Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Γενετικός Αλγόριθμος Ταξινόμησης Genetic AIRS**
Ονοματεπώνυμο Φοιτητή	**Δημήτρης Μαθιουδάκης**
Πατρώνυμο	**Εμμανουήλ**
Αριθμός Μητρώου	**ΜΠΠΛ/ 10032**
Επιβλέπων	**Τσιχριντζής Γεώργιος, Καθηγητής**

Ημερομηνία Παράδοσης **Ιανουάριος 2013**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Ευχαριστίες

Η παρούσα εργασία αποτελεί τη Διατριβή μου στα πλαίσια των μεταπτυχιακών σπουδών στο Τμήμα Πληροφορική και στο Πρόγραμμα Μεταπτυχιακών Σπουδών Πληροφορικής του Πανεπιστημίου Πειραιά. Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ Γ.Τσιχριντζή που μου εμπιστεύθηκε και πρότεινε ένα τόσο ενδιαφέρον θέμα δίνοντας μου έτσι την ευκαιρία να το μελετήσω σε βάθος. Στη συνέχεια, θα ήθελα να ευχαριστήσω θερμά τον Διδάκτορα Διονύσιο Σωτηρόπουλο για την ουσιαστική βοήθεια που μου παρείχε και για τις ώρες που αφιέρωσε πάνω στο θέμα της διατριβής μου.

Ένα πολύ μεγάλο ευχαριστώ στην οικογένειά μου και όλους τους φίλους που με στήριξαν και στηρίζουν σε όλες μου τις προσπάθειες και με βοήθησαν με τον τρόπο τους στην ολοκλήρωση της παρούσας εργασίας.

Περίληψη

Στην παρούσα μεταπτυχιακή διατριβή αναπτύσσεται μία υβριδική μέθοδος ταξινόμησης που βασίζεται από την μία πλευρά στις αρχές και τις θεωρίες των Τεχνητών Ανοσοποιητικών Συστημάτων και από την άλλη πλευρά στις αρχές των Γενετικών Αλγορίθμων. Εμπνευσμένος από τον αλγόριθμο ταξινόμησης AIRS (Artificial Immune Resource System), που συγκαταλέγεται στους αποδοτικότερους αλγορίθμους ταξινόμησης, καθώς και στις αποδοτικές τεχνικές των Γενετικών Αλγορίθμων υλοποιούμε ένα υβριδικό αλγόριθμο τον Genetic AIRS. Στη συνέχεια διερευνήσαμε την συμπεριφορά του αλγορίθμου αυτού σε σχέση με τον αλγόριθμο AIRS και παραθέτουμε τα πλεονεκτήματα και τα μειονεκτήματα που παρουσιάζει. Η κύρια διαφοροποίηση των αλγορίθμων αυτών εκτός της διαφορετικής υλοποίησης έγκειται στο γεγονός ότι ο AIRS εκτελείται τοπικά ενώ ο Genetic AIRS σφαιρικά. Η εργασία αυτή χωρίζεται σε δύο μέρη και έχει ως στόχο την θεωρητική υποστήριξη του αλγορίθμου αυτού στο πρώτο μέρος ενώ στο δεύτερο την εφαρμογή του σε διάφορα σύνολα δεδομένων. Πιο συγκεκριμένα η εργασία είναι δομημένη ως εξής:

- **Κεφάλαιο 1** στο κεφάλαιο αυτό παρουσιάζονται τα βασικά συστατικά της Μηχανικής Μάθησης με έμφαση στη μάθηση με επίβλεψη και στην ταξινόμηση, καθώς και στους βιολογικά εμπνευσμένους υπολογισμούς.
- **Κεφάλαιο 2** στο κεφάλαιο αυτό παρουσιάζουμε τις αρχές και θεωρίες των γενετικών αλγορίθμων.
- **Κεφάλαιο 3** στο κεφάλαιο αυτό παρουσιάζουμε τα Τεχνητά Ανοσοποιητικά Συστήματα.
- **Κεφάλαιο 4** στο κεφάλαιο αυτό παρουσιάζουμε τον Αλγόριθμο AIRS.
- **Κεφάλαιο 5** στο κεφάλαιο αυτό παρουσιάζουμε αλγόριθμο που προτείνουμε τον Genetic AIRS.
- **Κεφάλαιο 6** στο κεφάλαιο αυτό παρουσιάζουμε τα πειραματικά αποτελέσματα
- **Κεφάλαιο 7** στο κεφάλαιο αυτό βρίσκεται ο επίλογος.

