διάστατου* πίνακα *A* και μιας λέξης hyperSAX $\boldsymbol{h}$ ορίζεται ως εξής [9]:

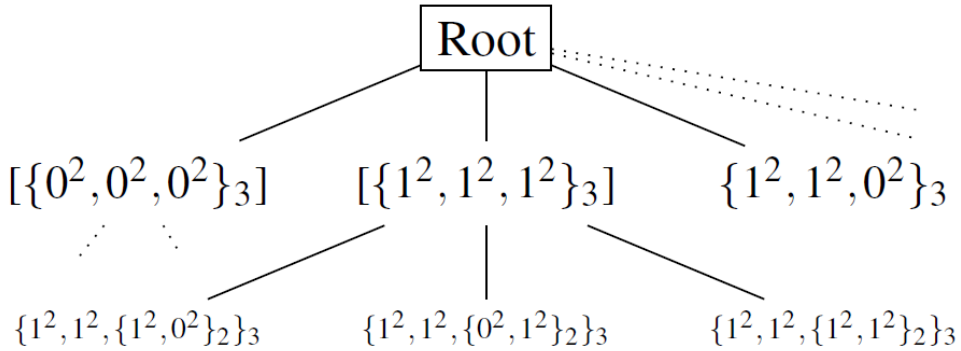
$$DistHyperword(A,h) = \sqrt{D(A,h) * \prod_{d=1}^{n} m_d}$$

υπενθυμίζοντας ότι m_d είναι το μέγεθος του πίνακα *A* στην διάσταση *d*.

3.4 Το Ευρετήριο hyperSAX

Το ευρετήριο είναι παρόμοιο με αυτό της *iSAX* με τρεις διαφορετικούς τύπους κόμβων: **Κόμβο ρίζας, Εσωτερικό Κόμβο και Τερματικό κόμβο**. Η διαφορά είναι ότι οι κόμβοι αποτελούνται από λέξεις hyperSAX αντί για iSAX λέξεις. Όταν χτίζουμε ένα ευρετήριο hyperSAX, παρέχουμε έναν βασικό τύπο

λέξης hyperSAX (υποδηλώνεται ως *b),* που θα υπαγορεύσει τις αρχικές διασπάσεις στον **κόμβο ρίζας**, και ένα όριο (υποδηλώνεται ως *th)* για το μέγιστο αριθμό πολυδιάστατων πινάκων σε κάθε **τερματικό κόμβο**. Το όριο είναι απαραίτητο για τη δημιουργία του ευρετηρίου, καθώς ρυθμίζει τη δημιουργία νέων **εσωτερικών κόμβων**, δηλαδή όταν ένας τερματικός κόμβος έχει φτάσει το μέγιστο της χωρητικότητά του, δημιουργείται ένας νέος εσωτερικός κόμβος στη θέση του. Το Σχήμα 3.1 δείχνει μια απεικόνιση του ευρετηρίου hyperSAX με βασικό τύπο λέξης hyperSAX $b = \langle 2,2,2 \rangle_3$.



Σχήμα 3.1: Εικόνα του ευρετηρίου hyperSAX. Οι κόμβοι που περιβάλλονται από τετράγωνες αγκύλες είναι εσωτερικοί κόμβοι [9].

Η κατασκευή του ευρετηρίου ξεκινά εισάγοντας πολυδιάστατους πίνακες σε έναν τερματικό κόμβο έως ότου ο τερματικός κόμβος φτάσει το όριο *th* , δηλαδή περιέχει *th* πολυδιάστατους πίνακες και θέλουμε να εισάγουμε και άλλον ένα οπότε και έχουμε $k = th + 1$ πίνακες που πρέπει να διανείμουμε. Μετατρέπουμε τον τερματικό κόμβο σε έναν εσωτερικό κόμβο λαμβάνοντας το αντίστοιχο τύπο λέξης hyperSAX και είτε αυξάνουμε την πληθικότητα ενός γράμματος (π.χ. $\langle 4,4 \rangle_1 \rightarrow \langle 4,8 \rangle_1$) είτε χωρίζουμε ένα γράμμα σε μια νέα λέξη hyperSAX (π.χ. $\langle 4,4 \rangle_1 \rightarrow \langle 4, \langle 4,4 \rangle_1 \rangle_1$). Η αύξηση της πληθικότητας προκαλεί ένα διαχωρισμό πληθικότητας (cardinality split) και πολλαπλασιάζει πάντα την πληθικότητα του επιλεγμένου γράμματος με 2. Ο διαχωρισμός ενός γράμματος σε μια νέα λέξη hyperSAX προκαλεί ένα διαχωρισμό διακριτοποίησης (discretization split) και πάντα οδηγεί σε μια λέξη hyperSAX μήκους 2.

