ΕΠΙΤΑΧΥΝΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΕΥΡΕΣΗΣ
ΤΗΣ ΕΚΤΙΜΗΣΗΣ ΤΗΣ ΔΙΑΣΤΑΣΗΣ ΣΥΣΧΕΤΙΣΗΣ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από την

ΚΑΤΕΡΙΝΑ ΜΠΑΚΑ

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ-ΕΦΑΡΜΟΓΕΣ

Απρίλιος 2010

ΑΦΙΕΡΩΣΗ

Αφιερώνεται σε όλους τους δασκάλους μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ πολύ τους φίλους μου Νάρια, Νίκο, Ειρήνη, Χριστίνα, Νίκο για τη συμπαράσταση και τις συμβουλές τους.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΕΠΕΞΗΓΗΣΕΙΣ ΣΥΜΒΟΛΙΣΜΩΝ

Συμβολισμός	*Επεξήγηση*
D_2	Διάσταση συσχέτισης
r	Απόσταση
t	Χρονική καθυστέρηση
m	Διάσταση Ενσωμάτωσης
N	Πλήθος σημείων σήματος

ΠΕΡΙΛΗΨΗ

Κατερίνα Μπάκα του Δημητρίου και της Δέσποινας
MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Απρίλιος, 2010
Επιτάχυνση του Αλγόριθμου Εύρεσης της Εκτίμησης της Διάστασης Συσχέτισης
Επιβλέπων : Γεώργιος Μανής

Η εκτίμηση της διάστασης συσχέτισης είναι μία παράμετρος που σχετίζεται με τον ελάχιστο αριθμό μεταβλητών που χρειάζονται για την μοντελοποίηση και ανάλυση μη γραμμικών συστημάτων. Η μη γραμμική ανάλυση μπορεί σε γενικές γραμμές να εφαρμοστεί οπουδήποτε και επιτρέπει σε οποιονδήποτε να λάβει πληροφορίες για το μη γραμμικό σύστημα το οποίο παράγει μη γραμμικές χρονοσειρές εντοπίζοντας και ποσοτικοποιώντας χαοτικές συμπεριφορές.

Δύο είναι τα βασικά προβλήματα που υπεισέρχονται στη μέθοδο υπολογισμού της εκτίμησης της διάστασης συσχέτισης. Το πρώτο πρόβλημα αφορά τον αριθμό των διανυσμάτων (N) που απαιτούνται για να υπολογιστεί σωστά η εκτίμηση της διάστασης συσχέτισης. Το δεύτερο αφορά τη διάσταση ενσωμάτωσης m που χρησιμοποιείται για την μοντελοποίηση του συστήματος.

Οι αλγόριθμοι που είχαν προταθεί μέχρι τώρα δεν μπορούσαν να υπολογίσουν την εκτίμηση της διάστασης συσχέτισης σε πραγματικό χρόνο για μεγάλο αριθμό διανυσμάτων ($N > 80000$) και διάσταση ενσωμάτωσης $m > 2$.

Στην παρούσα εργασία εξετάζεται αλγόριθμος που χρησιμοποιείται σε κάποια άλλη μέθοδο ανάλυσης μη γραμμικών συστημάτων. Ο αλγόριθμος προσαρμόστηκε ώστε να υπολογίζει την εκτίμηση της διάστασης συσχέτισης. Ο αλγόριθμος αυτός μπορεί να υπολογίσει την εκτίμηση της διάστασης συσχέτισης (D_2), και για μεγάλα συστήματα όπου $D_2 > 6$ και $N > 80000$, αποδοτικά.

EXTENDED ABSTRACT IN ENGLISH

Baka, Aikaterini, D.
MSc, Computer Science Department, University of Ioannina, Greece. April, 2010.
Fast Computation of Correlation Dimension Estimation.
Supervisor: George Manis.

Correlation Dimension Estimation (D_2) is a widely used parameter in nonlinear time series analysis. Using methods from non linear time series analysis we can have information about non linear system which produces non linear time series.

Two are the main issues when somebody computes Correlation Dimension Estimation. The first is the number of points (N) in the nonlinear time series. Few points mean wrong computation of correlation Dimension estimation. The second is the embedding dimension, m, which is used to model our system.

In this work we study all algorithms for computing Correlation Dimension. We see that a reliable correlation Dimension Estimation can be very difficult in case of very complex systems where $D_2 > 5$ and $N > 80000$ using those algorithms.

So we require efficient algorithms and implementations. We study algorithms from a similar problem, Approximate Entropy, and we achieve to make algorithms with very low complexity.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Η εκτίμηση της διάστασης συσχέτισης είναι μία παράμετρος που σχετίζεται με τον ελάχιστο αριθμό μεταβλητών που χρειάζονται για την μοντελοποίηση και ανάλυση μη γραμμικών συστημάτων.

Για την ανάλυση χρονοσειρών που προέρχονται από μη γραμμικά συστήματα χρησιμοποιούνται μέθοδοι που προέρχονται από τη θεωρία του αιτιοκρατικού χάους [15]. Οι μέθοδοι αυτές χρησιμοποιούνταν ευρέως στις προηγούμενες τρεις δεκαετίες. Η μη γραμμική ανάλυση μπορεί σε γενικές γραμμές να εφαρμοστεί οπουδήποτε και επιτρέπει σε οποιονδήποτε να λάβει πληροφορίες για το μη γραμμικό σύστημα το οποίο παράγει τις χρονοσειρές εντοπίζοντας και ποσοτικοποιώντας χαοτικές συμπεριφορές. Αποτελούν γενικές εφαρμογές και έχουν δώσει χρήσιμα αποτελέσματα σε διάφορα πεδία όπως η φυσική και η μηχανική μέχρι η ιατρική και τα οικονομικά.

Μεταξύ των παραμέτρων που μελετώνται στις αναλύσεις χρονοσειρών που προέρχονται από μη γραμμικά συστήματα είναι και οι διαστάσεις. Στα χαοτικά συστήματα οι διαστάσεις είναι μη γραμμικές. Οι πιο διαδεδομένες διαστάσεις είναι : η διάσταση χωρητικότητας D_0, η διάσταση πληροφοριών D_1 και η διάσταση συσχέτισης D_2. Αυτές είναι συγκεκριμένες περιπτώσεις των γενικότερων διαστάσεων D_q [24].

Η εκτίμηση της διάστασης συσχέτισης (D_2) υπολογίζεται με βάση τη μέθοδο που προτάθηκε από τους Grassberger και Procaccia το 1983[14]. Η μέθοδος αυτή προέκυψε λαμβάνοντας υπόψη το θεώρημα του Takens, το οποίο επιβεβαιώνει ότι είναι δυνατό να αναδομηθεί ένα μη γραμμικό σύστημα από ένα μονοδιάστατο σύστημα.[30]. Η χρήση της μεθόδου είναι πολύ διαδεδομένη λόγω της απλότητάς της, δεδομένου ότι απαιτεί ουσιαστικά τον υπολογισμό όλων των αποστάσεων μεταξύ όλων των διανυσμάτων στη διάσταση m.

Εντούτοις, υπάρχουν μερικές πτυχές που εάν και παραμελούνται, μπορούν να καταστήσουν την εκτίμηση της διάστασης συσχέτισης (D_2) αναξιόπιστη ή ακόμα και λανθασμένη [8]. Τουλάχιστον δύο βασικά ζητήματα προκύπτουν. Το πρώτο έχει να κάνει με τον αριθμό των διανυσμάτων (N) που απαιτείται για να υπολογιστεί σωστά η εκτίμηση της D_2. Ένας μη επαρκής αριθμός διανυσμάτων μπορεί να οδηγήσει σε λαθεμένο υπολογισμό της τιμής της D_2. Πράγματι ορισμένοι επιστήμονες εξαιτίας αυτού του προβλήματος θεώρησαν ότι τα διάφορα αποτελέσματα που δημοσιεύτηκαν ειδικά κατά τη διάρκεια των πρώτων χρόνων, είναι ανακριβή.

Το δεύτερο ζήτημα αφορά τη διάσταση ενσωμάτωσης m που χρησιμοποιείται για την μοντελοποίηση του συστήματος. Η διάσταση ενσωμάτωσης διαισθητικά σημαίνει η διάσταση m στον R^m , που για $m' > m$ η απεικόνιση του μοντέλου στον R^m είναι παρόμοια. Για μια αξιόπιστη εκτίμηση του D_2, ο υπολογισμός θα πρέπει να επαναληφθεί με αύξηση στη τιμή του m στον R^m μέχρις ότου η αύξηση δεν παράγει καμία διαφοροποίηση στην εκτίμηση της D_2. Εάν δεν επιτευχθεί τέτοιος κορεσμός δεν μπορεί να εκτιμηθεί η D_2 τουλάχιστον με τον διαθέσιμο αριθμό διανυσμάτων.

Ο υπολογισμός της εκτίμησης της διάστασης συσχέτισης (D_2) είναι δύσκολος για μεσαία ή υψηλά συστήματα όπου $D_2 > 5$-6. Αυτό συμβαίνει γιατί ο υπολογισμός της σε αυτά τα συστήματα απαιτεί μεγάλο αριθμό διανυσμάτων (N) που θα χρησιμοποιηθούν και οι καλύτεροι αλγόριθμοι που υπάρχουν έως τώρα είναι O(NlogN). Οι αλγόριθμοι αυτοί είναι σε θέση να επιτύχουν τον υπολογισμό της εκτίμησης της D_2 σε επίπεδο πραγματικού χρόνου κυρίως στα συστήματα που η D_2 είναι μικρή ($1 < m < 3$) και οι σύντομες χρονοσειρές αρκούν ($N < 20000$).

Στόχος της παρούσας μεταπτυχιακής εργασίας είναι η εύρεση κάποιου αλγόριθμου που θα επιταχύνει τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης. Ο αλγόριθμος αυτός θα πρέπει να μην είναι προσεγγιστικός αλλά να βασίζεται στην ακριβή λύση του προβλήματος. Δηλαδή η εκτιμώμενη D_2 να είναι πολύ κοντά στην πραγματική τιμή D_2 του συστήματος. Επίσης θα πρέπει να μπορεί να υπολογίσει την

D_2 και για μεγάλα συστήματα όπου $D_2 > 6$, να έχει πολυπλοκότητα χρόνου και μνήμης μικρότερη από τους αλγόριθμους που έχουν προταθεί μέχρι τώρα και να είναι όσο το δυνατόν πιο απλός στην υλοποίηση.

Οι επιμέρους στόχοι περιλαμβάνουν την κατηγοριοποίηση των αλγορίθμων που έχουν εμφανιστεί για τη λύση του προβλήματος και την ανάλυση των αλγορίθμων που βασίζονται στην ακριβή λύση του προβλήματος και δεν είναι παράλληλοι. Επίσης, ένας άλλος επί μέρους στόχος είναι η σύγκριση των αλγορίθμων που βασίζονται στην ακριβή λύση για την επιβεβαίωση τόσο της πολυπλοκότητας τους σε χρόνο και μνήμη, όσο και της δυνατότητας τους να ανταποκριθούν σε πραγματικό χρόνο σε μεγάλα συστήματα.

Η εργασία περιέχει 6 κεφάλαια. Το Κεφάλαιο 1 είναι η εισαγωγή. Στο Κεφάλαιο 2 παρουσιάζεται η μέθοδος υπολογισμού της εκτίμησης της διάστασης συσχέτισης και κάποιες εφαρμογές της μεθόδου. Το Κεφάλαιο 3 είναι μια ιστορική αναδρομή των αλγορίθμων που έχουν προταθεί για το πρόβλημα που μελετάται. Στο Κεφάλαιο 4 αναλύονται οι αλγόριθμοι που είχαν προταθεί παλαιότερα. Οι αλγόριθμοι αυτοί ανήκουν στην κατηγορία των αλγορίθμων που βρίσκουν την ακριβή λύση και δεν είναι παράλληλοι. Στο Κεφάλαιο 5 αναλύονται οι νέοι αλγόριθμοι. Τέλος, το Κεφάλαιο 6 περιέχει τα πειραματικά αποτελέσματα για την επιβεβαίωση της μελέτης και στο Κεφάλαιο 6 παρατίθενται και όλα τα συμπεράσματα που προέκυψαν.

ΚΕΦΑΛΑΙΟ 2. Η ΕΚΤΙΜΗΣΗ ΤΗΣ ΔΙΑΣΤΑΣΗΣ ΣΥΣΧΕΤΙΣΗΣ

2.1. Η μέθοδος υπολογισμού της εκτίμησης της διάστασης συσχέτισης

Η μέθοδος υπολογισμού της εκτίμησης της διάστασης συσχέτισης βασίστηκε στο θεώρημα του Takens [30] το οποίο επιβεβαιώνει ότι είναι δυνατόν να επαναδομηθεί ένα σύστημα από ένα μονοδιάστατο σήμα και προτάθηκε το 1983 από τους Grassberger και Procaccia [14].

Έστω x χρονοσειρά που προέρχεται από ένα σύστημα:

$x=(x_1, x_2, x_3, x_4, ..., x_{N-1}, x_N)$, και N είναι ο συνολικός αριθμός των διανυσμάτων της χρονοσειράς. Επιλέγεται μια διάσταση ενσωμάτωσης *m* και μια χρονική καθυστέρηση *t*, όπου *m, t* θετικοί ακέραιοι και ξαναφτιάχνεται μια χρονική σειρά Υ διανυσμάτων $Y=\left\{\vec{y}_i\right\}_{i=1}^{N_m}$, όπου $\vec{y}_i=\left[x_i, x_{i+t}, x_{i+2t}, \ldots, x_{i+(m-1)t}\right]$ και $N_m=N-(m-1)t$ στον *m* Ευκλείδειο χώρο.

Το κάθε διάνυσμα που προκύπτει παριστάνει το διάνυσμα τροχιάς του *i -στου* χρονικού διανύσματος του επαναδομημένου συστήματος.

Η εκτίμηση της διάστασης συσχέτισης (D_2) ορίζεται ως εξής:

$D_2 = \lim_{N\to\infty} \frac{\log(C(m,r,t))}{\log(r)}$,όπου r είναι η απόσταση της ακτίνας στην επαναδομημένη διάσταση του συστήματος. Ο δείκτης 2 στην D_2 χρησιμοποιείται γιατί η διάσταση συσχέτισης είναι ειδική περίπτωση της γενικής διάστασης D_q ,όπου q ακέραιος.

Το $C(m,r,t)$ είναι το παράγωγο της συσχέτισης και ορίζεται:

$$C(m,r,t) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} H\left[r - \|\vec{y}_i - \vec{y}_j\|\right]$$

στο οποίο τα όμοια ζεύγη μετρώνται [29], και

$$C(m,r,t) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} H\left[r - \|\vec{y}_i - \vec{y}_j\|\right], i \neq j$$

όπου τα όμοια ζεύγη δεν μετρώνται [13] όπως και οι αποστάσεις μεταξύ των ίδιων των διανυσμάτων. Τα $\vec{y}_i$ και $\vec{y}_j$ $\in Y = \left\{\vec{y}_i\right\}_{i=1}^{N_m}$ και H είναι η Heavyside συνάρτηση:

$$H(i) = \left\{ \begin{matrix} 1, & if\ i \geq 0 \\ 0, & if\ i < 0 \end{matrix} \right\}.$$

Η Ευκλείδεια νόρμα χρησιμοποιείται στην παραπάνω εξίσωση και δηλώνει ότι η διαφορά ανάμεσα στα $\vec{y}_i$ και $\vec{y}_j$ είναι η μέγιστη διάφορα των συντεταγμένων τους:

$$\|\vec{y}_i - \vec{y}_j\| =$$

$$\left\{ |x(i) - x(j)|^2 + |x(i+t) - x(j+t)|^2 + ... + |x(i + (m-1)t) - x(j + (m-1)t)|^2 \right\}^{\frac{1}{2}}$$

Εναλλακτικά, μπορεί να χρησιμοποιηθεί η νόρμα μεγίστου:

$$\|\vec{y}_i - \vec{y}_j\| =$$

$$\max\left\{ |x(i) - x(j)|, |x(i+t) - x(j+t)|, ..., |x(i + (m-1)t) - x(j + (m-1)t)| \right\}$$

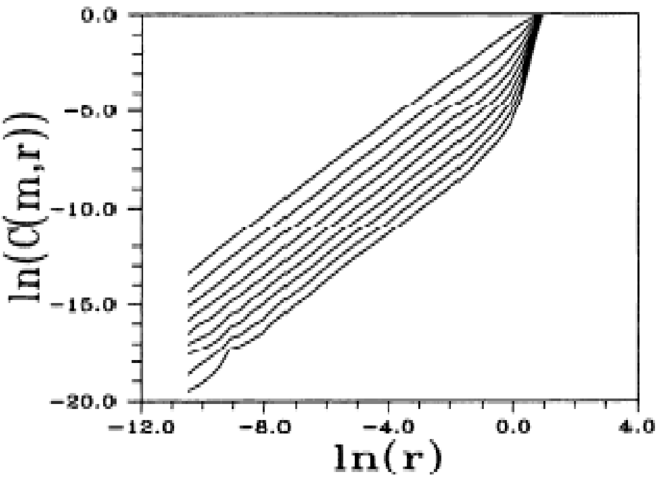
Ουσιαστικά το $C(m,r,t)$ εκφράζει το εξής: για συγκεκριμένο *m, r, t* βρες όλα τα ζεύγη $\vec{y}_i$ και $\vec{y}_j$ της επαναδομημένης χρονοσειράς Y των οποίων η απόσταση

$\|\vec{y}_i - \vec{y}_j\|$ είναι μικρότερη από r. Δηλαδή το $C(m,r,t)$ εκφράζει την πιθανότητα δύο διανύσματα στον επαναδομημένο R^m να έχουν μεταξύ τους απόσταση μικρότερη από r [7].

Σύμφωνα με τη μέθοδο πρέπει να αυξάνεται η διάσταση m του επαναδομημένου συστήματος και να υπολογίζεται το $\frac{\log(C(m,r,t))}{\log(r)}$, μέχρις ότου η αύξηση δεν παράγει καμία διαφοροποίηση στην εκτίμηση της D_2. Τότε η τιμή του $\frac{\log(C(m,r,t))}{\log(r)}$ θα συγκλίνει σε κάποια τιμή α:

$\frac{\log(C(m,r,t))}{\log(r)} \approx a \;\Rightarrow\; C \approx r^a$,όπου α είναι πραγματική τιμή, που εκφράζει τη γραμμική σύνδεση μεταξύ του $\log(C)$ και $\log(r)$ όπως φαίνεται στο Σχήμα 2.1.

Αυτό σημαίνει ότι $D_2 \approx a$ και το α είναι η εκτίμηση της διάστασης συσχέτισης της χρονοσειράς [23] .



Σχήμα 2.1 $\log(C)$ vs $\log(r)$

Σύμφωνα με τα παραπάνω, ιδιαίτερη σημασία αποκτά η επιλογή των κατάλληλων τιμών των παραμέτρων *t* και r. Η ανάλυση της χρονοσειράς εξαρτάται από το μέγεθος *(m-1)t*, το οποίο περιλαμβάνει και τη διάσταση ενσωμάτωσης *m* και την χρονική καθυστέρηση *t*. Ο περιορισμός λοιπόν της παραπάνω μεθόδου είναι το όριο του μεγέθους του παραθύρου, *(m-1)t*. Μια καλή επιλογή τιμής του παραθύρου, παρέχει καλή αναδόμηση του συστήματος στον R^m. Ωστόσο για πολύ μεγάλες τιμές του t είναι δύσκολο να αναδομηθεί το σύστημα αφού N_m = N-(m-1)t και έτσι παράγονται πολύ λίγα διανύσματα Y.