Abstract

This thesis presents an hybrid classification method based on one hand on theories and principles of Artificial Immune Systems and on the other hand on Genetic Algorithm techniques. The inspiration came from AIRS algorithm(Artificial Immune Resource System), one of the most accurate classification algorithms, and from Genetic Algorithm methods, so we developed one hybrid algorithm Genetic AIRS. Next, we investigated the behavior of the algorithm created, with respect to the AIRS algorithm and present the advantages and disadvatages. The main difference of these two algorithms beside the fact of different implementation is the fact that AIRS executes topically and Genetic AIRS spherically. The first part of this thesis is the theoritical base of the algorithm and the second part is the application on various data sets.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Μηχανική Μάθηση

1.1 Εισαγωγή

Η μηχανική μάθηση αποτελεί ένα ξεχωριστό πεδίο έρευνας και κατατάσσεται στο πεδίο της Τεχνητής Νοημοσύνης. Έχει σκοπό την ανάπτυξη μεθόδων έτσι ώστε να οδηγήσουν τα υπολογιστικά συστήματα στην απόκτηση γνώσης. Με άλλα λόγια, κύριος στόχος της μηχανικής μάθησης είναι η δημιουργία μηχανών ικανών να μάθουν και να αξιοποιήσουν την προηγούμενη γνώση τους σε νέα προβλήματα. Έχουν αναπτυχθεί διάφοροι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιούν συγκεκριμένες τεχνικές μάθησης, οι οποίοι έχουν εμφανίσει μεγάλη εμπορική επιτυχία. Για προβλήματα όπως η αναγνώριση φωνής (Speech Recognition) και η εξόρυξη γνώσης (Data Mining) από μεγάλες βάσεις δεδομένων, η χρήση αλγορίθμων μηχανικής μάθησης είναι πλέον ευρέως διαδεδομένη και αποτελεσματική. Μερικά προγράμματα που έχουν υλοποιηθεί είναι αυτό της αναγνώρισης λέξεων στο [28, 27] και [38], η πρόβλεψη του βαθμού ανάρρωσης ενός ασθενή που πάσχει από πνευμονία, η αναγνώριση της παράνομης χρήσης πιστωτικής κάρτας [21], καθώς και προγράμματα ικανά να μαθαίνουν να παίζουν τάβλι σε επίπεδο παγκόσμιων πρωταθλητών [36]. Ένας ορισμός που έχει δοθεί για την μηχανική μάθηση από τον Mitchell [30] είναι ο εξής:

Ορισμός 1 *Ένα πρόγραμμα υπολογιστή θεωρείται ότι μαθαίνει από εμπειρία E , σε σχέση με κάποια κατηγορία εργασιών T και μετρική αποτίμησης P , εάν η απόδοση στις εργασίες του T, όπως μετριέται από το P, βελτιώνεται με την εμπειρία E*

Οι αλγόριθμοι μηχανικής μάθησης μπορούν να χωριστούν σε δύο κατηγορίες, ανάλογα με την διαδικασία μάθησης που χρησιμοποιούν και το είδος της εξόδου που παράγουν. Οι δύο βασικές κατηγορίες είναι η Μάθηση με Επίβλεψη (Supervised Learning) και η Μάθηση χωρίς Επίβλεψη (Unsupervised Learning).

1.2 Μάθηση με Επίβλεψη

Μάθηση με επίβλεψη είναι η διαδικασία μηχανικής μάθησης κατά την οποία το σύστημα μαθαίνει επαγωγικά μία συνάρτηση που αποτελεί έκφραση του μοντέλου των δεδομένων εκπαίδευσης. Ένα σύστημα μάθησης με επίβλεψη, εκπαιδεύεται αρχικά σε ένα σύνολο παραδειγμάτων εκπαίδευσης. Κάθε παράδειγμα του συνόλου αυτού χαρακτηρίζεται από μια κατηγορία στην οποία ανήκει. Η διαδικασία κατά την οποία εφαρμόζεται μια μέθοδος μάθησης με επίβλεψη σε ένα πρόβλημα είναι η ακόλουθη:

- *Καθορισμός του τύπου του συνόλου εκπαίδευσης*. Ο χρήστης θα πρέπει να καθορίσει το είδος των δεδομένων που θα χρησιμοποιήσει για τον καθορισμό του συνόλου εκπαίδευσης.
- *Συλλογή του συνόλου εκπαίδευσης*. Το σύνολο εκπαίδευσης θα πρέπει να είναι αντιπροσωπευτικό του προβλήματος που καλείται να λύσει.
- *Καθορισμός των χαρακτηριστικών που εισάγονται στον αλγόριθμο*. Συνήθως τα χαρακτηριστικά αυτά αναπαριστώνται από ένα πίνακα χαρακτηριστικών.
- *Καθορισμός της δομής του αλγόριθμου μάθησης*. Ο χρήστης θα πρέπει να καθορίσει ποιον αλγόριθμο μάθησης θα χρησιμοποιήσει για την παραγωγή των αποτελεσμάτων.

- *Πειραματικός σχεδιασμός*. Στο στάδιο αυτό καθορίζεται ο τρόπος με τον οποίο θα πραγματοποιηθούν τα πειράματα. Υπάρχουν διάφοροι μέθοδοι όπως (Validation Set) ή (Cross-Validation)
- *Υπολογισμός της ακρίβειας του αλγόριθμου μάθησης*.

Στη μάθηση με επίβλεψη διακρίνονται τα εξής είδη προβλημάτων:

- **Πρόβλημα Ταξινόμησης**: αφορά τη δημιουργία μοντέλων πρόβλεψης διακριτών τάξεων
- **Πρόβλημα Παλινδρόμησης**: αφορά τη δημιουργία μοντέλων πρόβλεψης αριθμητικών τιμών

1.2.1 Ταξινόμηση

Το πρόβλημα της ταξινόμησης είναι ένα πρόβλημα εντοπισμού της κατηγορίας στην οποία ανήκει μία καινούργια παρατήρηση. Με άλλα λόγια, ταξινόμηση είναι το πρόβλημα της ανάθεσης ενός αντικειμένου σε μία ή περισσότερες προκαθορισμένες κατηγορίες (κλάσεις). Ορισμένα παραδείγματα ταξινόμησης είναι ο εντοπισμός spam emails, η πρόβλεψη καρκινικών κυττάρων χαρακτηρίζοντας τα ως καλοήθη ή κακοήθη και πολλά άλλα.

Πιο συγκεκριμένα η ταξινόμηση είναι μία διεργασία η οποία αποδίδει τον χαρακτηρισμό αληθές ή ψευδές, ανάλογα αν το υπό εξέταση αντικείμενο ανήκει ή όχι σε μία συγκεκριμένη κλάση. Η διεργασία αυτή δέχεται ως είσοδο μία συλλογή από εγγραφές που αποτελούνται από ένα σύνολο γνωρισμάτων(features). Ένα από τα γνωρίσματα είναι η κλάση στην οποία ανήκει η κάθε εγγραφή. Στόχος είναι η υλοποίηση ενός μοντέλου για την ταξινόμηση μιας νέας εγγραφής στην κλάση που ανήκει. Η υλοποίηση του μοντέλου αυτού είναι μια διαδικασία εκμάθησης μιας συνάρτησης στόχου f που απεικονίζει κάθε σύνολο γνωρισμάτων x σε μια από τις προκαθορισμένες κλάσεις y. Το σύνολο δεδομένων εισόδου της συνάρτησης, χωρίζεται σε ένα σύνολο εκπαίδευσης (Training Set) και ένα σύνολο ελέγχου (Test Set). Το σύνολο εκπαίδευσης χρησιμοποιείται στην διαδικασία υλοποίησης του μοντέλου ταξινόμησης, ενώ το σύνολο ελέγχου για να το επικυρώσει. Μια πιο αυστηρή διατύπωση της διαδικασίας της ταξινόμησης είναι η ακόλουθη:

Έστω ένα σύνολο εκπαίδευσης $X_{train} = \{(d_1, y_1), \cdots, (d_n, y_n)\}$ με $d_i = \{x_1, \cdots, x_k\}$ είναι τα χαρακτηριστικά του i-στου διανύσματος διάσταση k. Το $y_i \in L$ με $L = \{L_1, \cdots L_n\}$ είναι οι κλάσεις στις οποίες ανήκουν τα διανύσματα των χαρακτηριστικών. Στόχος της διαδικασίας της ταξινόμησης είναι η εύρεση μιας συνάρτησης μέσω της διαδικασίας της μάθησης από το x στο y. Η συνάρτηση αυτή περιγράφεται $D : \mathbb{R}^n \longrightarrow L$ και ονομάζεται ταξινομητής. Ο ταξινομητής αυτός ελέγχεται για την ακρίβεια του στο σύνολο ελέγχου $X_{test} = \{d_1, \cdots, d_n\}$ με $X_{training} \cap X_{test} = \oslash$