3.5 Διαχωρισμός πληθικότητας (cardinality split)

Για να μετρήσουμε τη χρησιμότητα ενός διαχωρισμού πληθικότητας, εξετάζουμε τις μέσες τιμές όλων των k υπό-πινάκων. Εάν αυτές οι μέσες τιμές απέχουν μεταξύ τους, είναι λογικό να αυξηθεί η πληθικότητα , επειδή οι πολυδιάστατοι πίνακες που περιέχονται στον τερματικό κόμβο είναι πιθανότερο να καταλήξουν σε διαφορετικά κλαδιά του ευρετηρίου. Υπολογίζουμε λοιπόν τη χρησιμότητα για ένα διαχωρισμό πληθικότητας ως εξής [9]:

$$C_{utility} = \sum_{i-1}^{k} \left| \bar{S}_i - \frac{1}{k} \sum_{j=1}^{k} \bar{S}_j \right|$$

Όπου $\bar{S}_i$ είναι η μέση τιμή του πολυδιάστατου υπό-πίνακα S_i.

3.6 Διαχωρισμός διακριτοποίησης (discretization split)

Για τον υπολογισμό της χρησιμότητας ενός διαχωρισμού διακριτοποίησης αρχικά κανονικοποιούμε τους k υπό-πίνακες με μέσο 0, δημιουργώντας έτσι $N_i = Norm\,(S_i)$ για όλα τα $0 < i \leq k$, όπου η $Norm(S_i)$ αφαιρεί $\bar{S}_i$ από όλα τα στοιχεία του S_i. Για να υπολογίσουμε την εναπομένουσα διακύμανση, πρέπει πρώτα να δημιουργήσουμε έναν πολυδιάστατο πίνακα M ίδιου μεγέθους με καθένα από τα κανονικοποιημένους υπό-πίνακες, που χρησιμεύει ως ένας μέσος όρος όλων των k κανονικοποιημένων υπό- πινάκων. Αυτός ο μέσος πολυδιάστατος πίνακας για κάθε στοιχείο M είναι [9]:

$$M_{i_1,\dots i_n} = \sum_{j=1}^{k} \frac{N_{j\,i_1,\dots i_n}}{k}$$

Όπου $N_{j\,i_1,\dots i_n}$ είναι το στοιχείο στη θέση $i_1, \dots, i_n$του κανονικοποιημένου υπό-πίνακα N_j.

Μπορούμε να προσδιορίσουμε τη $D_{utility}$ ως τη μέση διαφορά μεταξύ των M και των k κανονικοποιημένων υπό- πινάκων [9]:

$$D_{utility} = \sum_{i_1=1}^{m_1} \dots \sum_{i_n=1}^{m_n} \sum_{j=1}^{k} \frac{\left|N_{j\,i_1,\dots i_n} - M_{i_1,\dots i_n}\right|}{\prod_{d=1}^{n} m_d}$$

όπου n είναι ο αριθμός των διαστάσεων και m_j είναι το μήκος του M στην διάσταση j. Όσον αφορά τη διάσταση διαχωρισμού διακριτοποίησης, αξιολογείται κάθε φορά η $D_{utility}$ σε καθένα από τα δύο προκύπτοντα διαμερίσματα και επιλέγεται η διάσταση εκείνη με τη χαμηλότερη διαφορά μεταξύ των χρησιμοτήτων.

3.7 Σύγκριση Χρησιμοτήτων

Για να διαλέξουμε τελικά έναν από τους δύο διαχωρισμούς (πληθικότητας ή διακριτοποίησης) πρέπει να τους συγκρίνουμε και να διαλέξουμε αυτό με τη μεγαλύτερη χρησιμότητα. Ωστόσο τα δύο μέτρα δεν μπορούν να συγκριθούν άμεσα, καθώς το εύρος των δύο χρησιμοτήτων είναι διαφορετικό. Για να γίνουν αυτά συγκρίσιμα, μετατρέπουμε το εύρος του $C_{utility}$ ώστε να είναι ίδιο με το εύρος του $D_{utility}$, τέτοιο ώστε $C'_{utility} = 0{,}5 * C_{utility} * \alpha$.