Όσον αφορά την παράμετρο t, στην παρούσα εργασία επιλέχθηκε η τιμή της να είναι t = 1 ώστε να παραχθούν όσο το δυνατόν περισσότερα διανύσματα Y. Οι τιμές της παραμέτρου r είναι $0{,}1 \le r \le 0{,}3$ της τυπικής απόκλισης της χρονικής σειράς εισόδου. Ο αριθμός των διανυσμάτων της χρονικής σειράς (N) είναι το πολύ 80000, ενώ για τις τιμές του m ισχύει $m \in [1,5]$.

2.2. Εφαρμογές της εκτίμησης της διάστασης συσχέτισης

Η εκτίμηση της διάστασης συσχέτισης, επειδή είναι μία παράμετρος που σχετίζεται με τον ελάχιστο αριθμό μεταβλητών που χρειάζονται για την μοντελοποίηση και ανάλυση μη γραμμικών συστημάτων αποτελεί γενική εφαρμογή και έχει δώσει χρήσιμα αποτελέσματα σε διάφορα πεδία όπως η φυσική και η μηχανική μέχρι η ιατρική και τα οικονομικά.

Το 1988 η εκτίμηση της διάστασης συσχέτισης, χρησιμοποιώντας μια παραλλαγή του αλγορίθμου που προτάθηκε από τους Grassberger και Procaccia [14], χρησιμοποιήθηκε από φυσικούς για την μελέτη της αναταραχής πλάσματος tokamak TFR. Τα αποτελέσματα έδειξαν ότι δοσμένου ενός τυχαία παραγμένου σήματος δοκιμής, η αναταραχή πλάσματος μπορεί να θεωρηθεί μη γραμμικό σύστημα, που κατέχει έναν χαμηλό εκθέτη συσχέτισης, δείχνοντας κατά συνέπεια την μοντελοποίηση σε μικρή διάσταση ενσωμάτωσης [1].

Αργότερα (1995), η μέθοδος εφαρμόστηκε για να υπολογιστεί η διάσταση συσχέτισης (D_2) των ηλεκτροκαρδιογραφημάτων (ECG) και να μελετηθεί η διακύμανση της τιμής της D_2 σε σήματα που προέρχονταν από υγιείς και άρρωστους ανθρώπους. 28 σήματα μελετήθηκαν και αναλύθηκαν οι αλλαγές στις τιμές της εκτίμησης της διάστασης συσχέτισης. Για την ανάλυση αυτή, η εκτίμηση της D_2 υπολογίστηκε για διαφορετικές χρονικές στιγμές μέσα στο ίδιο σήμα ECG. Παρατηρήθηκε μια μεγαλύτερη διακύμανση στη τιμή της εκτιμώμενης D_2 στα σήματα που προέρχονταν από άτομα με κάποια παθολογία, ενώ η τιμή της εκτιμώμενης D_2 φαίνεται να είναι επαναλαμβανόμενη στην περίπτωση των υγιών. Ωστόσο παρατηρήθηκε ότι δεν υπάρχει καμία σαφής ένδειξη των δηλωτικών διαφορών στις τιμές των εκτιμώμενων διαστάσεων συσχέτισης που υπολογίζονται από τα σήματα που προέρχονταν από υγιείς και άρρωστους ανθρώπους [6],[2].

Το 2000 η εκτίμηση της διάστασης συσχέτισης χρησιμοποιήθηκε για να αναλυθεί η γνωστική δραστηριότητα και το διανοητικό φορτίο μελετώντας σήματα που προέρχονταν από ανθρώπινα ηλεκτροεγκεφαλογραφήματα (EEG). Τρεις πειραματικές καταστάσεις δημιουργήθηκαν: κατάσταση βασικών γραμμών και δύο γνωστικές καταστάσεις στόχου, ένας στόχος υπολογισμού και ένας στόχος χρονικής εκτίμησης. Ο στόχος υπολογισμού υποτίθεται ότι έπρεπε να προκαλέσει ένα υψηλότερο διανοητικό φορτίο από το στόχο χρονικής εκτίμησης, ο οποίος θεωρείται ως λιγότερο σύνθετος. Αυτό ελέγχθηκε από μια υποκειμενική κλίμακα εκτίμησης. Όλοι οι όροι/καταστάσεις διέφεραν σημαντικά στο υποκειμενικό κατ' εκτίμηση φορτίο στόχου. Η διάσταση συσχέτισης εμφανίστηκε να είναι υψηλότερη και στους δύο όρους στόχου έναντι του όρου βασικών γραμμών. Μια σύγκριση των δύο στόχων έδειξε ότι η διαφορά στη διάσταση συσχέτισης μεταξύ του υπολογισμού και της χρονικής εκτίμησης ήταν επίσης σημαντική, για την υψηλότερη τιμή του υπολογισμού. Το συμπέρασμα ήταν ότι η γνωστική και η διανοητική δραστηριότητα συνδέεται με μια υψηλότερη διάσταση συσχέτισης στο EEG. Αυτό υπονοεί ότι η διάσταση συσχέτισης είναι μια ευαίσθητη παράμετρος στην ανάλυση της ηλεκτρικής δραστηριότητας του εγκεφάλου [18].

Το 2002, εφαρμόστηκε από φυσικούς για να ελέγξει την κατάσταση ενός συστήματος κατά την εκκαθάριση, εξετάζοντας την επίδραση των παραλλαγών της εκκαθάρισης χάσματος στην εκτίμηση της διάστασης συσχέτισης σε ένα φέρον σύστημα. Το πρότυπο που αναλύθηκε στη μελέτη αυτή αντιπροσωπεύει έναν ελαστικά υποστηριγμένο στροφέα, που υποβάλλεται σε διέγερση από μια δυσαναλογία, η οποία είναι περιορισμένη στη δισδιάστατη γραμμική κάθετο μετακίνησης στον άξονα της περιστροφής. Από τους διαστημικούς υπολογισμούς πλήρους φάσης διαπιστώνεται ότι καθώς η εκκαθάριση μεταξύ του στροφέα αυξάνεται, υπάρχει μια ευδιάκριτη μείωση στην τιμή της διάστασης συσχέτισης. Κατά συνέπεια, μπορεί να είναι εφικτό να καθοριστούν τα μεταβαλλόμενα γεγονότα όπως η εκκαθάριση χάσματος με τον έλεγχο της διάστασης συσχέτισης του συστήματος [11].

Το 2004, χρησιμοποιήθηκε προκειμένου να εξεταστεί η παθολογία σε ασθενείς με σύνδρομο παρεμποδιστικού ύπνου, άπνοιας/υπόπνοιας (OSAHS). Ερευνήθηκαν οι μη γραμμικές ιδιότητες της αναπνευστικής κίνησης και οι παραλλαγές από αναπνοή σε αναπνοή κατά τη διάρκεια της αγρυπνίας με κλειστά μάτια. Η καταγραφή της αναπνευστικής κίνησης που χρησιμοποιεί την επαγωγική πληθυσμογραφία εκτελέσθηκε σε 14 ασθενείς με OSAHS και 13 θέματα ελέγχου για 2 ώρες στην ύπτια θέση κατά τη διάρκεια της ημέρας. Για να υπολογιστεί η διάσταση συσχέτισης (D_2) για την αναπνευστική κίνηση, χρησιμοποιήθηκε ο αλγόριθμος, που προτάθηκε από τους Grassberger και Procaccia. Οι δείκτες των παραλλαγών από αναπνοή σε αναπνοή υπολογίστηκαν. Για να υπολογίσουν την τιμή της D_2 και τις παραλλαγές από αναπνοή σε αναπνοή, δύο διαφορετικά τμήματα του ίδιου σήματος επιλέχτηκαν (200 δευτερόλεπτα το κάθε ένα). Η τιμή της D_2 για την αναπνευστική κίνηση στους ασθενείς με OSAHS ήταν σημαντικά μεγαλύτερη από αυτή των υγιών. Συμπερασματικά η συγκεκριμένη μελέτη έδειξε ότι οι μετρήσεις των διαστάσεων συσχέτισης για την αναπνευστική κίνηση σε μια συνοπτική περίοδο κατά τη διάρκεια της αγρυπνίας, μπορούν να είναι ένας χρήσιμος δείκτης για τον προσδιορισμό των ασθενών με το παρεμποδιστικό σύνδρομο ύπνου άπνοιας/υπόπνοιας [22].

Το 2006 η μέθοδος χρησιμοποιήθηκε για να μελετηθεί η συμπεριφορά των επίκεντρων των σεισμών, εστιάζοντας στο πρόβλημα των συστηματικών λαθών στα αποτελέσματα που υπολογίζονται. Μεταξύ των αποτελεσμάτων που λαμβάνονται υπ' όψιν είναι η ανομοιογένεια της διανομής του βάθους, τα οριακά αποτελέσματα, λάθη θέσης σεισμού, και η χρονική εξάρτηση. Συγκεκριμένα ο υπολογισμός της διάστασης συσχέτισης του σεισμού είναι μια από τις συζητημένες τεχνικές για την αξιολόγησή του. Ωστόσο διαφορετικές τιμές της εκτίμησης της διάστασης συσχέτισης μπορούν να ληφθούν για παρόμοια σήματα που προκύπτουν από σεισμούς λόγω ανεπάρκειας αριθμού στοιχείων και λαθών στις μετρήσεις. Είναι πιθανό ότι τέτοιες τεχνικές δυσκολίες εντείνονται όσο η τιμή της εκτίμησης της διάστασης συσχέτισης αυξάνεται λόγω του μεγάλου αριθμού στοιχείων που απαιτούνται για τον υπολογισμό της. Λαμβάνοντας υπόψη τα πιθανά λάθη οι συγγραφείς αυτής της μελέτης κατέληξαν στο συμπέρασμα ότι η τιμή της εκτίμησης της διάστασης συσχέτισης για ρηχή σεισμογένεια πλησιάζει το 2.20 [16],[27].

ΚΕΦΑΛΑΙΟ 3. ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

3.1. Αλγόριθμοι για τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης

Τα τελευταία 30 χρόνια η ανάλυση των ιδιοτήτων των μη γραμμικών χρονοσειρών έχουν γίνει μια βασική πρακτική για την μελέτη των μη γραμμικών συστημάτων. Όταν εφαρμόζονται σε οποιοδήποτε πεδίο μετρήσεων, ή αριθμητικών δεδομένων που προέρχονται από μη γραμμικά συστήματα, εμφανίζονται κάποια αποτελέσματα. Η ανάλυση τους προσδιορίζει την διαφορετικότητα των μετρήσεων οι οποίες χαρακτηρίζουν ή τουλάχιστον δίνουν κάποιες πληροφορίες για την δομή και την πολυπλοκότητα του συστήματος.

Ανάμεσα σε πολλά εργαλεία τα οποία είναι διαθέσιμα να χαρακτηρίσουν ένα μη γραμμικό σήμα, οι αποκαλούμενες διαστάσεις είναι αυτές οι οποίες έχουν το μεγαλύτερο νόημα. Οι πρώτοι αλγόριθμοι κουτιού οι οποίοι δημιουργήθηκαν για τον υπολογισμό των διαστάσεων ήταν υπολογιστικά πολύ δαπανηροί και δύσκολοι στην υλοποίηση τους.

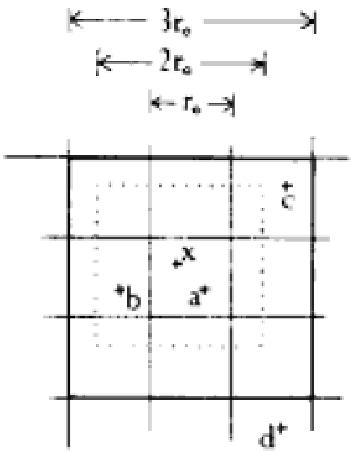
Ένα σημαντικό βήμα στο κομμάτι αυτό έγινε από τους Grassberger και Procaccia οι οποίοι έδειξαν πως οι αποκαλούμενες διαστάσεις συσχέτισης μπορούν να χρησιμοποιηθούν και να ορίσουν την διάσταση η οποία σχετίζεται με τις μη γραμμικές χρονοσειρές. Αυτός ήταν και ο πρώτος αλγόριθμος (Αλγόριθμος Original) που εμφανίστηκε για τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης το 1983 [14] σύμφωνα με τον οποίο υπολογίζονται οι αποστάσεις μεταξύ όλων των πιθανών ζευγών $\vec{y}_i, \vec{y}_j \in Y = \left\{\vec{y}_i\right\}_{i=1}^{N_m}$. Ο αλγόριθμος αυτός για πολλά χρόνια ήταν εξαιρετικά διαδεδομένος λόγω της απλότητας του.

Ωστόσο αργότερα παρουσιάστηκαν τα πρώτα προβλήματα και έτσι η μελέτη στράφηκε στο πώς να εφαρμόζεται η μέθοδος αυτή με προσοχή λόγο των περιορισμών αλλά και των παγίδων που κρύβει. Από υπολογιστικής άποψης το κόστος του αλγόριθμου των Grassberger – Procaccia (Αλγόριθμος Original) σε ένα σύνολο δεδομένων N είναι της τάξης $O(N^2)$. Για μεγάλες χρονοσειρές N > 60.000, στις οποίες και έχει νόημα ο αλγόριθμος, αυτή η εξάρτηση δευτέρου βαθμού κάνει αμέσως τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης μια διαδικασία με αυξημένο κόστος. Αυτό ειδικά είναι πολύ σημαντικό γιατί έχουμε να κάνουμε με χρονοσειρές οι οποίες ασχολούνται με αυξημένων διαστάσεων συστήματα όπως για παράδειγμα ηλεκτρο-φυσικά δεδομένα τα οποία προέρχονται από ηλεκτροεγκεφαλογραφήματα ή ηλεκτροκαρδιακές καταγραφές.

Το πρώτο πρόβλημα που μελετήθηκε ήταν να μπορεί να υπολογιστεί η εκτίμηση της διάστασης συσχέτισης σε πραγματικό χρόνο για μεγάλο αριθμό διανυσμάτων (N). Έτσι ο Theiler το 1987 [31], πρότεινε έναν νέο αλγόριθμο (Αλγόριθμος Box Assisted) για να μειωθεί το υπολογιστικό κόστος της D_2 ακόμη και για μεγάλα N. Σύμφωνα με τον αλγόριθμο αυτό τα διανύσματα $Y = \left\{\vec{y}_i\right\}_{i=1}^{N_m}$ που παράγονται από την χρονική σειρά εισόδου, χωρίζονται με βάση τις συντεταγμένες τους στον R^m σε κουτιά όπως φαίνεται στο Σχήμα 3.1 και αποθηκεύονται σε μια λίστα γραμμικών λιστών.

Σχήμα 3.1 Τα Διανύσματα της Χρονοσειράς Χωρίζονται σε Κουτιά

Πλέον δεν υπολογίζονται όλες οι αποστάσεις. Υπολογίζονται μόνο οι αποστάσεις μεταξύ ζευγαριών διανυσμάτων τα οποία βρίσκονται στο ίδιο ή σε γειτονικά κουτιά όπως φαίνεται στο Σχήμα 3.2.



Σχήμα 3.2 Υπολογίζονται οι αποστάσεις μεταξύ των διανυσμάτων a,x,c,b και όχι d

Αν και δημιουργείται ένας πρόσθετος φόρτος στη μνήμη ο αλγόριθμος του Theiler δεν μπόρεσε μειώσει την πολυπλοκότητα του Original αλγόριθμου. Ωστόσο ήταν ο πρόδρομος μιας ολόκληρης οικογένειας πιο εξελιγμένων αλγορίθμων κουτιού για τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης. Μάλιστα ο αλγόριθμος αυτός, για m > 2 συμπεριφέρεται χειρότερα από τον Original.

Μια εναλλακτική που παρουσιάστηκε το 1989 ώστε να μειωθεί το κόστος σε μνήμη λόγω της χρήσης κουτιών ήταν τα δέντρα (Αλγόριθμος k-d Trees) [3]. Τα k-d δέντρα αποτελούν πιο καλή επιλογή για αναζήτηση κοντινών γειτόνων σε δεδομένα k-διάστασης στον Ευκλείδειο χώρο. Ο αλγόριθμος αυτός είναι τάξης O(NlogN). Το ποσό μνήμης που απαιτείται για την κατασκευή ενός πολυδιάστατου δέντρου είναι ανάλογο προς τον αριθμό των στοιχείων που προκύπτουν από την αναδόμηση της χρονικής σειράς εισόδου στον R^m, έτσι δεν αυξάνεται όσο η διάσταση ενσωμάτωσης (m) αυξάνεται. Μπορεί ο αλγόριθμος αυτός να μην έχει τόσο μεγάλο κόστος σε μνήμη ωστόσο είναι ο μοναδικός αλγόριθμος που μπόρεσε να μει'ωσει την πολυπλοκότητα σε χρόνο του Original.

Το 1990 παρουσιάζεται ένας νέος αλγόριθμος βασισμένος στην οικογένεια αλγορίθμων κουτιού από τον Grassberger (Αλγόριθμος Optimized Box Assisted) [12], ο οποίος βελτιώνει το πρόβλημα του μεγάλου κόστους σε μνήμη του αλγόριθμου του Theiler (Αλγόριθμος Box Assisted) [31]. Στον αλγόριθμο αυτό τα διανύσματα χωρίζονται και πάλι σε κουτιά με βάση τις συντεταγμένες τους και η δομή αποθήκευσης είναι ένα πλέγμα, δηλαδή ένας m-διάστασης πίνακας που αυτή τη φορά συνδέεται με έναν και μόνο πίνακα μεγέθους N. Έτσι το κάθε κουτί δεν συνδέεται με μια διαφορετική λίστα μεγέθους N όπως στον Αλγόριθμο Box Assisted. Το μέγεθος του πίνακα σε κάθε διάσταση (L_k), αποτελεί άλλη μια παράμετρο του αλγορίθμου και υπολογίζεται ή εμπειρικά ή με δοκιμές. Ο Αλγόριθμος εξαρτάται από το μέγεθος του πλέγματος και δεν μπορεί να ανταποκριθεί σε πραγματικό χρόνο για μεγάλο αριθμό διανυσμάτων όσο η διάστασή του αυξάνεται και η διάσταση ενσωμάτωσης m αυξάνεται.

Εξαιτίας του ότι οι αλγόριθμοι κουτιού εξαρτώνται από το μέγεθος του κουτιού, το 2007 παρουσιάστηκε μια σχετική μελέτη [5] για το πως ένας σωστά ορισμένος αλγόριθμος κουτιού μπορεί να είναι εξίσου γρήγορος με τον Αλγόριθμο k-d Trees και ορισμένες φορές και ακόμα πιο γρήγορος για κάθε διάσταση ενσωμάτωσης. Αυτό που προτάθηκε είναι ένας τρόπος υπολογισμού του μεγέθους του κουτιού το οποίο να θεωρείται ιδανικό.

Με αυτούς τους αλγόριθμους ο υπολογισμός της εκτίμησης της διάστασης συσχέτισης (D_2) ήταν δύσκολος για μεσαία ή μεγάλα συστήματα όπου D_2 > 5-6, λόγω του ότι ο υπολογισμός της σε αυτά τα συστήματα απαιτεί μεγάλο αριθμό διανυσμάτων (N) και οι αλγόριθμοι που υπήρχαν ήταν της τάξης O(NlogN) στην καλύτερη περίπτωση. Για τον υπολογισμό της D_2 σε μεγάλα συστήματα «κόβονταν» πληροφορία, δηλαδή δεν χρησιμοποιούνταν χρονοσειρές με μεγάλο N. O Theiler πρότεινε για τη λύση αυτού του προβλήματος έναν διπλό υπολογισμό: έναν χρησιμοποιώντας όλο το σύνολο διανυσμάτων με μια μικρή κοντινή απόσταση και έναν χρησιμοποιώντας ένα μικρό πακέτο διανυσμάτων και έχοντας υπόψη όλες τις αποστάσεις Έτσι τα αποτελέσματα θεωρήθηκαν αναξιόπιστα [17]. Για αυτό το λόγο οι μελετητές στράφηκαν σε παράλληλους αλγόριθμους.