Μέτρα Αξιολόγησης

- **Πίνακας σύγχυσης- Confusion Matrix**

  Ο πίνακας σύγχυσης είναι ένας πίνακας που παρουσιάζει συνοπτικά την απόδοση ενός αλγορίθμου ταξινόμησης. Κάθε στήλη του πίνακα παρουσιάζει τις προβλέψεις κάθε κλάσης ενώ κάθε γραμμή παρουσιάζει το σύνολο των δεδομένων της κάθε κλάσης. Η κύρια διαγώνιος του πίνακα περιέχει τα στοιχεία που αντιστοιχούν στις σωστές προβλέψεις ενώ στις άλλες θέσεις περιέχει τις λάθος προβλέψεις. Για παράδειγμα στον πίνακα (1.1) a είναι οι παρατηρήσεις που ταξινομήθηκαν στην κλάση Α και είναι από την κλάση Α, ενώ b είναι οι παρατηρήσεις που ταξινομήθηκαν στην κλάση Β αλλά ανήκουν στην κλάση Β.

Πίνακας 1.1: Πίνακας Σύγχυσης

		Προβλέψεις	
		Κλάση Α	Κλάση Β
Πραγματικές Τιμές	Κλάση Α	a	b
	Κλάση Β	c	d

- **Ακρίβεια**

Ως ακρίβεια ορίζουμε την αναλογία των σωστών προβλέψεων ως προς τις συνολικές προβλέψεις.

$$Accuracy = \frac{N_{correct}}{N_{all}}$$

Το $N_{correct}$ είναι το πλήθος των σωστών προβλέψεων και N_{all} το πλήθος των συνολικών προβλέψεων.

- **Ορθότητα**
  Ορθότητα ορίζεται η αναλογία σωστών προβλέψεων ανά κλάση.

$$precision_c = \frac{{N^c}_{correct}}{{N^c}_{all}}$$

Αξιολόγηση Ταξινομητή

Η αξιολόγηση του ταξινομητή συνήθως βασίζεται στο μέτρο της ακρίβειας. Υπάρχουν τουλάχιστον τρεις τεχνικές υπολογισμού της ακρίβειας ενός ταξινομητή.

Έστω ένα σύνολο δεδομένων Z μεγέθους $N \times n$, που περιέχει N αντικείμενα με χαρακτηριστικά n διαστάσεων. Από αυτό το σύνολο θα δημιουργήσουμε ένα σύνολο εκπαίδευσης (Training Set) και ένα σύνολο ελέγχου (Test Set). Αν θεωρήσουμε το σύνολο ελέγχου ως ένα υποσύνολο του συνόλου εκπαίδευσης, τότε υπάρχει κίνδυνος η ακρίβεια του ταξινομητή μας να είναι ελλιπής, ως προς δεδομένα που δεν ανήκουν στο σύνολο που θέλουμε να εξετάσουμε. Υποθέτοντας ένα σύστημα βασισμένο σε μηχανική μάθηση και ένα σύνολο δεδομένων εκπαίδευσης, το σύστημα καλείται να προσαρμόσει το μοντέλο του όσον το δυνατόν καλύτερα στα δεδομένα εκπαίδευσης. Μια ιδανική διαδικασία αποτίμησης θα περιλάμβανε ένα δεύτερο σύνολο δεδομένων, το οποίο θα είναι ανεξάρτητο από τα δεδομένα εκπαίδευσης, αλλά ακολουθεί την ίδια κατανομή με τα δεδομένα εκπαίδευσης. Οι τεχνικές που είναι ευρέως διαδεδομένες είναι οι ακόλουθες:

- **Cross-Validation:** Η διαδικασία Cross-Validation αποτελεί έναν τρόπο αποτίμησης που προβλέπει την επίδοση ενός συστήματος σε ένα σύνολο δεδομένων, στην περίπτωση που δεν είναι διαθέσιμα νέα δεδομένα. Η μέθοδος Cross-Validation θεωρείται ότι δίνει καλύτερη εκτίμηση της επίδοσης ενός συστήματος σε σχέση με άλλες τεχνικές. Αυτό την καθιστά έναν δημοφιλή τρόπο αποτίμησης συστημάτων μηχανικής μάθησης. Έστω ένα σύνολο δεδομένων Z με N στοιχεία, στη μέθοδο αυτή επιλέγουμε ένα ακέραιο K (συνήθως ένα διαιρέτη του N) και τυχαία δημιουργούμε K υποσύνολα του συνόλου Z πλήθους $\frac{N}{K}$ έστω $S = \{S_1, \cdots, S_K\}$ με $Z = \{S_1 \cup S_2 \cdots \cup S_K\}$. Στη συνέχεια, χρησιμοποιούμε κάποιο υποσύνολο S_i ως σύνολο ελέγχου και το εξετάζουμε στο σύνολο εκπαίδευσης δηλαδή στην ένωση των υπολοίπων $K-1$ υποσυνόλων. Αυτή η διαδικασία επαναλαμβάνεται K φορές, επιλέγοντας κάθε φορά ένα διαφορετικό S_i υποσύνολο για σύνολο ελέγχου (Test Set). Η ακρίβεια υπολογίζεται ως ο μέσος όρος της ακρίβειας των K επαναλήψεων.

- **Hold out:** Η τεχνική Hold out είναι ο χωρισμός του συνόλου δεδομένων στα δύο τρίτα για εκπαίδευση και το υπόλοιπο ένα τρίτο για σύνολο ελέγχου.

- **Bootstrap:** Η τεχνική αυτή χωρίζει το σύνολο δεδομένων σε τυχαία υποσύνολα μεγέθους N με αντικατάσταση και έπειτα υπολογίζει την ακρίβεια του αλγορίθμου ως τον μέσο όρο που παρουσιάζουν αυτά τα N υποσύνολα.

Η αξιολόγηση ενός ταξινομητή είναι γενικά μία σύνθετη διαδικασία για το λόγο ότι η αξιολόγηση εξαρτάται σε μεγάλο βαθμό από το σύνολο των δεδομένων μας. Ένας ταξινομητής σε διαφορετικό σύνολο δεδομένων ή με διαφορετικό αλγόριθμο μάθησης παρουσιάζει διαφορετική ακρίβεια. Ο Dietterich [11] διαπίστωσε τέσσερα σημαντικά σημεία στα οποία οφείλεται η παραπάνω διαφοροποίηση.

1. *Η επιλογή του συνόλου ελέγχου*
   Διαφορετικά σύνολα ελέγχου παρουσιάζουν διαφορετική ταξινομητική ακρίβεια.

2. *Η επιλογή του συνόλου εκπαίδευσης*
   Η επιλογή του συνόλου εκπαίδευσης είναι πολύ ουσιαστική για την διεξαγωγή των πειραμάτων διότι μικρές αλλαγές μπορεί να αποφέρουν μεγάλες διαφοροποιήσεις στην ακρίβεια του αλγορίθμου.

3. *Η τυχαιότητα του αλγορίθμου*
   Η τυχαιότητα που παρουσιάζουν τελεστές των αλγορίθμων. Οι τελεστές αυτοί σε ορισμένους αλγορίθμους έχουν σχέση με τις παραμέτρους του αλγορίθμου ή με τυχαίες διαδικασίες στην εκπαίδευση του ταξινομητή.

4. *Το τυχαίο σφάλμα ταξινόμησης*
   Η πιθανότητα να υπάρχουν δεδομένα στο σύνολο ελέγχου τα οποία είναι λάθος ταξινομημένα.

1.2.2 Παλινδρόμηση

Η διαδικασία της παλινδρόμησης είναι μία διαδικασία προσδιορισμού της σχέσης μίας μεταβλητής y (εξαρτημένη μεταβλητή ή έξοδος) με μία ή περισσότερες άλλες μεταβλητές $x_1, x_2, \cdots, x_n$ (ανεξάρτητες μεταβλητές ή είσοδοι). Σκοπός αυτής της διαδικασίας είναι η πρόβλεψη της τιμής της εξόδου όταν είναι γνωστές οι είσοδοι. Η Ανάλυση Παλινδρόμησης μας βοηθά να κατανοήσουμε τη μεταβολή της εξαρτώμενης μεταβλητής y όταν μεταβάλλεται μία από τις ανεξάρτητες μεταβλητές x, ενώ οι άλλες ανεξάρτητες μεταβλητές μένουν σταθερές. Συνήθως, επιδιώκεται να εξακριβωθεί η αιτιώδης επίδραση μιας μεταβλητής επάνω σε άλλη.