3.8 Κατά προσέγγιση Αναζήτηση

Όταν πραγματοποιείται μια κατά προσέγγιση αναζήτηση χρονοσειράς, το ερώτημα μετατρέπεται σε λέξη hyperSAX και το ευρετήριο διασχίζεται με την ίδια πολιτική διαχωρισμού με την εισαγωγή και ταιριάζοντας τις αναπαραστάσεις hyperSAX σε κάθε εσωτερικό κόμβο . Όταν επιτευχθεί ένας αντίστοιχος κόμβος τερματικού, η καλύτερη αντιστοίχιση μπορεί να βρεθεί χρησιμοποιώντας τη διαδοχική σάρωση το πολύ th πολυδιάστατων πινάκων σε αυτόν τον κόμβο. Σε περίπτωση που δεν βρεθεί αντίστοιχος τερματικός κόμβος, επιλέγεται το πρώτο «child» του εσωτερικού κόμβου, μετά το οποίο συνεχίζει κατεβαίνοντας το δέντρο μέχρι να σταματήσει σε έναν τερματικό κόμβο.

3.9 Ακριβής αναζήτηση

Μια ακριβής αναζήτηση είναι απαραίτητη μόνο αν θέλουμε πάντα το καλύτερο δυνατό αποτέλεσμα από μια αναζήτηση, το οποίο απαιτεί μια διαδοχική σάρωση πολλών τερματικών κόμβων. Ξεκινάμε λαμβάνοντας το αποτέλεσμα μιας κατά προσέγγιση αναζήτησης (ένας τερματικός κόμβος), ο οποίος θα λειτουργεί ως το αρχικό καλύτερο-μέχρι-τώρα αποτέλεσμα (Best-So-Far). Αρχικοποιούμε επίσης μια ουρά προτεραιότητας των κόμβων και τις αποστάσεις κατώτερου ορίου, στις οποίες προσθέτουμε αρχικά τον κόμβο ρίζας (με απόσταση 0). Στη συνέχεια, διασχίζουμε το δέντρο εξάγοντας πάντα τον κόμβο με τη κατώτερη οριακή απόσταση στο ερώτημα από την ουρά προτεραιότητας. Όταν συναντάμε έναν εσωτερικό κόμβο (ή τον κόμβο ρίζας), υπολογίζονται οι κατώτερες οριακές αποστάσεις μεταξύ των «child» του και του ερωτήματος (χρησιμοποιώντας την *DistHyperword)* και στη συνέχεια όλα αυτά προστίθενται στην ουρά προτεραιότητας. Όταν συναντάμε έναν τερματικό κόμβο, πραγματοποιούμε μια διαδοχική σάρωση των πολυδιάστατων πινάκων που περιέχει αυτός ο κόμβος και ενημερώνουμε το καλύτερο μέχρι στιγμής (Best-So-Far) αποτέλεσμα αν βρούμε κάτι καλύτερο. Επειδή το μέτρο απόστασης έχει την ιδιότητα κατώτερου ορίου, μπορούμε να τερματίσουμε το βρόχο και να επιστρέψουμε το καλύτερο μέχρι στιγμής αποτέλεσμα, όταν εξάγουμε μια απόσταση από την ουρά προτεραιότητας που είναι μεγαλύτερη από την καλύτερη-μέχρι-τώρα (Best-So-Far) απόσταση. Ο Ψευδοκώδικας ακριβής αναζήτησης περιγράφεται στο Σχήμα 3.2.

```
   Data: query : A multidimensional array
   Result: The multidimensional array closest to the query
 1 bestNode = APPROXIMATESEARCH(query)
 2 bestDist = bestNode.MinimumDistanceTo(query)
 3 PriorityQueue pq
 4 pq.Add(root, 0)
 5 while pq is non-empty do
 6     (minNode, minDist) ← pq.ExtractMin()
 7     if minDist ≥ bestDist then
 8         pq.Clear()
 9     else if minNode is terminal then
10         tmp ← minNode.MinimumDistanceTo(A)
11         if bestDist > tmp then
12             bestDist ← tmp
13             bestNode ← minNode
14         end
15     else if minNode is internal or root then
16         foreach node ∈ minNode.children do
17             dist ← DistHyperword(query, node.Hyperword)
18             pq.Add(node, dist)
19         end
20     end
21 end
22 return bestNode.ClosestTo(query)
```

Σχήμα 3.2: Algorithm 1 – Ψευδοκώδικας ακριβής αναζήτησης [9].

Κεφάλαιο 4 – Εφαρμογή της μεθόδου hyperSAX

4.1 Πειράματα

Για τα πειράματά μας χρησιμοποιήθηκε η Scala μέσω της IntelliJ IDEA Community Edition 2020.2.3 x64. Τα δεδομένα προέρχονται από μία δισδιάστατη χρονοσειρά και αφορούν το κρυπτονόμισμα bitcoin με 1η διάσταση την Τιμή (Price) και 2η διάσταση τον αριθμό συναλλαγών (Volume). Τα δεδομένα είναι ωριαία και αφορούν το χρονικό διάστημα 08-10-15 13:00 έως 05-11-20 0:00.