Ο πρώτος παράλληλος αλγόριθμος εμφανίστηκε το 1994. Ο αλγόριθμος αυτός (Αλγόριθμος Parallel Box Assisted) [10], είναι ο Αλγόριθμος Box Assisted υλοποιημένος χρησιμοποιώντας το μοντέλο της εικονικής διαμοιραζόμενης μνήμης. Ο αλγόριθμος αυτός χρειάζεται 5.300 sec για N = 360.000 παρουσιάζοντας πολύ μεγάλη βελτίωση στον υπολογισμό της D_2 σε πολύ μεγάλα συστήματα.

Αργότερα (1998) ο Corana παρουσίασε άλλον έναν παράλληλο αλγόριθμο (Αλγόριθμος Parallel Optimized Box Assisted) [8], [9] για τον υπολογισμό της D_2. Σε αυτόν τον αλγόριθμο διατυπώνονται τρεις προσεγγίσεις: η πρώτη υπολογίζει όλες τις αποστάσεις μεταξύ των διανυσμάτων στο διάστημα φάσης, ενώ η δεύτερη και τρίτη προσέγγιση υπολογίζουν μόνο τις αποστάσεις οι οποίες είναι μικρότερες από μια απόσταση r. Η τρίτη προσέγγιση υλοποιείται με τον αλγόριθμο Optimized Box Assisted. Η παράλληλη λειτουργία αυτού του αλγόριθμου είναι σχεδιασμένη για πολύ-επεξεργαστικά συστήματα με κατανεμημένη μνήμη στον κάθε επεξεργαστή που χρησιμοποιούν το μοντέλο περάσματος μηνύματος. Η ενιαία υλοποίηση και υπολογιστική ανάλυση επέτρεψαν μια καθαρή σύγκριση ανάμεσα στις τρεις εκδόσεις. Η τρίτη έκδοση είναι ιδιαίτερα κατάλληλη για γρήγορη επεξεργασία πολύ μεγάλων χρονικών σειρών γεγονός που επιτρέπει την αποτίμηση του D_2 ακόμα και για μεσαία αλλά και για μεγάλα συστήματα στα οποία ένας εξαιρετικά μεγάλος αριθμός διανυσμάτων απαιτείται.

Το 2000 ένας νέος αλγόριθμος παρουσιάζεται από τον Corana (Αλγόριθμος Adaptive box-assisted) [7]. Ο αλγόριθμος αυτός είναι ένας προσαρμόσιμος αλγόριθμος κουτιού (Αλγόριθμος adaptive box assisted) και ανήκει στην κατηγορία των προσεγγιστικών λύσεων του προβλήματος. Το βασικό χαρακτηριστικό του αλγορίθμου είναι η χρήση μεταβλητού αριθμού διανυσμάτων με σκοπό να κρατηθεί ο αριθμός «κοντινών» ζευγαριών περίπου σταθερός στις διάφορες κλίμακες και σε διάφορες διαστάσεις ενσωμάτωσης.

Η διαδικασία απαιτεί ένα αριθμό βημάτων κατά τα οποία μειώνεται η απόσταση (παράμετρος r) μέσα στο κουτί, και σε κάθε βήμα μόνο τα γειτονικά ζευγάρια λαμβάνονται υπόψη χρησιμοποιώντας μια προσέγγιση κουτιού. Ο αλγόριθμος αυτός όμως χρειάζεται άλλη μία παράμετρο, τον ελάχιστο αριθμό κοντινών ζευγαριών διανυσμάτων (N_{th}). Με μια κατάλληλη επιλογή της παραμέτρου N_{th}, επιτυγχάνεται μια ουσιαστική μείωση του χρόνου υπολογισμού της εκτίμησης της διάστασης συσχέτισης, λαμβάνοντας ουσιαστικά τις ίδιες ακέραιες τιμές όπως στην πρότυπη διαδικασία.

Η πολυπλοκότητά του είναι O(NlogN). Και αυτός ο αλγόριθμος δεν μπορεί να υπολογίσει την εκτίμηση της D_2 για μεγάλα συστήματα (για τιμές της $D_2 > 5,6$) δεδομένου του πολύ μεγάλου αριθμού διανυσμάτων που απαιτούνται για τον υπολογισμό της.

Το 2003 άλλος ένας προσεγγιστικός αλγόριθμος προτάθηκε (Αλγόριθμος Δr neighborhood) [21]. Σύμφωνα με τον αλγόριθμο αυτό, κάποια διανύσματα (N_{ref}) επιλέγονται σαν διανύσματα αναφοράς. Οι αλγόριθμοι που είχαν προταθεί μέχρι τότε, δεν υπολόγιζαν όλες τις αποστάσεις μεταξύ όλων των διανυσμάτων της χρονικής σειράς αλλά μόνο μεταξύ όλων των «κοντινών» ζευγών. Ο αλγόριθμος αυτός για κάθε διάνυσμα αναφοράς υπολογίζει μόνο μια φορά την απόσταση από τα κοντινά του διανύσματα και θεωρεί ότι η απόσταση αυτή είναι ίδια και για όλα τα υπόλοιπα διανύσματα που είναι κοντινά του. Το διάνυσμα αναφοράς θεωρείται ως ο αντιπρόσωπος q άλλων διανυσμάτων. Έτσι ο υπολογισμός του C(m, r, t) γίνεται:

$$C(m,r,t)=\frac{2}{N_{ref}(N-1)}\sum_{i=1}^{N_{ref}}\sum_{j>i}^{N} H\,\|\vec{y}_i-\vec{y}_j\|$$

Η πολυπλοκότητα του αλγορίθμου σύμφωνα με τους συγγραφείς του, είναι $O(N*N_{ref}/q^2)$, όπου q ο αριθμός των διανυσμάτων που αντιπροσωπεύει κάθε διάνυσμα αναφοράς. Ο αλγόριθμος αυτός υπόσχεται πολύ καλά αποτελέσματα ακόμη για μεγάλες τιμές της διάστασης ενσωμάτωσης.

Ωστόσο δεν αναλύεται ο τρόπος που γίνεται η επιλογή των διανυσμάτων αναφοράς, ούτε το κόστος για την εύρεσή τους. Τέλος τα αποτελέσματα του αλγορίθμου μπορεί να είναι πολύ κοντά στην ακριβή λύση αλλά δεν είναι η ακριβής λύση. Για μεγάλο αριθμό διανυσμάτων σε συστήματα όπου $D_2 > 5{,}6$ τα αποτελέσματά του είναι πολύ αναξιόπιστα.

3.2. Κατηγοριοποίηση των αλγορίθμων για τον υπολογισμό της εκτίμησης της διάστασης συσχέτισης

Οι αλγόριθμοι που έχουν προταθεί μέχρι τώρα μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες. Στους αλγόριθμους που βρίσκουν ακριβή λύση και σε αυτούς που υπολογίζουν προσεγγιστικά την εκτίμηση της διάστασης συσχέτισης όπως φαίνεται στον Πίνακα 3.1 και Πίνακα 3.2.

Πίνακας 3.1 Αλγόριθμοι που βρίσκουν ακριβή λύση

Αλγόριθμος	Πολυπλοκότητα
Original	$O(N_m{}^2)$
Box Assisted	$counter*m*N_{box}*N_m{}^2$
k-d Trees	$O(N_m*\log N_m)$
Optimized Box Assisted	$O(L_1*L_2*\ldots*L_m*N_m{}^2)$

Counter = αριθμός γεμάτων κουτιών, N_{box} = αριθμός γειτονικών κουτιών. Αναλυτικότερα οι πολυπλοκότητες των Αλγορίθμων αυτών παρουσιάζονται στο επόμενο Κεφάλαιο.

Πίνακας 3.2 Αλγόριθμοι που βρίσκουν προσεγγιστική λύση

Αλγόριθμος	Πολυπλοκότητα
Adaptive box-assisted	$O(N_m \log N_m)$
Δr neighborhood	$O(N_m * N_{ref}/q^2)$

Η προσεγγιστική λύση μπορεί να επιφέρει μεγαλύτερα προβλήματα από το πρόβλημα του υπολογισμού της D_2 σε πραγματικό χρόνο για μεγάλα συστήματα. Αυτό συμβαίνει γιατί στον υπολογισμό της εκτίμησης της D_2 υπεισέρχονται και λάθη. Στατιστικά λάθη τα οποία είναι πιο σημαντικά σε μικρότερες κλίμακες και τα συστηματικά λάθη μετρήσεων. Σε συνδυασμό με το γεγονός ότι οι προσεγγιστικοί αλγόριθμοι βρίσκουν μια τιμή της D_2 «κοντά» στην πραγματική, τα αποτελέσματά τους μπορεί να είναι αναξιόπιστα. Για τον λόγο αυτό άλλωστε δεν χρησιμοποιούνται από τους ερευνητές αυτοί οι αλγόριθμοι.

Στην παρούσα εργασία μελετήθηκαν μόνο οι αλγόριθμοι που βρίσκουν ακριβή λύση και δεν είναι παράλληλοι.

ΚΕΦΑΛΑΙΟ 4. ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΗΣ D_2

4.1. Αλγόριθμος Original

Ο Original Αλγόριθμος έτσι όπως προτάθηκε από τους Grassberger και Procaccia [13] φαίνεται παρακάτω:

$$
\begin{array}{ll}
01: & Nm = N - (m-1)t \\
02: & \forall i, 1 \le i \le Nm \\
03: & \quad \forall j, 1 \le j \le m \quad do\ begin \\
04: & \qquad Y_{i,j} = X_{i+j*t} \\
05: & end \\
06: & \forall i, 1 \le i \le Nm \\
07: & \quad \forall j, 1 \le j \le Nm \quad do\ begin \\
08: & \qquad if\ \| \vec{y}_i - \vec{y}_j \|_m < r\ then \\
09: & \qquad\quad sum = sum + 1 \\
10: & end \\
11: & CorDim = \frac{1}{N^2} sum
\end{array}
$$

Αν X είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 το σήμα μετασχηματίζεται στην m διάσταση. Στις γραμμές 06-11 υπολογίζεται το C(m, r, t).

4.1.1. Πολυπλοκότητα του Αλγόριθμου Original

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος X στον R^m τότε t_{read} είναι της τάξης $O(m*N_m)$. Αν m = 1 τότε N_m = N.

Αν t_{com} είναι ο χρόνος που χρειάζεται για τον υπολογισμό του C(m, r, t) τότε t_{com} είναι της τάξης $O(m*N_m^2)$ αφού υπολογίζεται η απόσταση μεταξύ κάθε πιθανού ζεύγους y_i και y_j.

Συνολικά η πολυπλοκότητα του αλγορίθμου είναι της τάξης O $O(N_m^2)$, ενώ η πολυπλοκότητα σε μνήμη είναι $O(N_m)$.

4.2. Αλγόριθμος Box Assisted

Για μεγάλες χρονοσειρές, όπου το N > 60.000, $N_m{}^2$ ζεύγη $\vec{y}_i$ και $\vec{y}_j$ εξετάζονται για ομοιότητα. Για να μειωθεί το πλήθος των πιθανών ζευγών που περνούν το τεστ ομοιότητας προτάθηκε ο Αλγόριθμος Box Assisted. Ο Αλγόριθμος αυτός ορίζει κουτιά μεγέθους $R_1\, x\, R_2\, x\, R_3\, x \ldots x\, R_m$, όπου $R_1 = R_2 = R_3 = \ldots = R_m = r$ στον R^m Ευκλείδειο χώρο σύμφωνα με τις παραμέτρους m και r που επιλέχθηκαν. Για κάθε κουτί μόνο τα διανύσματα που βρίσκονται μέσα στα γειτονικά του κουτιά θα περάσουν το τεστ ομοιότητας $H \| \vec{y}_i - \vec{y}_j \|$.

Ο Αλγόριθμος αυτός αφού το σήμα X αναδομηθεί στον R^m Ευκλείδειο χώρο στο σήμα Y, ταξινομεί το Y λεξικογραφικά. Στη λεξικογραφική ταξινόμηση το a < b, όπου $a = (a_1, a_2, a_3, \ldots, a_m)$ και $b = (b_1, b_2, b_3, \ldots, b_m)$, αν και μόνο αν $a_k < b_k \ \forall\, k \le m$.

Η λεξικογραφική ταξινόμηση γίνεται χρησιμοποιώντας τον Αλγόριθμο quicksort που η πολυπλοκότητά του είναι τάξης O(NlogN).

Για να βρεθεί ο τρόπος που κάθε $\vec{y}_i$ μπαίνει σε κάποιο κουτί, έγινε μια υπόθεση επειδή δεν διασαφηνίζεται στην εργασία στην οποία προτάθηκε. Σύμφωνα με την υπόθεση που έγινε, για να βρεθεί σε ποιο κουτί μπαίνει κάθε $\vec{y}_i$ γίνεται δυαδική αναζήτηση στο λεξικογραφικά ταξινομημένο σήμα και αναζητούνται όλα τα $\vec{y}_i\,'$ στα οποία, αν $\vec{y}_i = (y_{i1}, y_{i2}, y_{i3}, ..., y_{im})$ τότε $\vec{y}_{\boldsymbol{i}}\,' = (y'_{i1} \leq y_{i1} + r, y'_{i2}, y'_{i3}, ..., y'_{im})$. Δηλαδή τα διανύσματα $\vec{y}_i\,'$ που πέφτουν στο ίδιο κουτί πρέπει στην πρώτη και μόνο στην πρώτη (αυτή είναι η υπόθεση) συντεταγμένη τους να έχουν +r απόσταση από την τιμή της πρώτης συντεταγμένης του $\vec{y}_i$. Η υπόθεση έγινε έτσι γιατί, η εύρεση του κατάλληλου κουτιού για κάθε $\vec{y}_i$ είναι πιο γρήγορη με αυτόν τον τρόπο από ότι αν ο αλγόριθμος έψαχνε για $\vec{y}_{\boldsymbol{i}}\,' = (y'_{i1} \leq y_{i1} + r, y'_{i2} \leq y_{i2} + r, y'_{i3} \leq y_{i3} + r, ..., y'_{im} \leq y_{im} + r)$. Όλα αυτά τα $\vec{y}_i\,'$ και το $\vec{y}_i$ μπαίνουν στο ίδιο κουτί. Στο επόμενο βήμα το $\vec{y}_i$ = $\vec{y}_{i+1}\,'$ και η δυαδική αναζήτηση ξεκινά από το αμέσως επόμενο διάνυσμα του $\vec{y}_i$. Αυτό είναι και το επόμενο κουτί.

Η παραπάνω διαδικασία υλοποιείται ως εξής: Τα διαφορετικά κουτιά κρατιούνται σε έναν πίνακα (box). Κάθε γεμάτο κουτί αυτού του πίνακα συνδέεται με έναν πίνακα μεγέθους N_m, γιατί N_m είναι ο μέγιστος πιθανός αριθμός διανυσμάτων που μπορεί να πέσουν σε ένα κουτί. Στο πίνακα αυτό δεν αποθηκεύεται η τιμή $\vec{y}_i$ καθεαυτή αλλά ο δείκτης i.

Επίσης για να μην γίνεται κατασπατάληση της μνήμης μόνο τα κουτιά που δεν είναι άδεια κρατιούνται στο πίνακα box. Στο σημείο αυτό έγινε άλλη μία υπόθεση εξαιτίας του ότι ο συγγραφέας του αλγόριθμου αυτού δεν λέει τον τρόπο με τον οποίο μπορούμε εξαρχής να γνωρίζουμε πόσα γεμάτα κουτιά θα υπάρξουν. Η υπόθεση είναι ότι ο πίνακας που κρατιούνται τα διαφορετικά κουτιά είναι μεγέθους L. L είναι ο αριθμός διαφορετικών κουτιών που παράγονται στον R^m έτσι ώστε κάθε $\vec{y}_i$ να «πέφτει» μέσα σε κάποιο κουτί.

Η τιμή του L υπολογίζεται ως εξής : L_k είναι ο αριθμός των κουτιών για διάσταση $k \in [1, m]$ του σήματος Y και $L_k = |\min_k(Y_{ik}) - \max_k(Y_{ik})| \frac{1}{r}$,

δηλαδή L_k = | (μικρότερη τιμή όλων των διανυσμάτων Y_i στη διάσταση k) - (μεγαλύτερη τιμή όλων των διανυσμάτων Y_i στη διάσταση k)| / r ,(με στρογγυλοποίηση προς τα πάνω γιατί πρέπει να είναι ακέραια η τιμή).

Τότε L = $L_1 x L_2 x L_3 x \ldots x L_k$. Στο πίνακα box τα γεμάτα κουτιά αποθηκεύονται το ένα μετά το άλλο προσπερνώντας τα άδεια κουτιά. Τελικά ο box θα είναι ένας δισδιάστατος πίνακας.

Για να υπολογιστεί τέλος το C(m, r, t) πρέπει να γίνει το τεστ ομοιότητας ανάμεσα στα πιθανά κοντινά ζεύγη $\vec{y}_i$ και $\vec{y}_j$. Για κάθε κουτί μόνο τα διανύσματα που βρίσκονται μέσα στα γειτονικά του κουτιά θα περάσουν το τεστ ομοιότητας. Και στο σημείο αυτό έγινε μια υπόθεση. Εξαιτίας του τρόπου που μπαίνει το κάθε $\vec{y}_i$ στο κουτί και δημιουργείται ο πίνακας box γειτονικά θεωρούνται τα κουτιά στα οποία η τιμή των $\vec{y}_i\,'$ είναι:

$\vec{y}_i\,' = (y'_{i1} > y_{i1} - r, y'_{i2}, y'_{i3}, \ldots, y'_{im})$ και $\vec{y}_i\,' = (y'_{i1} < y_{i1} + 2r, y'_{i2}, y'_{i3}, \ldots, y'_{im})$.

Επειδή το σήμα Y είναι λεξικογραφικά ταξινομημένο και ο πίνακας με τα κουτιά θα είναι ταξινομημένος. Έτσι για να βρεθούν οι γείτονες του κάθε κουτιού εκτελείται και πάλι δυαδική αναζήτηση (Αλγόριθμος Binary Search) στο πίνακα box.