Η βάση δεδομένων της χρονοσειράς αυτής παρέχεται δωρεάν και βρίσκεται στο:

https://www.cryptodatadownload.com/data/gemini/

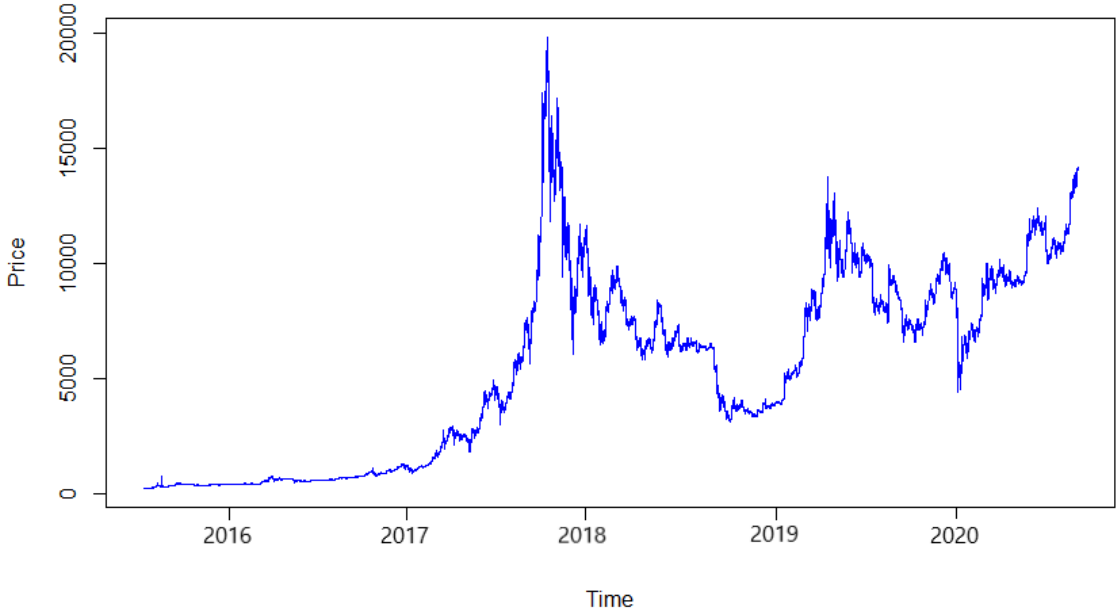
Στα δεδομένα δεν υπήρχαν ελλιπείς τιμές (missing values).

Η τιμή υπολογίστηκε ως ο μέσος όρος της τιμής ανοίγματος, της τιμής κλεισίματος, του χαμηλότερου και υψηλότερου σημείου. Η τιμή (Price) είναι μετρημένη σε USD ($).

Ο αριθμός συναλλαγών (Volume) είναι μετρημένος σε εκατομμύρια.

Αρχικά θα αναφέρουμε κάποια βασικά χαρακτηριστικά της χρονοσειράς καθώς και τα αντίστοιχα διαγράμματα.

Τιμή (Price)