Ο Αλγόριθμος Box Assisted σε τυπική μορφή φαίνεται παρακάτω:

$Nm = N - (m-1)t$

$\forall i, 1 \le i \le Nm$

$\quad \forall j, 1 \le j \le m \quad do\ begin$

$\qquad Y_{i,j} = X_{i + j*t}$

end

$lexicographic\ order(Y)$

$normalize(Y)$

$$\forall k, 1 \le k \le m \quad do\,begin$$

$$\quad L_k = \lceil |\min_k (Y_{i,k}) - \max_k (Y_{i,k})| \frac{1}{r} \rceil$$

$$end$$

$$L = L_1 \; x \; L_2 \; x \; L_3 \; x \ldots x \; L_m$$

$$counter = 0$$

$$\forall i, 1 \le i \le Nm \quad do\,begin$$

$$\quad index = 1$$

$$\quad box_{counter,index} = i$$

$$\quad high = binary\,search\,\,find\,\,higher\,k : y_{k,1} \le (y_{i,1} + r), \quad k \in [1, Nm]\,and\,\,(y_{i,1} + r) < y_{N_m,1}$$

$$\quad \forall candidate, i \le candidate \le high \quad do\,begin$$

$$\qquad index = index + 1$$

$$\qquad box_{counter,index} = candidate$$

$$\quad end$$

$$\quad counter = counter + 1$$

$$\quad i = high + 1$$

$$end$$

$$\forall i, 1 \le i \le counter \quad do\,begin$$

$$\quad low = binary\,search\,\,find\,\,lower\,k : \quad y_{box_{k,1},1} \le y_{box_{i,1},1} - r, \quad k \in [1, counter]\,and\,\,y_{box_{i,1},1} - r > 0$$

$$\quad high = binary\,search\,\,find\,\,higher\,k : \quad y_{box_{k,1},1} \le y_{box_{i,1},1} + 2 * r, \quad k \in [1, counter]\,and\,\,y_{box_{i,1},1} + 2 * r < y_{N_m,1}$$

$$\quad \forall j, low \le j \le high \quad do\,begin$$

$$\qquad \forall l, 1 \le l \le Nm$$

$$\qquad\quad \forall w, 1 \le w \le Nm \;\; do \;\; begin$$

$$\qquad\qquad if \; \| \vec{y}_{box_{j,l}} - \vec{y}_{box_{j,w}} \|_m < r \; then$$

$$\qquad\qquad\quad sum = sum + 1$$

$$\qquad\quad end$$

$$\quad end$$

$$end$$

$$CorDim = \frac{1}{N^2} sum$$

Αν X είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 το σήμα μετασχηματίζεται σε Y στην m διάσταση.

Στη γραμμή 06 ταξινομείται λεξικογραφικά ενώ στη γραμμή 07 το σήμα κανονικοποιείται ώστε να ξεκινά από το 0. Στις γραμμές 08-11 υπολογίζεται το μέγεθος L του πίνακα box. Στις γραμμές 12-23 φτιάχνεται ο πίνακας box και κάθε $\vec{y}_i$ μπαίνει στο κουτί που πρέπει σύμφωνα με τη διαδικασία που ορίστηκε παραπάνω.

Στις γραμμές 24-34 γίνονται τα τεστ ομοιότητας των $\vec{y}_i$ και $\vec{y}_j$ που ανήκουν σε γειτονικά κουτιά. Συγκεκριμένα στη γραμμή 25 βρίσκει το κουτί που έχει απόσταση –r από το ίδιο και στη γραμμή 26 βρίσκει το κουτί που έχει απόσταση +r από το ίδιο σύμφωνα με τη διαδικασία που ορίστηκε παραπάνω. Στη γραμμή 35 υπολογίζεται το C(m, r, t).

4.2.1. Πολυπλοκότητα του Αλγόριθμου Box Assisted

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος X στον R^m τότε t_{read} είναι της τάξης $O(m*N_m)$.

Αν $t_{lexicogr}$ είναι ο χρόνος που απαιτείται για τη λεξικογραφική ταξινόμηση του σήματος Y τότε $t_{lexicogr}$ είναι της τάξης $O(m*N_m*\log N_m)$. Αυτό συμβαίνει γιατί χρησιμοποιείται ο Αλγόριθμος quicksort που η πολυπλοκότητά του είναι O(NlogN) και για τις m διαστάσεις του $\vec{y}_i$.

Αν t_{normal} είναι ο χρόνος που απαιτείται για τη κανονικοποίηση του σήματος Y τότε t_{normal} είναι της τάξης $O(m*N_m)$.

Αν $t_{size\ of\ box}$ είναι ο χρόνος που απαιτείται για να υπολογιστεί το μέγεθος του πίνακα box τότε $t_{size\ of\ box}$ είναι της τάξης $O(m*2*N_m)$ αφού ψάχνει να βρει σε κάθε διάσταση τη μικρότερη τιμή του $\vec{y}_i$ και τη μεγαλύτερη τιμή του $\vec{y}_i$ και αυτό γίνεται $\forall i \in [1, N_m]$.

Αν $t_{findbox}$ είναι ο χρόνος που απαιτείται για βρεθεί σε ποιο κουτί «πέφτει» κάθε $\vec{y}_i$ τότε $t_{findbox}$ είναι της τάξης O(N_m* logN_m). Για κάθε διάνυσμα γίνεται δυαδική αναζήτηση, η οποία είναι O(logN).

Αν t_{com} είναι ο χρόνος που χρειάζεται για τον υπολογισμό του C(m, r, t) τότε t_{com} = (counter* (m*N_{box}*N_m^2 + 2*log counter)). N_{box} είναι ο αριθμός των γειτονικών κουτιών. Αυτό προκύπτει ως εξής: Για counter κουτιά εκτελείται 2 φορές δυαδική αναζήτηση για να βρεθούν τα «κοντινά» κουτιά (χρόνος 2*log counter), και για κάθε κοντινό κουτί γίνεται το τεστ ομοιότητας στα $\vec{y}_i$ που έχουν πέσει μέσα σε αυτά τα κουτιά (χρόνος = m*N_{box}*N_m^2). Οπότε η διαδικασία αυτή είναι της τάξης O (counter*m*N_{box}*N_m^2)

Συνολικά η πολυπλοκότητα του αλγορίθμου είναι της τάξης O(counter*m*N_{box}*N_m^2). Ωστόσο λόγο της όλης διαδικασίας εύρεσης κουτιού και «κοντινών» κουτιών, ο αλγόριθμος αυτός συμπεριφέρεται χειρότερα από τον Original. Για καλά διαλεγμένες τιμές των παραμέτρων και m < 3 όμως, ο συγγραφέας του Αλγόριθμου υποστηρίζει ότι συμπεριφέρεται καλύτερα από τον Original, ενώ οι καλές τιμές των παραμέτρων δεν αναφέρονται.

Η πολυπλοκότητά του σε μνήμη είναι της τάξης O(N_m + (counter x N_m) + (L-counter)). Όπου counter είναι ο αριθμός των γεμάτων κουτιών. Μόνο για τα γεμάτα κουτιά δεσμεύεται ο πίνακας μεγέθους N_m για τα διανύσματα που «πέφτουν» στο ίδιο κουτί. Με βάση την υπόθεση που έγινε για την εύρεση του αριθμού των κουτιών που χρειάζονται για να μπουν όλα τα διανύσματα σε κάποιο κουτί, ο πίνακας box έχει μέγεθος L και άρα χρειαζόμαστε και (L-counter) μνήμη ακόμη.

4.3. Αλγόριθμος k-d Trees

Ο Αλγόριθμος αυτός ακολουθεί την ίδια φιλοσοφία με τον Αλγόριθμο Box Assisted. Προσπαθεί να αποκλείσει δηλαδή άσκοπους υπολογισμούς για ομοιότητα μεταξύ $\vec{y}_i$ και $\vec{y}_j$ που σίγουρα δεν είναι όμοια. Και σε αυτόν τον Αλγόριθμο η σιγουριά προκύπτει από την ταξινόμηση του σήματος Y.

Συγκεκριμένα, ο Αλγόριθμος k-d Trees, χρησιμοποιεί τη δομή των k-d Trees για να ταξινομήσει στον R^k Ευκλείδειο χώρο το σήμα Y. Τα k-d Trees είναι μία δομή δυαδικού δέντρου με την διαφορά ότι τα δεδομένα που φυλάσσονται στους κόμβους του δέντρου είναι k διάστασης διανύσματα και εισάγονται στους κόμβους του με βάση την τιμή τους στην i διάσταση, όπου $i \in [1,k]$.

Σύμφωνα με τον Αλγόριθμο k-d Trees, αφού το σήμα X αναδομηθεί στο σήμα Y στον R^m, κατασκευάζεται το δυαδικό δέντρο σύμφωνα με τους παρακάτω κανόνες:

Ρίζα του δέντρου θα είναι το διάνυσμα $\vec{y}_1$ και το επίπεδο του δέντρου στη ρίζα είναι 0. Για κάθε άλλο $\vec{y}_i$, $i \in [1,N_m]$, το $\vec{y}_i$ μπαίνει στο δέντρο ακολουθώντας ένα μονοπάτι από τον ένα κόμβο του δέντρου στον άλλο.

Στη ρίζα, δηλαδή στο επίπεδο 0, συγκρίνεται η πρώτη συντεταγμένη του $\vec{y}_i$ δηλαδή το $\vec{y}_{i,1}$. Αν η τιμή του $\vec{y}_{i,1}$ είναι μικρότερη από τη τιμή του $\vec{y}_{1,1}$ τότε ο αλγόριθμος επισκέπτεται τον αριστερό απόγονο της ρίζας αλλιώς τον δεξί.

Στο επίπεδο 1, συγκρίνεται η δεύτερη συντεταγμένη του $\vec{y}_i$ και στο επίπεδο L, συγκρίνεται η συντεταγμένη με δείκτη k = 1 + (L mod m), όπου m η διάσταση που επιλέχθηκε. Ο κόμβος $\vec{y}_i$ εισάγεται στον πρώτο άδειο κόμβο που θα συναντήσει ακολουθώντας το μονοπάτι όπως φτιάχνεται παραπάνω.

Με τον τρόπο αυτό τα διανύσματα είναι ταξινομημένα και η εύρεση των πιθανά «κοντινών» διανυσμάτων γίνεται πολύ εύκολα. Για κάθε διάνυσμα $\vec{y}_i$ του Y, ο αλγόριθμος επισκέπτεται τους κόμβους $\vec{y}_j$ του δέντρου ξεκινώντας από τη ρίζα όπως περιγράφεται παρακάτω. Για κάθε κόμβο που επισκέπτεται στη διαδρομή, κάνει το τεστ ομοιότητας ανάμεσα στα $\vec{y}_i$ και $\vec{y}_j$.

Όποτε το διάνυσμα $\vec{y}_i$ επισκέπτεται έναν κόμβο $\vec{y}_j$ στο επίπεδο L του δέντρου υπολογίζει τα εξής:
Αν $y_{jk} < y_{ik} - r$, όπου k = 1 + (L mod m), τότε λόγω της κατασκευής του δέντρου για κάθε $\vec{y}_j$ στο αριστερό υποδέντρο του $\vec{y}_j$ θα ισχύει ότι $\| y_j - y_i \| > r$ και έτσι επισκέπτεται τον δεξί απόγονο του $\vec{y}_j$.

Αν $y_{jk} > y_{ik} + r$, όπου k = 1 + (L mod m), τότε αντίστοιχα για κάθε $\vec{y}_j$ στο δεξί υποδέντρο του $\vec{y}_j$ θα ισχύει ότι $\| y_j - y_i \| > r$ και έτσι επισκέπτεται τον αριστερό απόγονο του $\vec{y}_j$.

Τέλος αν ισχύει $y_{jk} < y_{ik} - r$ και $y_{jk} > y_{ik} + r$ τότε επισκέπτεται και τον δεξί και τον αριστερό απόγονο του $\vec{y}_j$.

Η παραπάνω διαδικασία εκτελείται αναδρομικά για κάθε κόμβο που επισκέπτεται στο μονοπάτι από τη ρίζα μέχρι και το τελευταίο επίπεδο του δέντρου. Για κάθε κόμβο που επισκέπτεται το $\vec{y}_i$ σε αυτό το μονοπάτι κάνει και το τεστ ομοιότητας για να υπολογίσει το C(m, r, t).

Ο Αλγόριθμος k-d Trees σε τυπική μορφή φαίνεται παρακάτω:

$: Nm = N - (m-1)t$

$: \forall i, 1 \leq i \leq Nm$

$: \quad \forall j, 1 \leq j \leq m \quad do\,begin$

$: \qquad Y_{i,j} = X_{i+j*t}$

$: end$

$: root = \vec{y}_1, L = 0$

$: \forall i, 1 \leq i \leq Nm \; do\; begin$

$\vec{y}_j = root$

$begin\ \ from\ \ root\ \ and\ \ recursively\ \forall\ new\ \vec{y}_j, search\ place\ \ for\ \vec{y}_i$

$\quad find\ L$

$\quad k = 1 + L \bmod m$

$\quad if\ (y_{i,k} < y_{j,k})\ then$

$\qquad \vec{y}_j = visit\ \ left\ descendant\ of\ \vec{y}_j$

$\qquad if\ (left\ descendant\ of\ \vec{y}_j\ is\ Null)\ \ then$

$\qquad\qquad left\ descendant\ of\ \vec{y}_j = \vec{y}_i$

$\quad else$

$\qquad \vec{y}_j = visit\ \ right\ descendant\ of\ \vec{y}_j$

$\qquad if\ (right\ descendant\ of\ \vec{y}_j\ is\ Null)\ \ then$

$\qquad\qquad right\ descendant\ of\ \vec{y}_j = \vec{y}_i$

end

$\forall i, 1 \leq\ i \leq\ Nm\ \ do\ \ begin$

$\vec{y}_j = root$

$begin\ \ from\ \ root\ \ and\ \ recursively\ \forall\ new\ \vec{y}_j, visit$

$\quad sum = \ \ sum + H\left[r - \|\vec{y}_i - \vec{y}_j\|\right]$

$\quad find\ L$

$\quad k = 1 + L \bmod m$

$\quad if\ (y_{j,k} < y_{i,k} - r)\ then$

$\qquad visit\ right\ descendant\ of\ \vec{y}_j$

$\qquad \vec{y}_j = right\ descendant\ of\ \vec{y}_j$

$\qquad sum = \ \ sum + H\left[r - \|\vec{y}_i - \vec{y}_j\|\right]$

$\quad if\ (y_{j,k} > y_{i,k} + r)\ then$

$\qquad visit\ left\ descendant\ of\ \vec{y}_j$

$\qquad \vec{y}_j = left\ descendant\ of\ \vec{y}_j$

$\qquad sum = \ \ sum + H\left[r - \|\vec{y}_i - \vec{y}_j\|\right]$

$\quad if\ (y_{j,k} > y_{i,k} + r\ \ and\ \ \ y_{j,k} < y_{i,k} - r)\ then$

$\qquad visit\ left\ descendant\ of\ \vec{y}_j$

$\qquad \vec{y}_j = left\ descendant\ of\ \vec{y}_j$

$\qquad sum = \ \ sum + H\left[r - \|\vec{y}_i - \vec{y}_j\|\right]$

$$39: \quad visit\ right\ descendant\ of\ \vec{y}_j$$
$$40: \quad \vec{y}_j = right\ descendant\ of\ \vec{y}_j$$
$$41: \quad sum = sum + H\left[r - \| \vec{y}_i - \vec{y}_j \|\right]$$
$$42: end$$
$$43: C(m,r,t) = \frac{1}{N^2} sum$$

Αν Χ είναι το σήμα εισόδου και Ν ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 το σήμα μετασχηματίζεται σε Y στην m διάσταση.

Στη γραμμή 06 εισάγεται η ρίζα του δέντρου. Στις γραμμές 07-20 εισάγονται στο δέντρο όλα τα διανύσματα $\vec{y}_i$. Η διαδικασία γίνεται αναδρομικά (γραμμή 09) για κάθε κόμβο που θα επισκεφθεί το $\vec{y}_i$ στο μονοπάτι που ακολουθεί όπως περιγράφτηκε παραπάνω. L είναι το επίπεδο που βρίσκεται κάθε φορά.

Στις γραμμές 21-43 υπολογίζεται το C(m, r, t). Η διαδικασία εκτελείται αναδρομικά (γραμμή 23) για κάθε κόμβο που επισκέπτεται στο μονοπάτι από τη ρίζα μέχρι και το τελευταίο επίπεδο του δέντρου.

4.3.1. Πολυπλοκότητα του Αλγόριθμου k-d Trees

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος Χ στον R^m τότε t_{read} είναι της τάξης $O(m*N_m)$.

Αν t_{insert} είναι ο χρόνος που απαιτείται για τη κατασκευή του δέντρου τότε t_{insert} είναι της τάξης $O(N_m*\log N_m)$ γιατί το δέντρο είναι δυαδικό.

Αν t_{com} είναι ο χρόνος που χρειάζεται για τον υπολογισμό του C(m, r, t) τότε t_{com} είναι της τάξης $O(m* N_m*\log N_m)$ αφού υπολογίζεται η απόσταση μεταξύ των κόμβων που επισκέπτεται κάθε $\vec{y}_i$ στο μονοπάτι που ακολουθεί.

Συνολικά η πολυπλοκότητα του αλγορίθμου είναι της τάξης O O(m* N_m*log N_m), ενώ η πολυπλοκότητα σε μνήμη είναι O(N_m + N_m*treenodes), όπου treenodes το μέγεθος της δομής του κάθε κόμβου του δέντρου. Η δομή αυτή πρέπει να περιέχει το δείκτη του διανύσματος, το αριστερό και το δεξί παιδί του κόμβου.

4.4. Αλγόριθμος Optimized Box Assisted

Ο Αλγόριθμος αυτός προτάθηκε ως μία βελτιστοποίηση του Αλγόριθμου Box Assisted. Και αυτός προσπαθεί να αποκλείσει άσκοπους υπολογισμούς για ομοιότητα μεταξύ κάθε $\vec{y}_i$ και $\vec{y}_j$, που σίγουρα δεν είναι όμοια, βάζοντας κάθε διάνυσμα του Y σε κάποιο κουτί.

Σύμφωνα με τον Αλγόριθμο Optimized Box Assisted κατασκευάζεται ένα πλέγμα (BOX) ένας m-διάστασης πίνακας δηλαδή από κουτιά (και όχι μία λίστα από κουτιά όπως στον Αλγόριθμο Box Assisted). Κάθε κουτί έχει μέγεθος $R_1 \, x \, R_2 \, x \, R_3 \, x \ldots x \, R_m$, όπου $R_1 = R_2 = R_3 = \ldots = R_m = r$ στον R^m Ευκλείδειο χώρο σύμφωνα με τις παραμέτρους m και r που επιλέχθηκαν. Για κάθε κουτί μόνο τα διανύσματα που βρίσκονται μέσα στα γειτονικά του κουτιά θα περάσουν το τεστ ομοιότητας $H \| \vec{y}_i - \vec{y}_j \|$.

Για παράδειγμα, για διάσταση ενσωμάτωσης m = 3 κατασκευάζεται ένας κύβος που αποτελείται από μικρότερους κύβους, τα λεγόμενα κουτιά. Το μέγεθος του πλέγματος αποτελεί παράμετρο του αλγορίθμου και υπολογίζεται με δοκιμές ή εμπειρικά. Επειδή δεν γίνεται να είναι γνωστό το μέγεθος της κάθε διάστασης του πλέγματος ή να υπολογίζεται εμπειρικά έγινε μία υπόθεση.

Σύμφωνα με την υπόθεση που έγινε το μέγεθος της κάθε διάστασης του πλέγματος (L_k) υπολογίζεται ως εξής:

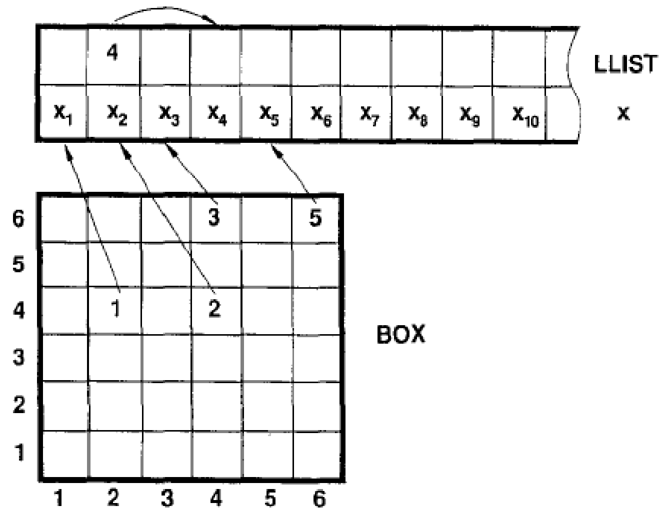
$L_k = |\min_k(Y_{ik}) - \max_k(Y_{ik})| \frac{1}{r}$, δηλαδή L_k = | (μικρότερη τιμή όλων των διανυσμάτων Y_i στη διάσταση k) - (μεγαλύτερη τιμή όλων των διανυσμάτων Y_i στη διάσταση k)| / r, (με στρογγυλοποίηση προς τα πάνω γιατί πρέπει να είναι ακέραια η τιμή).

Ο Αλγόριθμος αυτός αφού το σήμα X αναδομηθεί στον R^m Ευκλείδειο χώρο στο σήμα Y, «βάζει» κάθε διάνυσμα του Y σε κάποιο κουτί. Μάλιστα δεν βάζει καθεαυτό το διάνυσμα $\vec{y}_i$ στο κουτί αλλά μόνο το δείκτη i. Για κάθε $\vec{y}_i$ υπολογίζεται η θέση του κουτιού στο πλέγμα που αντιστοιχεί. Ουσιαστικά υπολογίζει για κάθε διάσταση του πλέγματος τη θέση του διανύσματος, και αυτό γίνεται με τον εξής τρόπο:

$$k = \lceil Int(y_{i,k} / r) \bmod L_k \rceil, \forall k \in [1, m].$$

Μόνο το πρώτο $\vec{y}_i$ που θα μπει στο κουτί αποθηκεύεται στο πλέγμα και μετατρέπεται στην αρχή μιας λίστας από $\vec{y}_i$ που «πέφτουν» στο ίδιο κουτί. Ο Αλγόριθμος αυτός δεν κρατάει τόσες λίστες όσα είναι και τα κουτιά (όπως ο Αλγόριθμος Box Assisted) αλλά, δεσμεύει μόνο μία πίνακα (LLIST) μεγέθους N_m και σε αυτόν τον πίνακα αποθηκεύει όλες τις λίστες από διανύσματα που πέφτουν στο ίδιο κουτί. Ακριβώς επειδή το πλήθος των διανυσμάτων είναι N_m, ο πίνακας αυτός είναι αρκετός για να αποθηκευτούν όλες οι λίστες πιθανά όμοιων διανυσμάτων, γλιτώνοντας έτσι πολύ μνήμη.

Για να γίνει κατανοητός ο τρόπος που μπαίνουν τα διανύσματα στο πλέγμα και στο πίνακα η διαδικασία εξηγείται με ένα παράδειγμα. Στο σχήμα 4.1 φαίνεται ένα παράδειγμα πλέγματος για διάσταση ενσωμάτωσης m = 2. Στο παράδειγμα αυτό το μέγεθος L_1 = 6 και το μέγεθος L_2 = 6.



Σχήμα 4.1 Πλέγμα για διάσταση ενσωμάτωσης m = 2 και ο Πίνακας LLIST

Το πλέγμα και ο πίνακας αρχικοποιούνται με την τιμή -1. Αυτό σημαίνει ότι είναι κενή η θέση στο πλέγμα ή στο πίνακα. Έστω ότι το $\vec{y}_1$ πέφτει στο BOX $_{2,4}$. Αποθηκεύεται ο δείκτης 1 στο πλέγμα και γίνεται η αρχή μιας λίστας. Σύμφωνα με τον αλγόριθμο διατρέχεται ο πίνακας LLIST και στη πρώτη κενή θέση δείχνει η αρχή της νέας λίστας. Έστω ότι το $\vec{y}_2$ πέφτει στο BOX $_{4,4}$. Αποθηκεύεται ο δείκτης 2 στο πλέγμα και γίνεται η αρχή μιας λίστας. Το $\vec{y}_2$ θα δείχνει στη θέση 2 του πίνακα LLIST. Το $\vec{y}_4$ πέφτει και αυτό στο BOX $_{4,4}$. Αν στο BOX $_{4,4}$ έχει ήδη αποθηκευτεί κάποιο διάνυσμα, ακολουθείται η λίστα μέχρι να βρεθεί κενή θέση στο πίνακα. Πάντα το τελευταίο στοιχείο της λίστας θα πρέπει να διατρέχει το πίνακα LLIST και να δείχνει στη πρώτη κενή θέση. Στη θέση που δείχνει το τελευταίο της λίστας μπαίνει το επόμενο διάνυσμα του BOX $_{4,4}$, στο παράδειγμα αυτό το 4. Έτσι το 2 δείχνει στο 4. Αν και άλλο διάνυσμα πέσει στο BOX$_{4,4}$ έστω το 6 το 4 θα δείχνει στο 6 κ.ο.κ.

Αφού μπουν όλα τα διανύσματα του Y στο πλέγμα και στο πίνακα γίνεται τεστ ομοιότητας ανάμεσα σε όλα τα διανύσματα που βρίσκονται σε γειτονικά κουτιά και στο ίδιο το κουτί. Γειτονικά είναι τα 8 κουτιά που βρίσκονται γύρω από το κουτί. Στο παράδειγμα παραπάνω, γείτονες του κουτιού(3,3) είναι τα κουτιά (2,2), (2,3), (2,4), (3,2), (2,4), (3,4), (4,4), και το (4,4). Αυτό επαναλαμβάνεται για κάθε διάσταση του πλέγματος. Για παράδειγμα στο κύβο θα έπρεπε να βρεθούν τα γειτονικά στη διάσταση x τα γειτονικά στη διάσταση y και τα γειτονικά στη διάσταση z.

Ο Αλγόριθμος σε τυπική μορφή είναι ο παρακάτω:

$Nm = N - (m-1)t$

$\forall i, 1 \leq i \leq Nm$

$\quad \forall j, 1 \leq j \leq m \quad do\,begin$

$\qquad Y_{i,j} = X_{i+j*t}$

end

$normalize(Y)$

$\forall k, 1 \leq k \leq m \quad do\,begin$

$\quad L_k = \ulcorner |\min_k(Y_{i,k}) - \max_k(Y_{i,k})| \frac{1}{r} \urcorner$

end

$initialize\,all\,boxes\;from\;BOX$

$\quad box = -1$

$\forall i, 1 \leq i \leq Nm \quad do\,begin$

$\quad LLIST_i = -1$

end

$\forall i, 1 \leq i \leq Nm \quad do\,begin$

$\quad k = \ulcorner Int(y_{i,k} / r) \bmod L_k \urcorner, \forall k \in [1, m]$

$\quad find\;box\;from\;BOX$

$\quad if\;box = -1\;then$

$\qquad box = i,$

$\qquad find\;j\;from\;LLIST : LLIST_j = -1, j \in [1, N_m]$

$\qquad j\;is\,end\;of\;list$

$\quad else$

$\qquad find\;from\;BOX,\;LLIST\;j : j = end\;of\;list,\; j \in [1, N_m]$

$\qquad LLIST_j = i$

$\qquad find\;j\;from\;LLIST : LLIST_j = -1, j \in [1, N_m]$

$\qquad j\;is\,end\;of\;list$

end

$\forall box \in BOX$

$\quad \forall \dim ension\,of\;BOX\;\;do\,begin$

$\qquad find\;all\;j \in neighbouring\;boxes\,of\;BOX$

$\qquad \forall i, 1 \leq i \leq Nm\;\;do\,begin$

$\qquad\quad candidates_i = j$

$\qquad end$

$\qquad \forall i, 1 \leq i \leq Nm$

$\qquad\quad \forall j, 1 \leq j \leq Nm \quad do\,begin$

$\qquad\qquad if\; \| \vec{y}_{candidates_i} - \vec{y}_{candidates_j} \|_m < r\,then$

$\qquad\qquad\quad sum = sum + 1$

$\qquad end$

$\quad end$

end

$CorDim = \frac{1}{N^2} sum$

Αν X είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 το σήμα μετασχηματίζεται σε Y στην m διάσταση.

Στη γραμμή 06 το σήμα κανονικοποιείται ώστε να ξεκινά από το 0. Στις γραμμές 07-09 υπολογίζεται το μέγεθος L του πίνακα BOX για κάθε διάσταση k. O BOX είναι ένας m-διάστασης πίνακας, όπου m η διάσταση ενσωμάτωσης που επιλέχθηκε. Κάθε κελί αυτού του πίνακα είναι ένα κουτί. Στις γραμμές 10-11 κάθε κουτί του πίνακα BOX αρχικοποιείται με τη τιμή -1 που σημαίνει ότι είναι άδειο. Στις γραμμές 12-14 αρχικοποιείται κάθε κελί του πίνακα LLIST με τη τιμή -1 που σημαίνει ότι τα κελιά είναι άδεια.

Στις γραμμές 15-27 κατασκευάζονται οι λίστες με τα διανύσματα που πέφτουν στο ίδιο κουτί. Κάθε $\vec{y}_i$ μπαίνει στο κουτί που πρέπει σύμφωνα με τη διαδικασία που ορίστηκε παραπάνω. Πρώτα πρέπει να υπολογιστεί η θέση του κελιού στο πίνακα BOX, δηλαδή το κουτί, που «πέφτει» το διάνυσμα (γραμμές 16-17). Αμέσως μετά γίνεται έλεγχος αν το κουτί είναι άδειο, δηλαδή αν η τιμή του κελιού στο πίνακα BOX είναι -1 (γραμμή 18). Στις γραμμές 18-27 παρουσιάζεται η διαδικασία που ακολουθείται για να μπουν όλα τα διανύσματα σε κάποιο κουτί.

Στις γραμμές 28-41 γίνονται τα τεστ ομοιότητας των $\vec{y}_i$ και $\vec{y}_j$ που ανήκουν σε γειτονικά κουτιά. Συγκεκριμένα για κάθε κουτί, για κάθε κελί του πίνακα BOX m-διάστασης, πρέπει να βρεθούν σε κάθε διάσταση μέχρι m, τα 8 γειτονικά του κουτιά (γραμμή 30). Τα τεστ ομοιότητας θα γίνουν ανάμεσα σε όλα τα διανύσματα που βρίσκονται μέσα σε όλα τα γειτονικά κουτιά και στο κουτί που εξετάζεται.

Επειδή ο συγγραφέας του Αλγορίθμου δεν διασαφηνίζει τον τρόπο που το κάνει αυτό, έγινε μία υπόθεση. Όλα αυτά τα διανύσματα γράφονται σε ένα πίνακα μεγέθους N_m γιατί η εύρεση του i-στου κόμβου είναι πιο γρήγορη στους πίνακες από ότι στις γραμμικές λίστες (γραμμές 31-33).

Τέλος, στις γραμμές στη γραμμή 34-41 γίνονται τα τεστ ομοιότητας και υπολογίζεται το C(m, r, t).

4.4.1. Πολυπλοκότητα του Αλγόριθμου Optimized Box Assisted

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος X στον R^m τότε t_{read} είναι της τάξης $O(m*N_m)$.

Αν t_{normal} είναι ο χρόνος που απαιτείται για τη κανονικοποίηση του σήματος Y τότε t_{normal} είναι της τάξης $O(m*N_m)$.

Αν $t_{size\ of\ box}$ είναι ο χρόνος που απαιτείται για να υπολογιστεί το μέγεθος του πίνακα box τότε $t_{size\ of\ box}$ είναι της τάξης $O(m*2*N_m)$ αφού ψάχνει να βρει σε κάθε διάσταση τη μικρότερη τιμή του $\vec{y}_i$ και τη μεγαλύτερη τιμή του $\vec{y}_i$ και αυτό γίνεται $\forall i \in [1, N_m]$.

Αν $t_{findbox}$ είναι ο χρόνος που απαιτείται για να βρεθεί σε ποιο κουτί «πέφτει» κάθε $\vec{y}_i$ τότε $t_{findbox}$ είναι της τάξης $O(N_m^2)$. Το πλήθος των διανυσμάτων $\vec{y}_i$ είναι N_m. Για να υπολογιστεί η θέση στο BOX, το κουτί δηλαδή, θέλει χρόνο της τάξης O(1) ενώ για να βρεθεί θέση στο πίνακα LLIST απαιτεί χρόνο της τάξης N_m. Συνολικά, N_m x N_m χρόνο.

Αν t_{com} είναι ο χρόνος που χρειάζεται για τον υπολογισμό του C(m, r, t) τότε t_{com} είναι της τάξης $O(L_1*L_2*\ldots*L_m*N_m^2)$. Αυτό προκύπτει ως εξής: Η διαδικασία απαιτεί $L_1*L_2*\ldots*L_m$ χρόνο για να βρεθούν οι γείτονες του κουτιού σε κάθε διάσταση, ενώ επειδή το πλέγμα υλοποιείται με τη δομή του πίνακα για να βρεθούν οι 8 γείτονες του κουτιού δεν απαιτείται επιπλέον χρόνος. Για να γίνει το τεστ ομοιότητας ανάμεσα σε αυτά τα κουτιά απαιτείται χρόνος N_m^2, αφού τα διανύσματα αποθηκεύονται στο πίνακα candidates μεγέθους N_m.

Συνολικά η πολυπλοκότητα του αλγορίθμου είναι της τάξης $O(L_1*L_2*\ldots*L_m*N_m^2)$. Ωστόσο λόγο της δομής του m-διάστατου πίνακα BOX, όσο αυξάνεται το μέγεθος του πίνακα σε κάθε διάσταση (L_k) ο αλγόριθμος αυτός συμπεριφέρεται χειρότερα από τον Original. Για αυτό στην εργασία αναφέρεται ότι το μέγεθος αυτό είναι παράμετρος που εκτιμάται με δοκιμές. Για καλά διαλεγμένες τιμές των παραμέτρων και $m < 3$, ο συγγραφέας του Αλγόριθμου υποστηρίζει ότι συμπεριφέρεται καλύτερα από τον Original, ενώ οι καλές τιμές των παραμέτρων δεν αναφέρονται.

Η πολυπλοκότητά του σε μνήμη είναι της τάξης $O(L_1 * L_2 * \ldots * L_m + N_m + N_m)$. Όπου $L_1 * L_2 * \ldots * L_m$ είναι το μέγεθος του πλέγματος BOX, N_m θέσεις μνήμης για το σήμα Y και N_m θέσεις μνήμης για το πίνακα LLIST.

ΚΕΦΑΛΑΙΟ 5. ΝΕΟΙ ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΗΣ D_2

5.1. Εισαγωγή

Οι αλγόριθμοι που εξετάζονται στο κεφάλαιο αυτό είναι οι αλγόριθμοι που έχουν προταθεί για τον υπολογισμό της Κατά Προσέγγιση Εντροπίας (ApEn). Η Κατά Προσέγγιση εντροπία είναι άλλη μία μέθοδος που χρησιμοποιείται για την ανάλυση χρονοσειρών που προέρχονται από μη γραμμικά συστήματα [26], [25].

Ουσιαστικά η ApEn ποσολογεί τη μη προβλεψιμότητα της διακύμανσης της χρονοσειράς εισόδου. Μικρή τιμή της ApEn σημαίνει ότι η διακύμανση της χρονοσειράς εισόδου είναι μικρή, άρα και προβλέψιμη. Μεγάλη τιμή της ApEn σημαίνει ότι η διακύμανση της χρονοσειράς εισόδου είναι μεγάλη, άρα μη προβλέψιμη.

Ο πρώτος αλγόριθμος για τον υπολογισμό της ApEn προτάθηκε το 1991 από τον Pincus (Αλγόριθμος Basic) [26] . Έκτοτε η μέθοδος αυτή βρήκε εφαρμογή σε πολλά προβλήματα της βιολογίας, της φυσικής, της ιατρικής ακόμη και των οικονομικών [19], [4], [28].

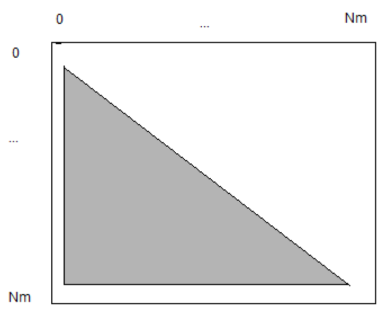
Αργότερα προτάθηκαν και άλλοι αλγόριθμοι για τον υπολογισμό της ApEn. Ο Αλγόριθμος Matrix είναι πολύ απλός και είναι σχεδόν ίδιος με τον Αλγόριθμο Basic. Οι Αλγόριθμοι Bucket Assisted και Sorted Bucket Assisted προτάθηκαν το 2008 και 2009 αντίστοιχα [20], [19], [21] και κατάφεραν να μειώσουν κατά πολύ την $O(N^2)$ πολυπλοκότητα του Αλγόριθμου Basic και του Αλγόριθμου Matrix.

Στη παρούσα εργασία προσαρμόστηκαν οι παραπάνω αλγόριθμοι έτσι ώστε να υπολογίζουν την εκτίμηση της διάστασης συσχέτισης. Στο Κεφάλαιο αυτό παρουσιάζονται οι αναπροσαρμοσμένοι αλγόριθμοι Matrix, Bucket Assisted και Sorted Bucket Assisted. Ο Αλγόριθμος Basic δεν εξετάζεται γιατί είναι παρόμοιος με τον Αλγόριθμο Original.

5.2. Αλγόριθμος Matrix

Ο Αλγόριθμος αυτός υπολογίζει παρόμοια με τον Αλγόριθμο Original την εκτίμηση της διάστασης συσχέτισης. Η διαφορά τους είναι ότι ο Αλγόριθμος Matrix υπολογίζει την τιμή $H\|\vec{y}_i - \vec{y}_j\|$ για κάθε $\vec{y}_i$ και $\vec{y}_j$, και τα αποθηκεύει σε έναν πίνακα S διάστασης N_m x N_m.

Στην συνέχεια διατρέχει μόνο ένα μέρος του S για να υπολογίσει το C(m, r, t). Θεωρεί ότι, αφού η τιμή της $H\|\vec{y}_i - \vec{y}_j\|$ και της $H\|\vec{y}_j - \vec{y}_i\|$είναι ίδια δεν χρειάζεται να το μετρήσουμε στο άθροισμα δύο φορές. Έτσι διατρέχει μόνο το μέρος του πίνακα όπου j < i+1 όπως φαίνεται στο Σχήμα 5.1.



Σχήμα 5.1 Στον πίνακα S μετρώνται οι αποστάσεις μόνο για j < i+1

Ο προσαρμοσμένος Αλγόριθμος Matrix γραμμένος σε τυπική μορφή φαίνεται παρακάτω:

$$
\begin{array}{ll}
01: & Nm = N - (m-1)t \\
02: & \forall i, 1 \leq i \leq Nm \\
03: & \quad \forall j, 1 \leq j \leq m \quad do\,begin \\
04: & \qquad Y_{i,j} = X_{i+j*t} \\
05: & end \\
06: & \forall i, 1 \leq i \leq Nm \\
07: & \quad \forall j, 1 \leq j \leq Nm \quad do\,begin \\
08: & \qquad if \ \| \vec{y}_i - \vec{y}_j \|_m < r\, then \\
09: & \qquad\quad S[i][j] = 1 \\
10: & \qquad else \\
11: & \qquad\quad S[i][j] = 0 \\
12: & end \\
13: & \forall i, 1 \leq i \leq Nm \\
14: & \quad \forall j, 1 \leq j \leq i+1 \quad do\,begin \\
15: & \qquad sum = sum + S[i][j] \\
16: & end \\
17: & CorDim = \dfrac{2}{N(N-1)} sum
\end{array}
$$

Αν X είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 μετασχηματίζεται το σήμα στην m διάσταση. Στις γραμμές 06-12 φτιάχνεται ο πίνακας S όπως παραπάνω, και τέλος στις γραμμές 13-17 υπολογίζεται το C(m, r, t).