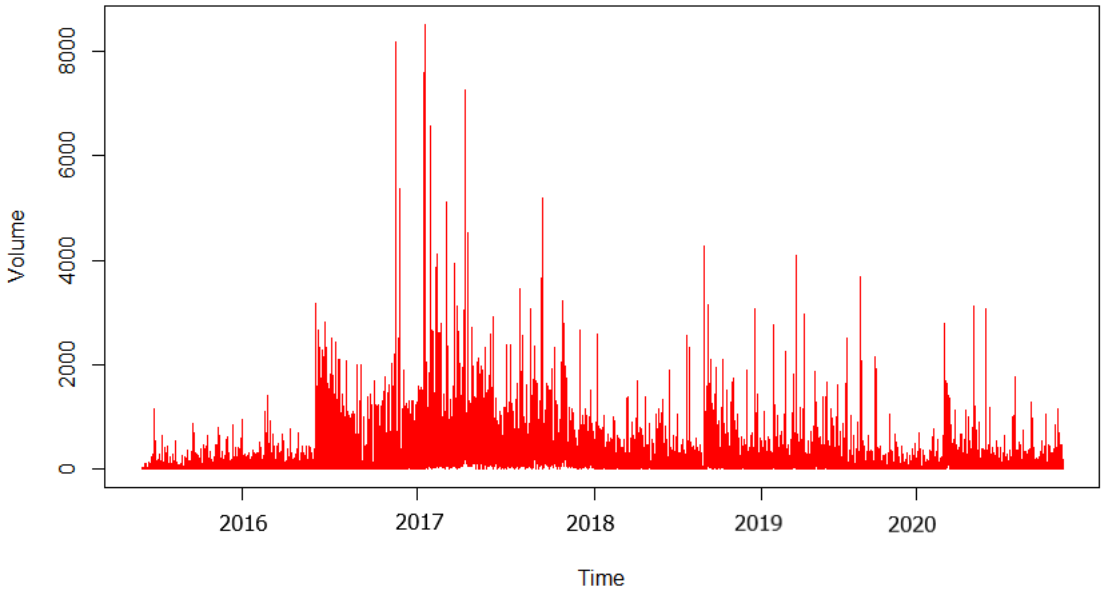


Σχήμα 4.1: Απεικόνιση της μονοδιάστατης χρονοσειράς της τιμής του bitcoin (Price).

Περιγραφικά Στατιστικά της Τιμής

Ελάχιστο	Μέγιστο	Μέση Τιμή	Διάμεσος
243.6	19834.2	5420	5790.5

Αριθμός συναλλαγών (volume)

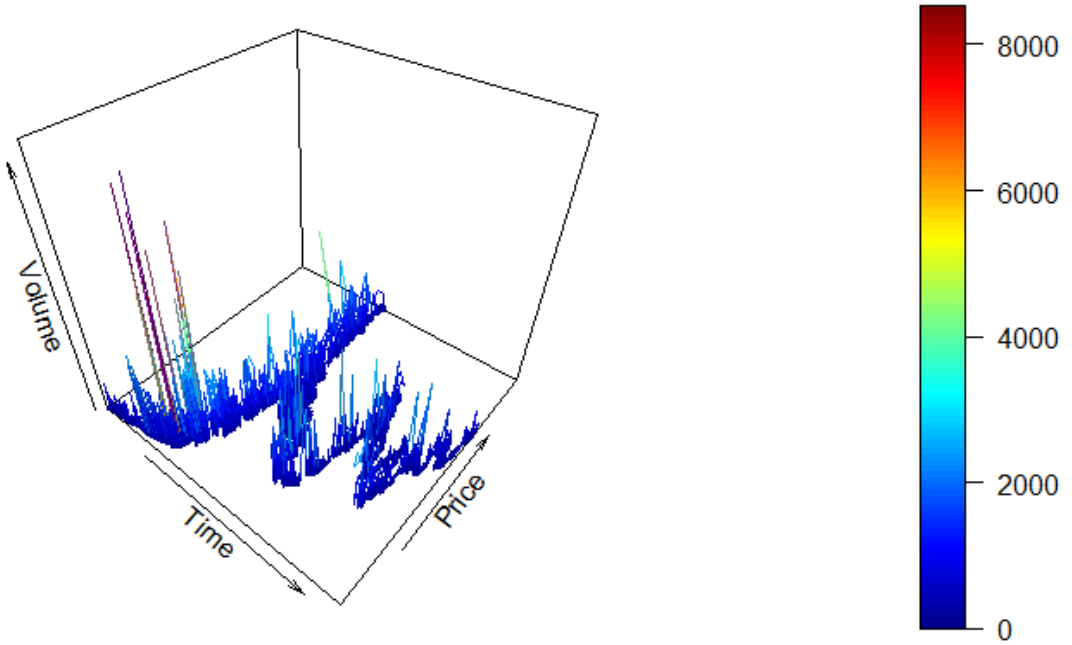


Σχήμα 4.2: Απεικόνιση της μονοδιάστατης χρονοσειράς του αριθμού συναλλαγών του bitcoin (Volume).