5.2.1. Πολυπλοκότητα του Αλγόριθμου Matrix

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος X στον R^m τότε t_{read} είναι της τάξης $O(m*N_m)$.

Αν t_S είναι ο χρόνος που χρειάζεται για να φτιαχτεί ο πίνακας S τότε t_S είναι της τάξης $O(N_m{}^2)$

Αν t_{com} είναι ο χρόνος που χρειάζεται για τον υπολογισμό του C(m, r, t) τότε t_{com} είναι της τάξης $O(m*N_m^2)$ αφού υπολογίζεται η απόσταση μεταξύ κάθε πιθανού ζεύγους y_i και y_j.

Συνολικά η πολυπλοκότητα του αλγορίθμου είναι της τάξης $O(m*N_m^2)$, ενώ η πολυπλοκότητα σε μνήμη είναι $O(N_m + N_m^2)$ λόγω και της επιβάρυνσης του πίνακα S.

5.3. Αλγόριθμος Bucket Assisted

Ο Αλγόριθμος Bucket Assisted ακολουθεί μια τελείως διαφορετική προσέγγιση από αυτές που είχαν προταθεί μέχρι τώρα.

Αρχικά η χρονοσειρά εισόδου x, $x=(x_1,x_2,x_3,x_4,...,x_{N-1},x_N)$, όπου N ο συνολικός αριθμός σημείων της χρονοσειράς, αναδομείται στον R^m όπως αναφέρεται στην §2.1:

$X = \left\{\vec{x}_i\right\}_{i=1}^{N_m}$ και $\vec{x}_i=\left\{\vec{x}_{i+t},\vec{x}_{i+2t},...,\vec{x}_{i+(m-1)t}\right\}$.

Αμέσως μετά υπολογίζεται ένα νέο σήμα X = {X_i}, όπου $i\in[1,N_m]$ και

$$X_i = x_{i+t}, x_{i+2t}, \ldots, x_{i+(m-1)t} = \sum_{j=0}^{N_m} x_{i+jt} .$$

Με βάση τα παραπάνω αν $\|\vec{x}_i-\vec{x}_j\|<r$ τότε:

$$\Leftrightarrow |x_i - x_j| < r, |x_{i+t} - x_{j+t}| < r, \ldots, |x_{i+(m-1)t} - x_{j+(m-1)t}| < r$$
$$\Leftrightarrow |x_i - x_j| + |x_{i+t} - x_{j+t}| + \ldots + |x_{i+(m-1)t} - x_{j+(m-1)t}| < mr$$

Και αφού $|a+b|\leq|a|+|b|$ τότε:

$$|[x_{i+t}, x_{i+2t}, \ldots, x_{i+(m-1)t}] - [x_{j+t}, x_{j+2t}, \ldots, x_{j+(m-1)t}]| < mr$$
$$\Leftrightarrow |X_i - X_j| < mr$$

Άρα δύο διανύσματα του νέου σήματος X θεωρούνται όμοια αν $|X_i - X_j| < mr$ ή αλλιώς αν $\| X_i - X_j \| < r$ και $\| X_i - X_j \|$ είναι η νόρμα μεγίστου ή η Ευκλείδεια νόρμα.

Έστω ένα σύνολο από κάδους (B). $B = \left\{ B_1, B_2, ..., B_{h_N} \right\}$ και $h_N = \lceil \max(X_i) / r \rceil$. Όλοι οι κάδοι έχουν μέγεθος r. Κάθε διάνυσμα X_i αντιστοιχίζεται σε ένα κάδο. Το διάνυσμα X_i μπαίνει στο κάδο B_h, $h = \lceil X_i / r \rceil$. Έτσι αν το διάνυσμα X_i που αντιστοιχεί στο $\vec{x}_i$ πέσει στο κάδο B_h, τότε το διάνυσμα X_j που αντιστοιχεί στο $\vec{x}_j$ και ισχύει $\| \vec{x}_i - \vec{x}_j \| < r$ θα βρίσκεται σε έναν από τους κάδους :

$$B_{h-m}, B_{h-m+1}, B_{h-m+2}, \ldots, B_h, B_{h+1}, B_{h+2}, \ldots, B_{h+m}$$

Με τον τρόπο αυτό δεν υπολογίζονται κάποιες αποστάσεις $\| \vec{x}_i - \vec{x}_j \|$ που σίγουρα ισχύει: $\| \vec{x}_i - \vec{x}_j \| > r$. Επιπλέον ο αλγόριθμος λαμβάνει υπ’ όψιν του ότι: αν $\| \vec{x}_i - \vec{x}_j \| < r$ τότε και $\| \vec{x}_j - \vec{x}_i \| < r$ οπότε η απόσταση μεταξύ $\vec{x}_i$ και $\vec{x}_j$ υπολογίζεται μόνο μια φορά.

Ο προσαρμοσμένος Bucket Assisted περιγράφεται σε τυπική μορφή παρακάτω:

01 : $Nm = N - (m-1)t$
02 : $\forall i, 1 \le i \le Nm$
03 : $\quad \forall j, 1 \le j \le m \quad do\,begin$
04 : $\qquad X_i = X_i + x_{i+j*t}$
05 : end
06 : $\forall i, 1 \le i \le Nm \quad do\ begin$
07 : $\quad normalize(X_i)$
08 : end
09 : $\forall i, 1 \le i \le Nm \quad do\ begin$
10 : $\quad C[i] = 1$
11 : end
12 : $Nb = \lceil \max(X_i) / r \rceil$
13 : $\forall i, 1 \le i \le Nm \quad do\,begin$
14 : $\quad p = \lceil X_i / r \rceil$
15 : $\quad fill\ \ bucket_p\ \ with\ i$
16 : end
17 : $\forall ib, 1 \le ib \le Nb$
18 : $\quad \forall jb > 0, ib - m \le jb \le ib - 1 \ \ do\ begin$
19 : $\qquad \forall i, i \in bucket_{ib}$
20 : $\qquad\quad \forall j, (j \in bucket_{ib}\ \ or\ j \in bucket_{jb}, and\ i > j)$
21 : $\qquad\qquad if\ \| \vec{x}_i - \vec{x}_j \|_m < r\,then$
22 : $\qquad\qquad\quad C[i] = C[i] + 1$
23 : $\qquad\qquad\quad C[j] = C[j] + 1$
24 : $\qquad\qquad endif$
25 : end
26 : $\forall i, 1 \le i \le Nm \quad do\ begin$
27 : $\quad sum = sum + C[i]$
28 : end
29 : $CorDim = \dfrac{2}{N(N-1)} sum$

Αν x είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 μετασχηματίζεται το σήμα στην m διάσταση και φτιάχνεται το νέο σήμα X. Στις γραμμές 06-08 το σήμα κανονικοποιείται ώστε να ξεκινά από το 1. Κάθε C[i] κρατάει πόσα X_j, $j \in [1, Nm]$, είναι όμοια με το X_i . Στις γραμμές 09-11 αρχικοποιούνται τα C[i] = 1. Στη γραμμή 12 υπολογίζεται ο αριθμός των κάδων που προκύπτει από τη χρονοσειρά εισόδου. Στις γραμμές 13-16 κάθε X_i μπαίνει στους κάδους. Στη λίστα του κάθε κάδου κρατάμε το δείκτη i και όχι το ίδιο το X_i. Κάθε λίστα έχει μέγεθος N_m. Στις γραμμές 17-25 υπολογίζονται τα C[i] και C[j]. Για κάθε X_i που ανήκει στον κάδο ib τα X_j που θα υπολογιστεί αν είναι όμοια (γραμμή 21) βρίσκονται ή μέσα στον ίδιο κάδο ή στους m προηγούμενούς του (γραμμή 18). Στη γραμμή 29 υπολογίζεται το C(m, r, t).

5.3.1. Πολυπλοκότητα του Αλγόριθμου Bucket Assisted

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος x στον R^m και για τη μετατροπή του στο σήμα X τότε $t_{read} = m*N_m + N_m$ και είναι της τάξης $O(m*N_m)$.

Αν t_{normal} είναι ο χρόνος που απαιτείται για τη κανονικοποίηση του σήματος X τότε t_{normal} είναι της τάξης $O(N_m)$.

Αν t_{fill} είναι ο χρόνος που χρειάζεται για να μπουν τα X_i κάδους τότε $t_{fill} = O(N_m)$.

Αν N_{points} είναι ο μέσος όρος των διανυσμάτων σε κάθε κάδο τότε ο χρόνος που χρειάζεται κάθε X_i για να υπολογίσει το C[i] $O(m*N_{points})$ αφού ψάχνει και στους m προηγούμενους κάδους. Το C[i] υπολογίζεται για όλα τα i $\in [1, Nm]$. Άρα αν t_{com} είναι ο χρόνος που χρειάζεται για να υπολογιστούν όλα τα C[i] άρα και το C(m, r, t), t_{com} είναι της τάξης $O(m*N_{points} * N_m)$ αφού υπολογίζεται η απόσταση μεταξύ κάθε πιθανού ζεύγους y_i και y_j.

Συνολικά η πολυπλοκότητα σε χρόνο του αλγορίθμου είναι της τάξης $O(m*N_{points}*N_m)$ το οποίο είναι πολύ μικρότερο από $O(N_m^2)$ συνήθως. Η πολυπλοκότητα σε μνήμη είναι $O(N_m + N_b*N_m)$ αφού υπάρχουν N_b κάδοι συνολικά. Και για κάθε κάδο δεσμεύεται μία λίστα που περιέχει το πολύ N_m διανύσματα.

5.4. Αλγόριθμος Sorted Bucket Assisted

Ο Αλγόριθμος αυτός είναι μια επέκταση του Αλγόριθμου Bucket Assisted. Σύμφωνα με τον αλγόριθμο ακολουθείται και πάλι η διαδικασία που περιγράφεται στη §5.3.

Για να αποκλειστούν όμως και άλλοι υπολογισμοί για την ομοιότητα μεταξύ των X_i και X_j τα διανύσματα $\vec{x}_k$, που αντιστοιχούν στους δείκτες k που αποθηκεύουμε σε κάθε κάδο, ταξινομούνται με βάση τη πρώτη συντεταγμένη τους. Ο αλγόριθμος ταξινόμησης που χρησιμοποιήθηκε είναι ο quicksort.

Εφόσον τα διανύσματα είναι ταξινομημένα σε κάθε κάδο για να βρεθούν τα υποψήφια X_j που θα υπολογιστεί η ομοιότητα με κάθε X_i, εκτελείται δυαδική αναζήτηση. Μία φορά για να βρεθεί ποια είναι η χαμηλότερη θέση j που για τη πρώτη συντεταγμένη του $\vec{x}_j$ το $\|\vec{x}_j - \vec{x}_i\| < r$. Και μία φορά για να βρεθεί ποια είναι η υψηλότερη θέση j που για τη πρώτη συντεταγμένη του $\vec{x}_j$ το $\|\vec{x}_j - \vec{x}_i\| < r$.

Μόνο τα X_j που βρίσκονται ανάμεσα σε αυτές τις θέσης εξετάζονται για ομοιότητα με το X_i. Με τον τρόπο αυτό αποκλείονται οι υπολογισμοί πολλών ακόμη ζευγών $\vec{x}_i, \vec{x}_j$. Τέλος ο αλγόριθμος αυτός για να αποκλείσει και άλλους άσκοπους υπολογισμούς ομοιότητας μεταξύ ζευγών $\vec{x}_i, \vec{x}_j$ σπάει τους κάδους με βάση μια παράμετρο split factor.

Με τον τρόπο αυτό το νέο μέγεθος του κάθε κάδου γίνεται split factor / παλαιό μέγεθος και ως εκ τούτου εξετάζονται για ομοιότητα τα X_j που βρίσκονται στους m*split factor προηγούμενους κάδους. Έτσι πολύ λιγότερα X_i μπαίνουν σε κάθε κάδο και πολύ λιγότερες ομοιότητες μεταξύ $\vec{x}_i, \vec{x}_j$ υπολογίζονται.

Ο προσαρμοσμένος Sorted Bucket Assisted περιγράφεται σε τυπική μορφή παρακάτω:

$$
\begin{array}{l}
Nm = N - (m-1)t \\
\forall i, 1 \le i \le Nm \\
\quad \forall j, 1 \le j \le m \quad do\, begin \\
\qquad X_i = X_i + x_{i + j*t} \\
end \\
\forall i, 1 \le i \le Nm \quad do\ begin \\
\quad normalize(X_i) \\
end \\
\forall i, 1 \le i \le Nm \quad do\ begin \\
\quad C[i] = 1 \\
end \\
Nb = \lceil split * \max(X_i) / r \rceil \\
\forall i, 1 \le i \le Nm \quad do\, begin \\
\quad p = \lceil X_i / r \rceil \\
\quad fill\ \ bucket_p\ \ with\ i \\
end \\
\forall ib, 1 \le ib \le Nb \\
\quad sort(bucket_{ib})\, asc \\
\quad \forall jb > 0,\ ib - m * split \le jb \le ib - 1 \ \ do\ begin \\
\qquad \forall i, i \in bucket_{ib} \\
\qquad\quad find\, candidates\ j,\ \ j \in candidates\ \ if \\
\qquad\qquad (j \in bucket_{ib}\ or\ j \in bucket_{jb}, and\ i > j) \\
\qquad\qquad and\, (x_j - r \le x_i \le x_j + r) \\
\qquad \forall j, j \in candidates \\
\qquad\quad if\ \| \vec{x}_i - \vec{x}_j \|_m < r\, then \\
\qquad\qquad C[i] = C[i] + 1 \\
\qquad\qquad C[j] = C[j] + 1 \\
\qquad\quad endif \\
end \\
\forall i, 1 \le i \le Nm \quad do\ begin \\
\quad sum = sum + C[i] \\
end \\
CorDim = \dfrac{2}{N(N-1)} sum
\end{array}
$$

Αν x είναι το σήμα εισόδου και N ο αριθμός των σημείων του σήματος τότε στην γραμμή 01 υπολογίζεται το μέγεθος του παραγόμενου σήματος σύμφωνα με τις τιμές των παραμέτρων m και t που έχουν επιλεχθεί. Στις γραμμές 02-05 μετασχηματίζεται το σήμα στην m διάσταση και φτιάχνεται το νέο σήμα X. Στις γραμμές 06-08 το σήμα κανονικοποιείται ώστε να ξεκινά από το 1. Κάθε C[i] κρατάει πόσα X_j, $j \in [1, Nm]$, είναι όμοια με το X_i .

Στις γραμμές 09-11 αρχικοποιούνται τα C[i] = 1. Στη γραμμή 12 υπολογίζεται ο αριθμός των κάδων που προκύπτει από τη χρονοσειρά εισόδου. Στις γραμμές 13-16 κάθε X_i μπαίνει στο κάδο που πρέπει. Στη λίστα του κάθε κάδου κρατάμε το δείκτη i και όχι το ίδιο το X_i. Η λίστα του κάθε κάδου έχει μέγεθος Nm, Στις γραμμές 17-30 υπολογίζονται τα C[i] και C[j]. Στη γραμμή 18 ταξινομούνται τα διανύσματα $\vec{x}_i$ του κάθε κάδου (σε αύξουσα σειρά σύμφωνα με τη πρώτη συντεταγμένη τους) με βάση τους δείκτες i που έχουν αποθηκευτεί σε κάθε κάδο.

Για κάθε X_i που ανήκει στον κάδο ib τα X_j που θα υπολογιστεί αν είναι όμοια (γραμμή 21) βρίσκονται ή μέσα στον ίδιο κάδο ή στους m*split προηγούμενούς του (γραμμή 19) και ισχύει: $x_j - r \leq x_i \leq x_j + r$. Αν low είναι το j εκείνο που $x_j - r \leq x_i$ και high είναι το j εκείνο που $x_i \leq x_j + r$ τότε candidates είναι τα low < j < high. Στη γραμμή 31 υπολογίζεται το C(m, r, t).

5.4.1. Πολυπλοκότητα του Αλγόριθμου Sorted Bucket Assisted

Η πολυπλοκότητα του αλγόριθμου είναι:

Αν t_{read} είναι ο χρόνος που απαιτείται για το μετασχηματισμό του αρχικού σήματος x στον R^m και για τη μετατροπή του στο σήμα X τότε $t_{read} = m*N_m + N_m$ και είναι της τάξης $O(m*N_m)$.

Αν t_{normal} είναι ο χρόνος που απαιτείται για τη κανονικοποίηση του σήματος X τότε t_{normal} είναι της τάξης $O(N_m)$.

Αν t_{fill} είναι ο χρόνος που χρειάζεται για να μπουν τα X_i κάδους τότε $t_{fill} = O(N_m)$.

Αν t_{sort} είναι ο χρόνος που χρειάζεται για να ταξινομηθούν τα X_i κάδους τότε $t_{sort} = O(N_{points} \log N_{points})$, όπου N_{points} είναι ο μέσος όρος των διανυσμάτων σε κάθε κάδο.

Αν $N_{candidat}$ ο αριθμός των υποψήφιων X_j για το τεστ ομοιότητας τότε ο χρόνος που χρειάζεται κάθε X_i για να βρεθούν τα candidates$_j$ είναι $2^* N_{points} \log N_{points}$ λόγω της δυαδικής αναζήτησης. Για να υπολογιστεί το C(m, r, t) απαιτείται χρόνος $m^*N_{candidat}{}^*N_m$ αφού για κάθε διάνυσμα γίνεται το τεστ ομοιότητας μόνο με τα candidates διανύσματα. Άρα αν t_{com} είναι ο χρόνος που χρειάζεται για να υπολογιστούν όλα τα C[i] άρα και το C(m, r, t), t_{com} είναι της τάξης $O(m^*N_{candidat}{}^*N_m)$.

Συνολικά η πολυπλοκότητα σε χρόνο του αλγορίθμου είναι της τάξης O $O(m^*N_{candidat}{}^*N_m)$ το οποίο συνήθως είναι πολύ μικρότερο από $O(N^2)$ και μικρότερο από το $O(m^*N_{points}{}^*N_m)$ του Bucket Assisted. $N_{candidat} < N_{points}$ συνήθως.

Η πολυπλοκότητα σε μνήμη είναι $O(N_m + N_b{}^*N_m)$ αφού υπάρχουν N_b κάδοι συνολικά. Και για κάθε κάδο δεσμεύεται μία λίστα που περιέχει το πολύ N_m διανύσματα.

ΚΕΦΑΛΑΙΟ 6. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1. Εισαγωγή

Οι Αλγόριθμοι γράφτηκαν σε προγράμματα σε γλώσσα C, ενώ τα πειράματα πραγματοποιήθηκαν σε υπολογιστή SUN Blade1500 64-bit, 1.5 GHz Ultra Sparc IIIi επεξεργαστή και 8GB DDR.

Μελετήθηκαν οι Αλγόριθμοι των Κεφαλαίων 4 και 5 για χρονοσειρές εισόδου που προέρχονται από ηλεκτροκαρδιογράφηματα (N = 4096), και από τυχαία κατασκευασμένο σήμα που οι τιμές του ανήκουν στο [0,2) και N = 80000.

Εξετάστηκε ο χρόνος που χρειάζεται η CPU να υπολογίσει το C(m, r, t) για παραμέτρους, m = 1, 2, 3, 4, 5, t = 1, και r = 0.1 , 0.15, 0.2, 0.25, 0.3.

Συγκεκριμένα μελετήθηκαν τα εξής:

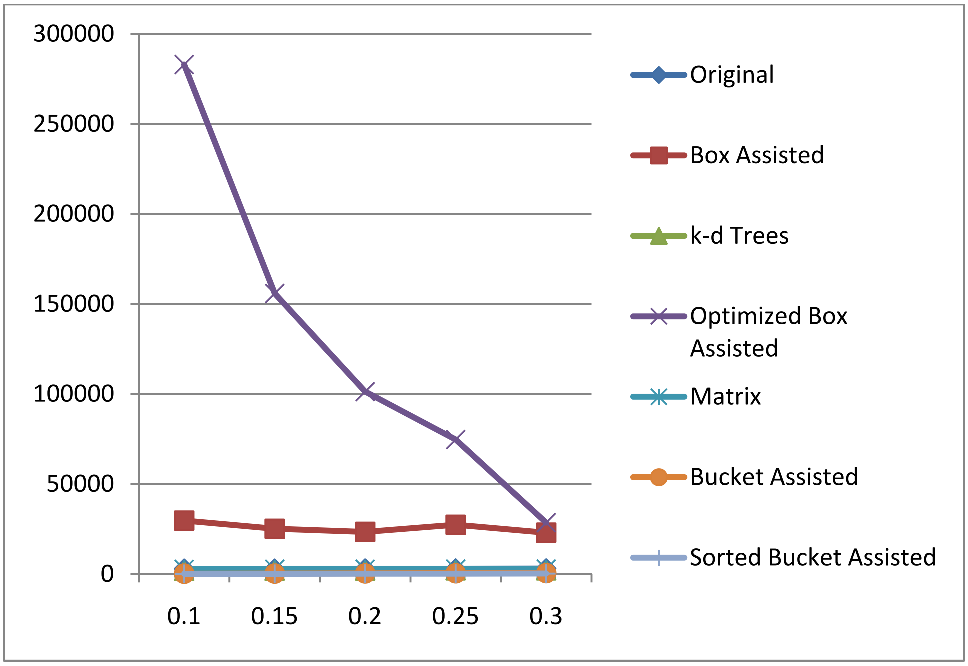
- Πόσο αλλάζει η πολυπλοκότητα κάθε αλγόριθμου αυξάνοντας την απόσταση r για διάσταση m σταθερή σε μία τιμή.

- Πόσο αλλάζει η πολυπλοκότητα κάθε αλγόριθμου αυξάνοντας την διάσταση ενσωμάτωσης m και κρατώντας την απόσταση r σταθερή σε μία τιμή.
- Πόσο αλλάζει η πολυπλοκότητα κάθε αλγόριθμου αυξάνοντας τον αριθμό των σημείων της χρονοσειράς εισόδου. Από 20.000 σημεία έως 80.000 σημεία.
- Πως συμπεριφέρονται οι αλγόριθμοι χρησιμοποιώντας για το τεστ ομοιότητας την Ευκλείδεια νόρμα και τη νόρμα μεγίστου.
- Πόσο διαφορετικά είναι τα αποτελέσματα αν δεν παρθεί σαν είσοδος όλη η χρονοσειρά εισόδου. Εναλλακτικά μπορεί να κατασκευαστεί μία χρονοσειρά εισόδου (μεγέθους N/2) που παίρνει ένα σημείο από την αρχική χρονοσειρά και το επόμενο το αφήνει. Αυτό εξετάστηκε ως εναλλακτική στις λύσεις του Theiler που πρότεινε να παίρνετε σαν χρονοσειρά εισόδου, η χρονοσειρά από την αρχή μέχρι τη μέση για να μειωθεί το N.
- Ποιος είναι ο πιο γρήγορος Αλγόριθμος για την εύρεση της εκτίμησης της διάστασης συσχέτισης. Πως κατατάσσονται οι Αλγόριθμοι με βάση τις επιδόσεις τους στο χρόνο που απαιτούν για να υπολογίσουν την D_2.
- Ποιοι Αλγόριθμοι έχουν μεγάλο κόστος σε μνήμη.

6.2. Παρουσίαση των αποτελεσμάτων

6.2.1. Αύξηση της παραμέτρου r

Στο Σχήμα 6.1 παρουσιάζονται τα αποτελέσματα σε msec των μετρήσεων που έγιναν στο χρόνο χρησιμοποίησης της CPU για όλους τους Αλγόριθμους. Ο χρόνος είναι η μικρότερη τιμή για όλα τα m και t=1. Τα αποτελέσματα για αυτές τις τιμές των παραμέτρων φαίνονται στο Παράρτημα στον Πίνακα Π.1.

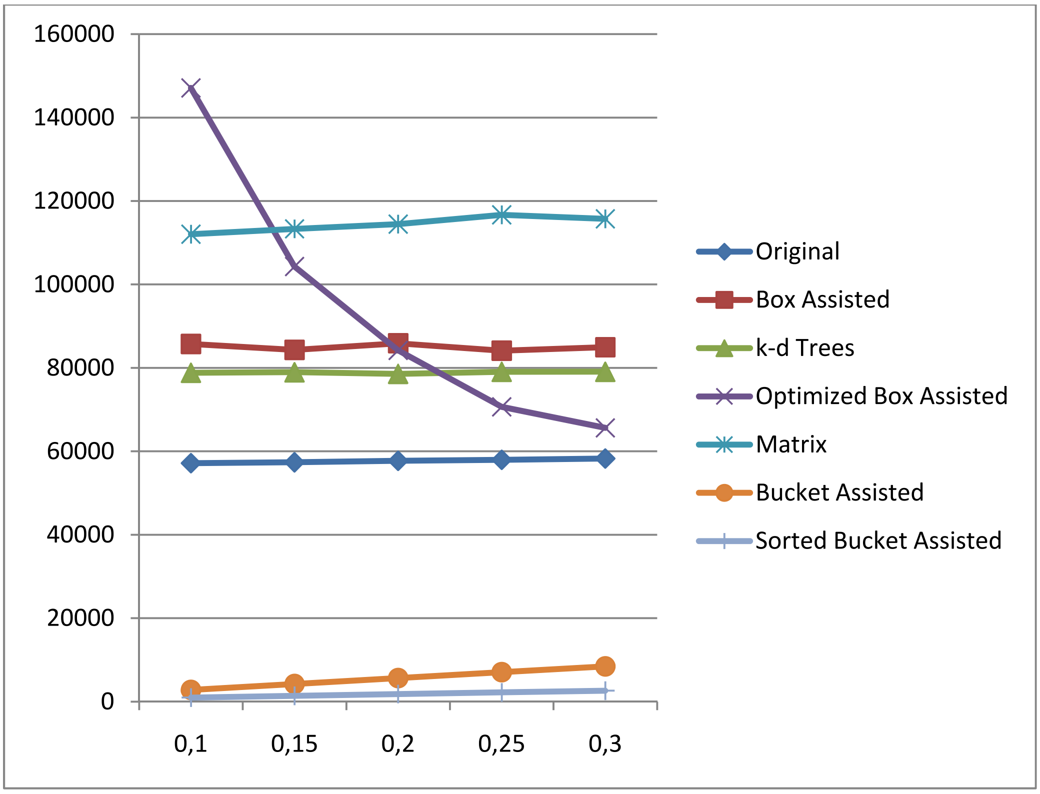


Σχήμα 6.1 Αποτελέσματα σε msec για N=4096, t=1, από ηλεκτροκαρδιογράφημα

Η παράμετρος r δεν επηρεάζει την απόδοση των αλγορίθμων εκτός του Αλγόριθμου Optimized Box Assisted. Αυτό συμβαίνει διότι το μέγεθος του πλέγματος εξαρτάται πολύ από την παράμετρο r. Όσο μεγαλώνει το r τόσο λιγότερα κουτιά έχει. Συνεπώς λιγότερα γειτονικά κουτιά να ελέγξει για ομοιότητα ανάμεσα στα διανύσματά τους.

Στο Σχήμα 6.2 φαίνονται τα αποτελέσματα για τη μικρότερη τιμή του χρόνου χρησιμοποίησης της CPU για όλα τα m καθώς αυξάνεται η παράμετρος r. Εδώ τα δεδομένα προέρχονται από το τυχαία κατασκευασμένο σήμα για N = 20000. Τα αποτελέσματα για αυτές τις τιμές των παραμέτρων φαίνονται στο Παράρτημα στον Πίνακα Π.2.

Στα Σχήματα 6.1 και 6.2 φαίνεται ο χρόνος που χρειάζεται για να υπολογίσει το C(m, r, t) ο κάθε Αλγόριθμος για τυχαία παραγμένο σήμα μεγέθους N = 20000. Στο Σχήμα 6.2 φαίνεται ξεκάθαρα πόσο πιο γρήγορος είναι ο Αλγόριθμος Sorted Bucket Assisted σε σχέση με τους άλλους Αλγόριθμους που εξετάζονται για μεγάλο N, όπου και έχουν νόημα οι Αλγόριθμοι.

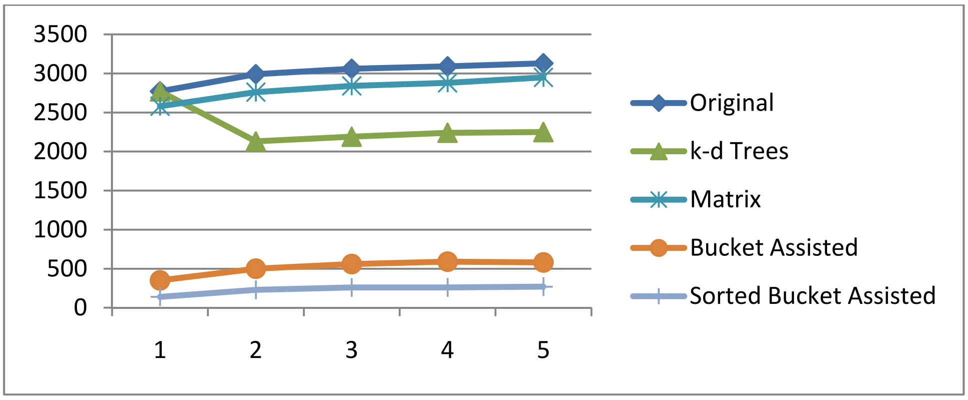


Σχήμα 6.2 Αποτελέσματα σε msec για N=20000, t=1, από τυχαία κατασκευασμένο σήμα

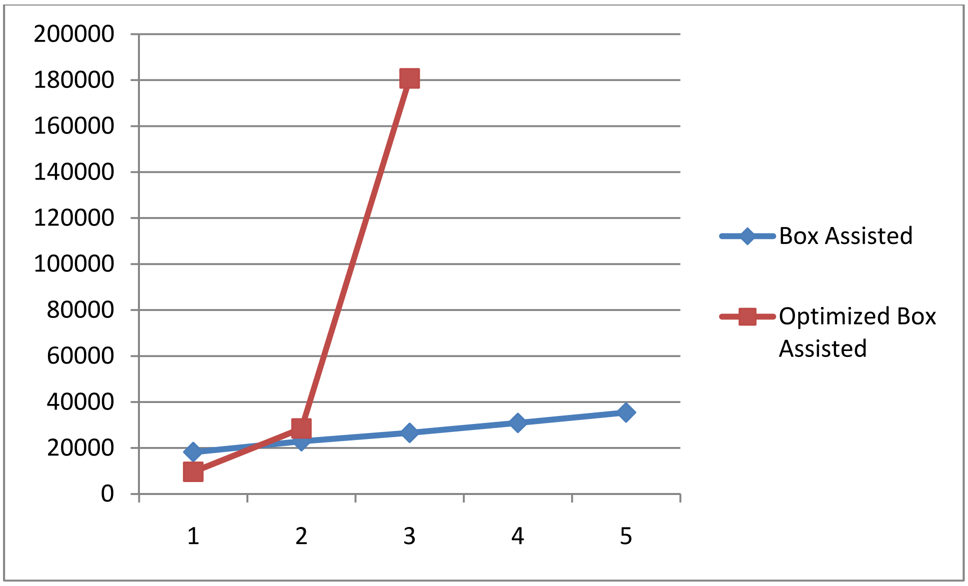
Ακόμη και ο Αλγόριθμος k-d Trees που είναι της τάξης (ONlogN) φαίνεται να συμπεριφέρεται χειρότερα από τον Original και τον Box Assisted. Αυτό συμβαίνει γιατί το k-d Tree που κατασκευάζεται δεν είναι ισοζυγισμένο και το δέντρο μπορεί να εκφυλιστεί. Κάτι τέτοιο συνέβη σε αυτή τη περίπτωση. Και σε αυτό το Σχήμα μόνο ο Αλγόριθμος Optimized Box Assisted φαίνεται να επηρεάζεται πάρα πολύ από την αύξηση της παραμέτρου r.

6.2.2. Αύξηση της παραμέτρου m

Στο Σχήμα 6.3 και 6.4 φαίνονται τα αποτελέσματα σε msec για r = 0,3 καθώς η διάσταση ενσωμάτωσης m μεγαλώνει. Το σήμα εισόδου προέρχεται από ηλεκτροκαρδιογράφημα και το N = 4096. T Εφόσον μόνο για τον Αλγόριθμο Optimized Box Assisted παίζει σημαντικό ρόλο στον υπολογισμό του C(m, r, t) η τιμή της απόστασης r εξετάζονται όλοι οι Αλγόριθμοι για r = 0,3. Για αυτή τη τιμή του r ο Αλγόριθμος Optimized Box Assisted κάνει το λιγότερο χρόνο για να υπολογίσει το C(m, r, t). Στο Σχήμα 6.4 φαίνονται οι Αλγόριθμοι Box Assisted και Optimized Box Assisted, ενώ στο σχήμα 6.3 φαίνονται όλοι οι άλλοι Αλγόριθμοι. Τα αποτελέσματα για αυτές τις τιμές των παραμέτρων φαίνονται στο Παράρτημα στον Πίνακα Π.3.



Σχήμα 6.3 Αποτελέσματα σε msec καθώς αυξάνεται το m για N=4096, t=1, r=0,3 από ηλεκτροκαρδιογράφημα

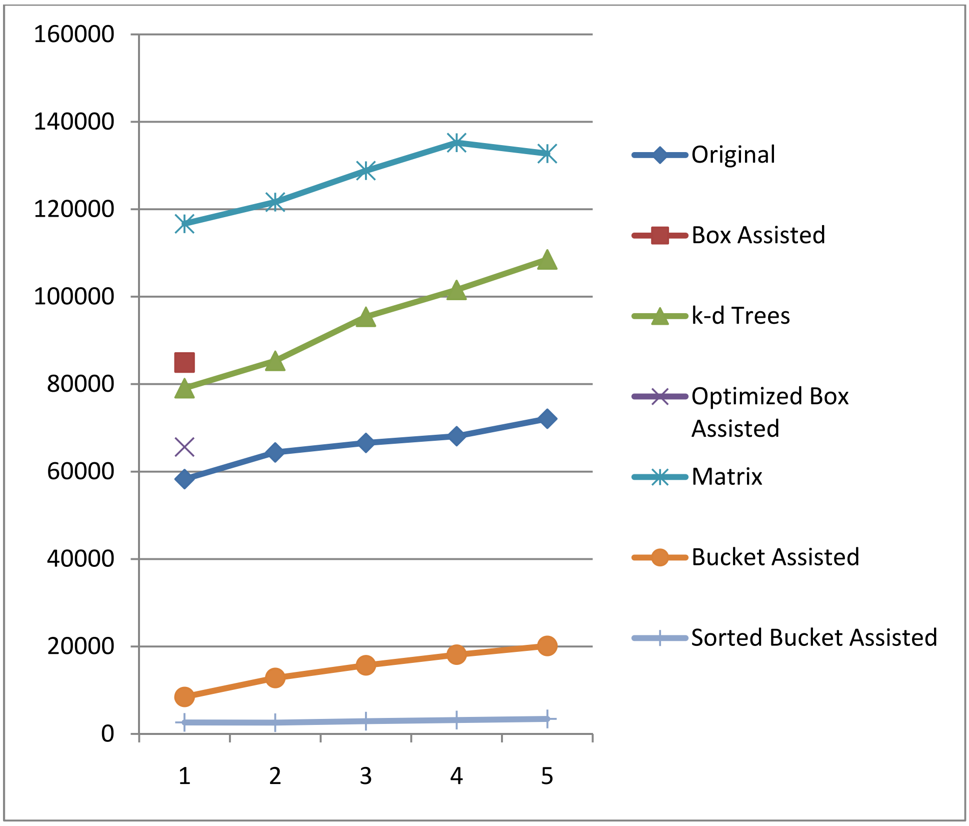


Σχήμα 6.4 Αποτελέσματα σε msec καθώς αυξάνεται το m για N=4096, t=1, r=0,3 από ηλεκτροκαρδιογράφημα

Από τα παραπάνω σχήματα προκύπτει το συμπέρασμα ότι όσο αυξάνεται η διάσταση ενσωμάτωσης m όλοι οι Αλγόριθμοι παραμένουν σχεδόν σταθεροί (με μικρές διακυμάνσεις) εκτός από τον Αλγόριθμο Optimized Box Assisted. Αυτό συμβαίνει διότι στον Αλγόριθμο αυτό κατασκευάζεται πλέγμα m-διάστασης και έτσι ο χρόνος που απαιτείται για να υπολογίσει το C(m, r, t) αυξάνεται εκθετικά με την αύξηση του m. Για αυτό το λόγο ο Αλγόριθμος αυτός δεν εξετάζεται για m > 3.

Και οι δύο Αλγόριθμοι της οικογένειας των Αλγορίθμων κουτιού ωστόσο, κάνουν πολύ περισσότερο χρόνο να υπολογίσουν το C(m, r, t) για N = 4096, συγκριτικά με τους άλλους Αλγόριθμους.

Στο Σχήμα 6.5 φαίνονται τα αποτελέσματα σε msec για r = 0,3 καθώς η διάσταση ενσωμάτωσης m μεγαλώνει. Το σήμα εισόδου είναι τυχαία κατασκευασμένο και το N = 20000. Οι Αλγόριθμοι Box Assisted και Optimized Box Assisted δεν μπορούσαν να υπολογίσουν το C(m, r, t) για N=20000 και m >1 για αυτό και δεν φαίνονται στο σχήμα. Οι Αλγόριθμοι αυτοί για m > 1 και N > 10000 σημεία περίπου έχουν πρόβλημα με τη μνήμη. Αυτό συμβαίνει λόγω της δομής που χρησιμοποιούν όπως περιγράφηκε στην §4.2 και §4.4.