Περιγραφικά Στατιστικά του αριθμού των συναλλαγών

Ελάχιστο	Μέγιστο	Μέση Τιμή	Διάμεσος
0	8526.75	149.75	59.99

Δισδιάστατη απεικόνιση της χρονοσειράς



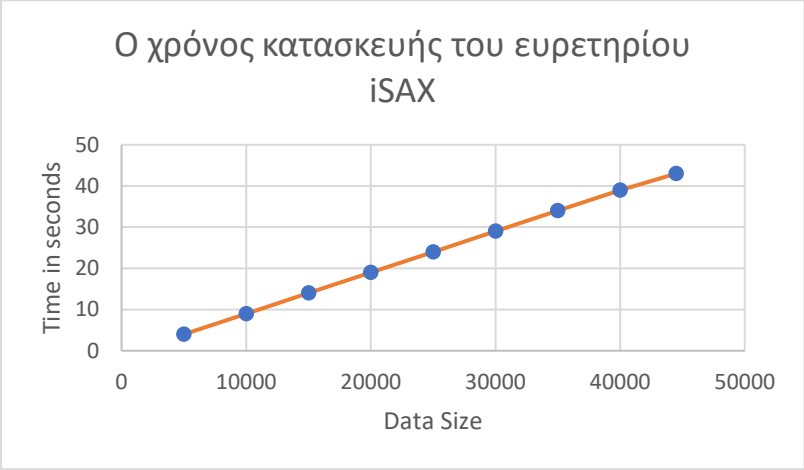
Σχήμα 4.3: Απεικόνιση της δισδιάστατης χρονοσειράς του bitcoin (Price +Volume).

4.2 Χρόνος κατασκευής του ευρετηρίου

Στο πρώτο πείραμα, για το χρόνο κατασκευής του ευρετηρίου χρησιμοποιήθηκε αρχικά η τεχνική iSAX στη μονοδιάστατη χρονοσειρά Price του Bitcoin με βασική πληθικότητα b=4, μήκος λέξης w=5 και όριο th=50.

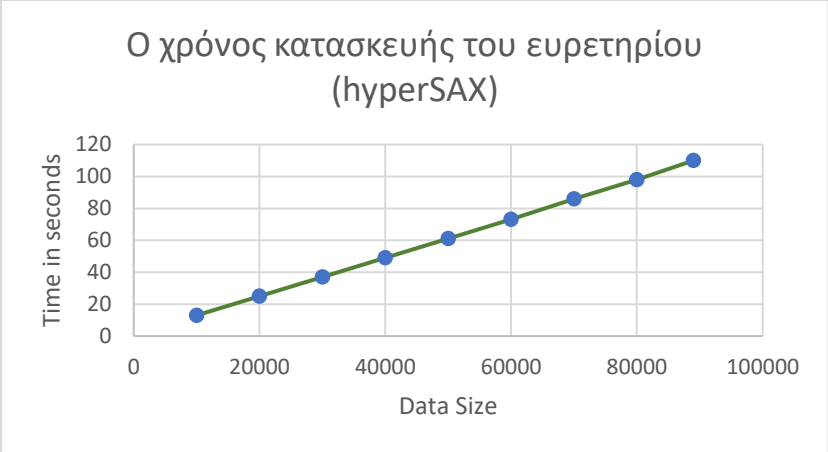
Στη συνέχεια χρησιμοποιήθηκε η τεχνική hyperSAX για τη δισδιάστατη χρονοσειρά Price vs Volume με βασικό τύπο λέξης $b = \langle 4,4,4,4,4 \rangle_2$ (δηλαδή πληθικότητα 4, μήκος λέξης 5) με διάσταση d=2 και όριο th=50.

Ο λόγος που επιλέξαμε μήκος λέξης 5 είναι για να μπορεί να διαιρεθεί η χρονοσειρά, που έχει μήκος 44505, σε ίσα κομμάτια (44505/5=8901).



Σχήμα 4.4: Απεικόνιση του χρόνου κατασκευής του ευρετηρίου της τεχνικής iSAX.

Στο παραπάνω διάγραμμα φαίνεται πως η μέθοδος iSAX απαιτεί λιγότερο χρόνο για να κατασκευάσει το ευρετήριο και να έχει μία γραμμική αύξηση καθώς μεγαλώνει το μέγεθος των δεδομένων.



Σχήμα 4.5: Απεικόνιση του χρόνου κατασκευής του ευρετηρίου της τεχνικής hyperSAX.

Η μέθοδος HyperSAX φαίνεται να χρειάζεται περισσότερο χρόνο για να κατασκευάσει το ευρετήριό της καθώς μεγαλώνει το μέγεθος των δεδομένων. Ας μην ξεχνάμε ωστόσο ότι η μέθοδος hyperSAX προσπαθεί να ευρετηριάσει δισδιάστατες χρονοσειρές. Παρατηρούμε πως και η hyperSAX έχει και αυτή μία γραμμική αύξηση καθώς μεγαλώνει το μέγεθος των δεδομένων.