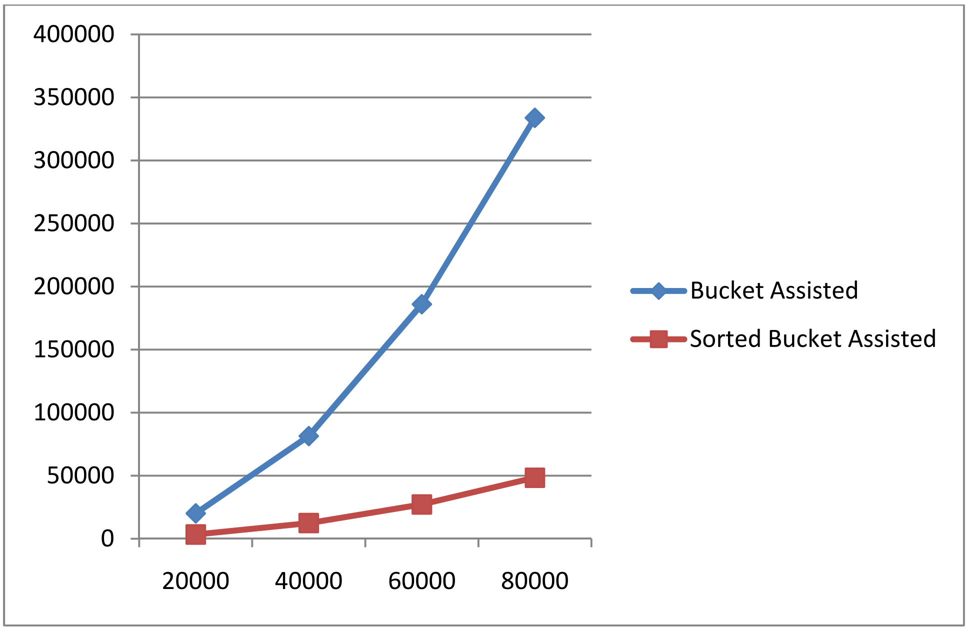


Σχήμα 6.5 Αποτελέσματα σε msec καθώς αυξάνεται το m για N=20000, t=1, r=0,3 από τυχαία κατασκευασμένο σήμα

Και στο Σχήμα 6.5 φαίνεται ότι όλοι οι Αλγόριθμοι παρουσιάζουν διακύμανση, της τάξης των 10000 μέχρι 30000 msec, καθώς αυξάνεται το m, ενώ και πάλι ο Αλγόριθμος Sorted Bucket Assisted δεν έχει τέτοια διακύμανση και απαιτεί πολύ λιγότερο χρόνο. Τα αποτελέσματα για αυτές τις τιμές των παραμέτρων φαίνονται στο Παράρτημα στον Πίνακα Π.4.

6.2.3. Αύξηση της παραμέτρου N

Εξαιτίας του γεγονότος ότι οι Αλγόριθμοι Bucket Assisted και Sorted Bucket Assisted είναι γρηγορότεροι κατά πολύ από τους άλλους αλγόριθμους παρουσιάζονται μόνο αυτοί για μεγαλύτερα N. Και οι δύο αυτοί αλγόριθμοι κάνουν περισσότερο χρόνο για να υπολογίσουν το C(m, r, t) για m=5 και r=0.3. Στο Σχήμα 6.6 φαίνονται τα αποτελέσματα των δύο αυτών Αλγορίθμων καθώς το N αυξάνεται, με τιμές m=5, r=0.3, αφού για m <5 και r<0.3 απαιτούν λιγότερο χρόνο για τον υπολογισμό του C(m, r, t). Τα αποτελέσματα για αυτές τις τιμές των παραμέτρων φαίνονται στο Παράρτημα στον Πίνακα Π.5.



Σχήμα 6.6 Αποτελέσματα σε msec για m=3, r=0.3 καθώς αυξάνεται το N για τυχαία κατασκευασμένο σήμα εισόδου

Ο Αλγόριθμος Sorted Bucket Assisted υπολογίζει ακόμη και για N=80000 σε λιγότερο από 50 sec το C(m, r, t). Στο Σχήμα φαίνεται πως η ταξινόμηση των κάδων μείωσε κατά 300sec περίπου τον υπολογισμό των C(m, r, t).

6.2.4. Νόρμα Μεγίστου και Ευκλείδεια Νόρμα

Όλοι οι Αλγόριθμοι εξετάστηκαν, με παραμέτρους m=1, r=0.3, t=1 και N=20000 για τυχαία κατασκευασμένο σήμα εισόδου, κατά τον υπολογισμό του C(m, r, t) χρησιμοποιώντας την Ευκλείδεια νόρμα και τη νόρμα μεγίστου.

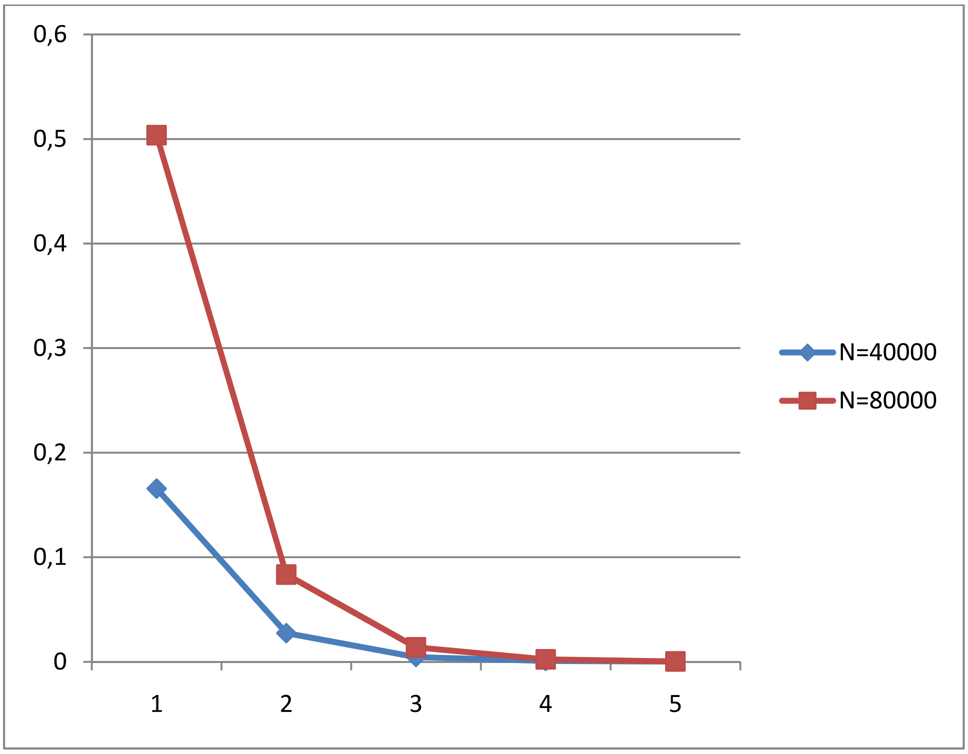
Τα αποτελέσματα έδειξαν ότι η νόρμα μεγίστου είναι πιο αποδοτική ενώ η Ευκλείδεια Νόρμα απαιτεί περισσότερο χρόνο λόγω των περισσότερων πράξεων που χρειάζονται για τον υπολογισμό της. Εξαιτίας του γεγονότος αυτού όλα τα πειράματα που έγιναν και που παρουσιάζονται στο κεφάλαιο αυτό υπολογίζουν το C(m, r, t) χρησιμοποιώντας τη νόρμα μεγίστου.

6.2.5. Μείωση της χρονοσειράς εισόδου

Ως λύση στο πρόβλημα που υπάρχει κατά τον υπολογισμό του C(m, r, t) σε μεγάλα συστήματα δηλαδή για $D_2 > 5$, ο Theiler πρότεινε έναν διπλό υπολογισμό: έναν χρησιμοποιώντας όλο το σύνολο σημείων με μια μικρή κοντινή απόσταση και έναν χρησιμοποιώντας ένα μικρό πακέτο σημείων έχοντας υπόψη όλες τις αποστάσεις. Ωστόσο τα αποτελέσματα θεωρήθηκαν αναξιόπιστα [17].

Εναλλακτικά για να μειωθεί το πλήθος N των σημείων του σήματος εισόδου εξετάστηκε πόσο αλλάζει η τιμή του C(m, r, t) αν από το σήμα εισόδου κατασκευαστεί ένα άλλο σήμα μεγέθους N/2, όπου τα σημεία του σήματος αυτού θα είναι το 1^o, το 3^o , το 5^o , …., το i-στο + 2 σημείο του αρχικού σήματος.

Εξετάστηκαν οι Αλγόριθμοι για m=1,…,5, r=0.3, t=1 για τυχαία κατασκευασμένο σήμα εισόδου, μεγέθους N =80000. Τα αποτελέσματα φαίνονται στο Σχήμα 6.7 ενώ οι τιμές όλων των παραμέτρων φαίνονται στο Πίνακα Π.7 του Παραρτήματος.



Σχήμα 6.7 Η τιμή του C(m, r, t) για το αρχικό σήμα (N=80000) και για το κατασκευασμένο σήμα (40000) όσο το m αυξάνεται, r=0.3 και t=1

Όπως φαίνεται στο παραπάνω σχήμα, για m < 3 οι τιμές του C(m, r, t) παρουσιάζουν μία διαφορά μέχρι και 0,4. Εξαιτίας της μεγάλης ευαισθησίας της μεθόδου υπολογισμού της $D_2 = \lim_{N\to\infty} \frac{\log(C(m,r,t))}{\log(r)}$ και του γεγονότος ότι το C(m, r, t) πρέπει να υπολογίζεται κάθε φορά αυξάνοντας τη διάσταση ενσωμάτωσης m ακόμη και μία τέτοια διαφορά οδηγεί σε λανθασμένα αποτελέσματα.

6.2.6. Συμπεράσματα

Τα συμπεράσματα που προέκυψαν από τη μελέτη αυτή επιβεβαιώνουν τα προβλήματα που αναφέρθηκαν κατά την Εισαγωγή. Ο υπολογισμός της εκτίμησης της διάστασης συσχέτισης (D_2) είναι δύσκολος για μεσαία ή υψηλά συστήματα όπου $D_2 > 5$. Αυτό συμβαίνει γιατί ο υπολογισμός της σε αυτά τα συστήματα απαιτεί μεγάλο αριθμό διανυσμάτων (N) που θα χρησιμοποιηθούν.

Οι Αλγόριθμοι που είχαν προταθεί μέχρι τώρα δεν μπορούν να υπολογίσουν αποδοτικά την εκτίμηση της διάστασης συσχέτισης λόγω της αυξημένης πολυπλοκότητας τους σε χρόνο και μνήμη. Όσο το N αυξάνεται και η διάσταση ενσωμάτωσης m αυξάνεται οι Αλγόριθμοι αυτοί απαιτούν πολύ περισσότερο χρόνο.

Ο Αλγόριθμος Sorted Bucket Assisted όμως, φαίνεται να μπορεί να ανταπεξέλθει σε λίγα sec και χωρίς ρύθμιση επιπλέον παραμέτρων ακόμη και για μεγάλα συστήματα. Μέχρι τώρα είναι η καλύτερη λύση που έχει προταθεί για τον υπολογισμό της D_2 αποδοτικά, σύμφωνα με την ακριβή λύση και με πολύ απλό τρόπο.

ΑΝΑΦΟΡΕΣ

[1] Barkley H. J., Gevrais F., Olivain J., Quemeneur,A., and Truc A., "Estimate of the correction for tokamak plasma turbulence", Plasma Physics and Controlled Fusion, Great Britain, Vol. 30(3), pp. 217-234, 1988.

[2] Bhandarkar S. M., Chirravuri S., and Whitmire D., "Analysis of Heart Rate Variability on a Massively Parralel Processor",IEEE, 1996.

[3] Bingham S. and Kot M., "Multidimensional trees, range searhing, and a correlation dimension algorthm of reduced complexity", PHYSICS LETTERS A, Vol. 140(6), pp. 327-330, 1989.

[4] Bruhn J., Ropcke H., Rehberg B., Bouillon T., and Hoeft A., "*Electroencephalogram Approximate Entropy Correctly Classifies the Occurrence of Burst Suppression Pattern as Increasing AnestheticDrug Effect*", Anesthesiology, Vol. 93, pp. 981-985, 2000.

[5] Bueno-Orovio A. and Perez-Garcia V., "Enhanced box and prism assisted algorithms for computing the correlation dimension", Chaos, Solitons and Fractals, Vol. 34, pp. 509-518, 2007.

[6] Casaleggio A. and Corana A., "Correlation Dimension Estimation from Electrocardiograms", Chaos, Solrmns & Fractals, Vol. 5, pp. 713-726, 1995.

[7] Corana A., "Adaptive box-assisted algorithm for correlation-dimension estimation", PHYSICAL REVIEW E, Vol. 62(6), pp. 7872-7881, 2000.

[8] Corana A., "Computing the correlation dimension on a network of workstations", Concurrency: Pract.Exp, Vol. 10(10), pp. 737-762, 1998.

[9] Corana A., "Practical aspects Parallel computation of the correlation dimension from a time series", Parallel Computing, Vol. 25, pp. 639-666, 1999.

[10] Corana A., Milleri L., and Sciarretta S., "A Box Assisted Parallel Algorithm to Compute the Correlation Dimension from a Time Series",1994.

[11] Craig C., Neilson R. D., and Penman J., "The use of correlation monitoring of systems with clearance ", Journal of Sound and Vibration, Vol. 231(1), pp. 1-17, 2002.

[12] Grassberger P., "An optimized box-assisted algorithm for fractal dimensions", PHYSICS LETTERS A, Vol. 148(1,2), pp. 63-68, 1990.

[13] Grassberger P. and Procaccia I., Physica D, Vol. 189, 1983.

[14] Grassberger P. and Procaccia I., "Measuring the strangeness of strange attractors", Phys.D, Vol. 9, pp. 189-208, 1983.

[15] Grassberger P., Schreiber T., and Schaffrath C., "None linear time sequence analysis", Int.J.Bifurcation Chaos, Vol. 1, pp. 521-547, 1991.

[16] Kagan Yan Y., "Earthquake spatial distribution: the correlation dimension ", NG, Vol. 43B, pp. 1158-2006.

[17] Kantz H. and Schreiber T., "Nonlinear Time series Analysis ", Cambridge University Press, 1997.

[18] Lamberts J., Van de Broek P. L. C., Bener L., Van Egmond J., Dirksen R., and Coenen A. M. L., "Correlation Dimension of the Human Electroencephalogram Corresponds with Cognitive Load", Neuropsychobiology, Vol. 41, pp. 149-153, 2000.

[19] Manis G., "Fast computation of approximate entropy", Computer Methods and Programs In Biomedicine, Vol. 91, pp. 48-54, 2008.

[20] Manis G. and Nikolopoulos S., "Speeding up the Computation of Approximate Entropy", IFMBE Proceedings, Vol. 16, pp. 782-788, 2007.

[21] George Manis and Roberto Sassi, "Sample Entropy and Approximate Entropy: How to compute them fast,", University of Ioannina, Dept. of Computer Science, Technical Report, 2009.

[22] Miao X., He W., Yang H., and Fang, L., " A Speedy Algorithm for Estimating the Correlation Dimension ", IJMPB, Vol. 17(22-24), pp. 4284-4289, 2003.

[23] Miyata M., Burioka N., Sako T., Suyama H., Fukuoka Y., Tomita K., Higami S., and Shimizu E., "A short daytime test using correlation dimension for respiratory movement in OSAHS", Eur Respir J, Vol. 23, pp. 885-890, 2004.

[24] Nikolopoulos S., Journal of Biomedical Informatics, Vol. 36, pp. 202-217, 2003.

[25] Pawelzik K. and Schuster H. G., "Jeneralized dimensions and entropies from a measured time series", Phys.Rev.A, Vol. 35, pp. 481-484, 1987.

[26] Pincus S., "Approximate Entropy (ApEn) as a complexity measure ", CHAOS, Vol. 5, pp. 110-117, 1995.

[27] Pincus S., "Approximate entropy as a measure of system complexity", Proc.Nati.Acad.Sci.USA, Vol. 88, pp. 2297-2301, 1991.

[28] Roy P. N. S. and Narth S. K., "Precursory correlation dimensions for three great earthquakes", Current Science, Vol. 93(11), pp. 1522-1529, 2007.

[29] Rukhin A. L., "Approximate Entropy for Testing Randomness", J.Appl.Prob., Israel, Vol. 39, pp. 88-100, 2000.

[30] Smith L. A., PHYSICS LETTERS A, Vol. 1331988.

[31] Takens F., "Detecting strange attractors in turbulence", Lect.Notes Math, Vol. 898, pp. 366-381, 1981.

[32] Theiler J, "Efficient algorithm for estimating the correlation from a set of discrete points", Phys.Rev.A, Vol. 36(9), pp. 4456-4462, 1987.

ΠΑΡΑΡΤΗΜΑ

Πίνακας Π.6.1 Αποτελέσματα όλων των αλγορίθμων για τη μικρότερη τιμή σε msec)του χρόνου χρησιμοποίησης της CPU για όλα τα m, t = 1, N=4096. Το σήμα προέρχεται από ηλεκτροκαρδιογράφημα

r	Original	Box Assisted	k-d Trees	Optimized Box Assisted	Matrix	Bucket Assisted	Sorted Bucket Assisted
0.1	2880	29650	1780	282990	2620	150	80
0.15	2910	25120	1870	155780	2670	230	110
0.2	2940	23350	1950	101280	2690	320	150
0.25	2970	27340	2050	74670	2740	410	190
0.3	2990	22920	2050	28500	2760	500	230

Πίνακας Π.6.2 Αποτελέσματα όλων των αλγορίθμων για τη μικρότερη τιμή σε msec του χρόνου χρησιμοποίησης της CPU για όλα τα m, t = 1, N=20000. Το σήμα είναι τυχαία κατασκευασμένο

r	Original	Box Assisted	k-d Trees	Optimized Box Assisted	Matrix	Bucket Assisted	Sorted Bucket Assisted
0,1	57160	85720	78840	147100	112050	2820	980
0,15	57370	84300	78970	104290	113320	4240	1400
0,2	57720	85900	78570	84200	114460	5640	1830
0,25	57970	84110	79070	70660	116680	7070	2220
0,3	58270	84950	79080	65620	115720	8460	2630

Πίνακας Π.6.3 Αποτελέσματα σε msec καθώς αυξάνεται το m για N=4096, t=1, r=0,3 από ηλεκτροκαρδιογράφημα

m	1	2	3	4	5
Original	2770	2990	3060	3090	3130
Box Assisted	18210	22920	26570	30910	35460
k-d Trees	2770	2130	2190	2240	2250
Optimized Box Assisted	9640	28500	180770		
Matrix	2580	2760	2840	2880	2950
Bucket Assisted	350	500	560	590	580
Sorted Bucket Assisted	140	230	260	260	270

Πίνακας Π.6.4 Αποτελέσματα σε msec για N=20000, t=1, r=0.3 για τυχαία κατασκευασμένο σήμα

m	Original	Box Assisted	k-d Trees	Optimized Box Assisted	Matrix	Bucket Assisted	Sorted Bucket Assisted
1	58270	84950	79080	65620	116680	8460	2630
2	64380	Memory Error	85340	Memory Error	121640	12790	2560
3	66570	Memory Error	95410	Memory Error	128830	15690	2910
4	68090	Memory Error	101560	Memory Error	135220	18120	3170
5	72100	Memory Error	108540	Memory Error	132730	20110	3430

Πίνακας Π.6.5 Αποτελέσματα σε msec για m=5, r=0.3 καθώς το N αυξάνεται για τυχαία κατασκευασμένο σήμα εισόδου

N	Bucket Assisted	Sorted Bucket Assisted
20000	20110	3430
40000	81460	12480
60000	185980	27220
80000	333880	48410

Πίνακας Π6.6 Τιμή του C(m,r,t) για το αρχικό (N=80000) και το κατασκευασμένο σήμα εισόδου (N=40000) καθώς αυξάνεται το m και r=0.3, t=1

m	C(m,r,t) για το κατασκευασμένο σήμα (N=40000)	C(m,r,t) για το αρχικό σήμα (N=80000)
1	0,165638	0,503628
2	0,027447	0,08346
3	0,004566	0,013857
4	0,000778	0,002327
5	0,00015	0,000417

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Η Κατερίνα Μπάκα γεννήθηκε στη Θεσσαλονίκη το 1983. Μεγάλωσε στη Θεσσαλονίκη και έζησε εκεί μέχρι το 2001. Από το 2001 κατοικεί στα Ιωάννινα. Σπούδασε στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων από όπου και αποφοίτησε το 2007.

Από το 2007 φοιτά στο Μεταπτυχιακό Πρόγραμμα Σπουδών του Τμήματος Πληροφορικής.