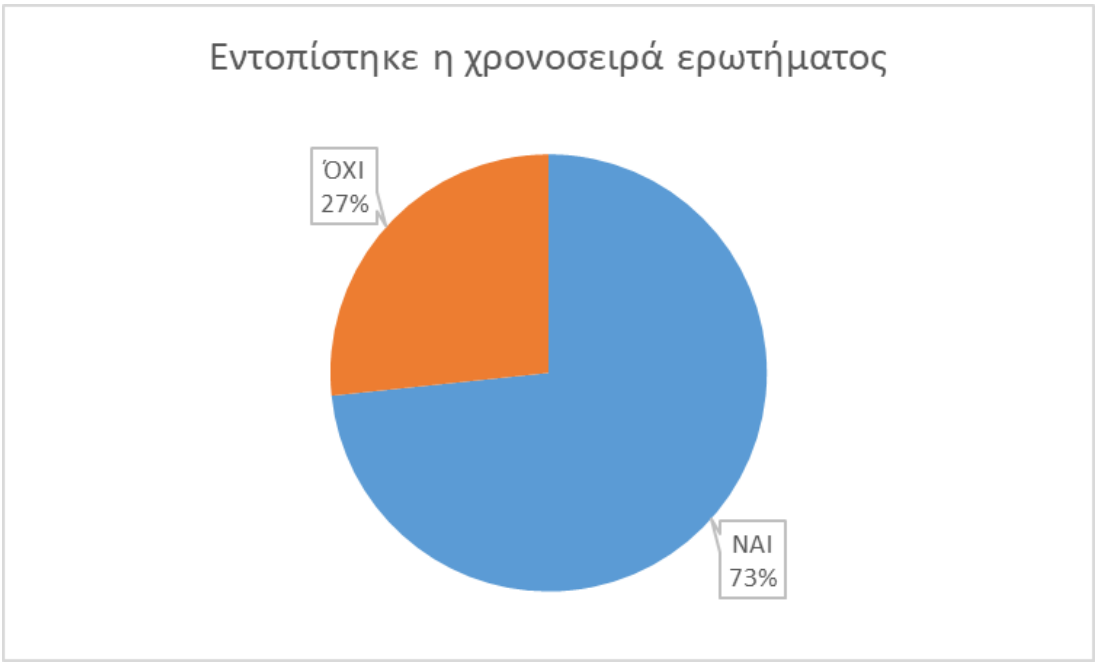
4.3 Ακρίβεια της κατά προσέγγισης αναζήτησης της hyperSAX

Στο δεύτερο πείραμα, αφού δημιουργήθηκε το ευρετήριο της hyperSAX για τη δισδιάστατη χρονοσειρά, διαλέξαμε τυχαία 30 χρονοσειρές μήκους 25.

Ο λόγος που διαλέξαμε χρονοσειρές μήκους 25 είναι για να μπορεί να διαιρεθεί από το μήκος λέξης (25/5) σε ίσα κομμάτια.

Στη συνέχεια πραγματοποιήσαμε την κατά προσέγγιση αναζήτηση με ερώτημα μία δισδιάστατη χρονοσειρά κάθε φορά. Το ερώτημα όπως έχουμε πει μετατρέπεται σε λέξη hyperSAX και το ευρετήριο διασχίζεται με την ίδια πολιτική διαχωρισμού $b = \langle 4,4,4,4,4 \rangle_2$ και th=50. Όταν επιτευχθεί ένας αντίστοιχος κόμβος τερματικού, η αντιστοίχιση μπορεί να βρεθεί χρησιμοποιώντας τη διαδοχική σάρωση σε αυτό τον κόμβο.

Στο διάγραμμα που ακολουθεί παρουσιάζονται τα αποτελέσματα.



Σχήμα 4.6: Απεικόνιση της ακρίβειας της κατά προσέγγισης Αναζήτησης

Από τα αποτελέσματα παρατηρούμε ότι η κατά προσέγγιση αναζήτηση στο 73% περίπου των περιπτώσεων καταφέρνει και βρίσκει τη χρονοσειρά ερωτήματος.

Κεφάλαιο 5 – Συμπεράσματα

Σε αυτήν την εργασία έγινε μία εκτενής αναφορά σε υπάρχουσες τεχνικές της οικογένειας iSAX. Είδαμε πως μπορούν να χρησιμοποιηθούν αυτές οι τεχνικές για εργασίες εξόρυξης, ευρετηρίασης και αναζήτησης ομοιότητας σε δεδομένα που αφορούν μονοδιάστατες χρονοσειρές. Ο χρόνος που απαιτείται για την κατασκευή του ευρετηρίου με αυτές τις τεχνικές είναι αρκετά μικρός, οπότε ενδέχεται να είναι αποτελεσματικές για πολύ μεγάλα σε μέγεθος δεδομένα.

Εξετάσαμε επίσης και την τεχνική hyperSAX και είδαμε πως μπορεί να χρησιμοποιηθεί για πολυδιάστατα δεδομένα. Πραγματοποιήθηκε μία προσπάθεια να ευρετηριαστούν δισδιάστατες χρονοσειρές μετρώντας το χρόνο κατασκευής του ευρετηρίου αλλά και την ακρίβεια για εργασίες αναζήτησης ομοιότητας. Τα αποτελέσματα φαίνονται ικανοποιητικά αλλά υπάρχει πάντα χώρος για βελτίωση. Η τεχνική hyperSAX είναι από τις λίγες τεχνικές που υπάρχουν για εργασίες εξόρυξης σε πολυδιάστατα δεδομένα και ενδεχόμενες αλλαγές στη μέθοδο μπορούν βελτιώσουν τη χρησιμότητά της ή να αποτελέσουν βάση για μελλοντικές εργασίες.

Κεφάλαιο 6 – Βιβλιογραφία

[1] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Min. Knowl. Discov.*, vol. 13, no. 3, 2006, doi: 10.1007/s10618-005-0039-x.

[2] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Min. Knowl. Discov.*, vol. 28, no. 4, 2014, doi: 10.1007/s10618-013-0322-1.

[3] M. Last, Y. Klein, and A. Kandel, "Knowledge discovery in time series databases," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 1. 2001, doi: 10.1109/3477.907576.

[4] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, 2001, doi: 10.1007/pl00011669.

[5] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Min. Knowl. Discov.*, vol. 15, no. 2, 2007, doi: 10.1007/s10618-007-0064-z.

[6] J. Shieh and E. Keogh, "ISAX: Indexing and mining terabyte sized time series," 2008, doi: 10.1145/1401890.1401966.

[7] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "iSAX 2.0: Indexing and mining one billion time series," 2010, doi: 10.1109/ICDM.2010.124.

[8] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh, "Beyond one billion time series: Indexing and mining very large time series collections with iSAX2+," *Knowl. Inf. Syst.*, vol. 39, no. 1, 2014, doi: 10.1007/s10115-012-0606-6.

[9] J. E. Gydesen, H. Haxholm, N. S. Poulsen, S. Wahl, and B. Thiesson, "HyperSAX: Fast approximate search of multidimensional data," in *ICPRAM 2015 - 4th International Conference on Pattern Recognition Applications and Methods, Proceedings*, 2015, vol. 1, doi: 10.5220/0005185201900198.

[10] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," *ACM SIGMOD Rec.*, vol. 23, no. 2, 1994, doi: 10.1145/191843.191925.

[11] H. Andrè-Jönsson and D. Z. Badal, "Using signature files for querying time-series data," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1997, vol. 1263, doi: 10.1007/3-540-63223-9_120.

[12] K. pong Chan and A. W. chee Fu, "Efficient time series matching by wavelets," *Proc. - Int. Conf. Data Eng.*, 1999, doi: 10.1109/icde.1999.754915.

[13] C. Shahabi, X. Tian, and W. Zhao, "TSA-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data," 2000, doi: 10.1109/ssdm.2000.869778.

[14] P. Geurts, "Pattern extraction for time series classification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2001, vol. 2168, doi: 10.1007/3-540-44794-6_10.

[15] J. F. Roddick, K. Hornsby, and M. Spiliopoulou, "An updated bibliography of

temporal, spatial, and spatio-temporal data mining research," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2001, vol. 2007, doi: 10.1007/3-540-45244-3_12.

[16] S. Raychaudhuri, J. M. Stuart, and R. B. Altman, "Principal components analysis to summarize microarray experiments: application to sporulation time series.," *Pac. Symp. Biocomput.*, 2000.

[17] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory networks for anomaly detection in time series," 2015.

[18] J. M. Hellerstein, E. Koutsoupias, and C. H. Papadimitriou, "On the analysis of indexing schemes," 1997, doi: 10.1145/263661.263688.

[19] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *ACM SIGMOD Rec.*, vol. 14, no. 2, 1984, doi: 10.1145/971697.602266.

[20] K. Chakrabarti and S. Mehrotra, "Hybrid tree: an index structure for high dimensional feature spaces," *Proc. - Int. Conf. Data Eng.*, 1999, doi: 10.1109/icde.1999.754960.

[21] I. Assent, R. Krieger, F. Afschari, and T. Seidl, "The TS-tree: Efficient time series search and retrieval," 2008, doi: 10.1145/1353343.1353376.