Μέθοδος Παλινδρόμησης
Αν Y είναι η εξαρτημένη μεταβλητή εξόδου και υπάρχουν k ανεξάρτητες μεταβλητές εισόδου, το μοντέλο που αποδίδει τη σχέση τους έχει την εξής μορφή

$$Y = g(x_1, x_2, \cdots, x_k) + \epsilon$$

με $(x_1, x_2, \cdots, x_k)$ οι τιμές των ανεξάρτητων μεταβλητών.
Y η εξαρτημένη μεταβλητή που έχει μέση τιμή $E(Y) = g(x_1, x_2, \cdots, x_k)$
ϵ είναι μία τυχαία μεταβλητή που καθορίζει το σφάλμα της πρόβλεψης της Y. Δηλαδή τη μεταβλητότητα της Y γύρω από τη μέση τιμή της λόγω της ύπαρξης μη ελεγχόμενων τυχαίων παραγόντων.
Στόχος της παλινδρόμησης είναι η ελαχιστοποίηση του τυχαίου σφάλματος πρόβλεψης. Δηλαδή η ελαχιστοποίηση της συνάρτησης

$$D = Y - g(x_1, x_2, \cdots, x_k)$$

Η ελαχιστοποίηση αυτή επιτυγχάνεται με τη μέθοδο της ελαχιστοποίησης του μέσου τετραγωνικού σφάλματος. Επειδή στην πράξη η διαδικασία αυτή είναι δύσκολη χρησιμοποιούμε συγκεκριμένες μορφές για τη συνάρτηση g όπως

$$g(x_1, x_2, \cdots, x_k) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

και προσδιορίζουμε τους άγνωστους παράγοντες $\beta_0, \beta_1, \cdots, \beta_k$ έτσι ώστε να ελαχιστοποιούν την συνάρτηση.

Γραμμική Παλινδρόμηση
Η πιο απλή περίπτωση παλινδρόμησης είναι η γραμμική παλινδρόμηση. Η γραμμική παλινδρόμηση είναι η περίπτωση που η εξαρτημένη μεταβλητή Y είναι γραμμικός συνδυασμός των εξαρτημένων μεταβλητών $X_1, X_2, \cdots, X_k$ δηλαδή

$$Y = a + b(X) + \epsilon$$

Λογιστική Παλινδρόμηση
Μία άλλη κατηγορία παλινδρόμησης για τη λύση προβλημάτων δυαδικής εξαρτημένης μεταβλητής είναι η λογιστική παλινδρόμηση. Σε αντίθεση με τη γραμμική παλινδρόμηση που προσαρμόζει μία ευθεία γραμμή στα δεδομένα, η λογιστική παλινδρόμηση προσαρμόζει μία συνάρτηση S στα δεδομένα. Έστω $Y = \{0, 1\}$ με $Y = 0$ αντιστοιχεί σε μία κατάσταση της μεταβλητής (αληθής) ενώ $Y = 1$ σε μια άλλη κατάσταση (ψευδής). Έστω ότι p η πιθανότητα να συμβεί $Y = 1$ δηλαδή $p = P(Y = 1)$. Τότε το μοντέλο της λογιστικής παλινδρόμησης επιχειρεί να λύσει την ακόλουθη εξίσωση:

$$g(x) = \ln(\frac{p}{1-p}) = \beta_0 + \beta_1 x + \epsilon$$

με

$$\frac{p}{1-p} = \mathrm{e}^{(\beta_0+\beta_1 x+\epsilon)}$$

Ο λόγος $\frac{p}{1-p}$ λέγεται λόγος των πιθανοτήτων της μεταβλητής Y, ενώ η κατανομή του νεπέριου λογαρίθμου αυτής της ποσότητας κυμαίνεται στο διάστημα $[0, 1]$.

1.3 Μάθηση χωρίς Επίβλεψη

Στη μάθηση χωρίς επίβλεψη το σύστημα έχει στόχο να ανακαλύψει συσχετίσεις και ομάδες από τα δεδομένα, βασιζόμενο μόνο στις ιδιότητές τους. Σαν αποτέλεσμα προκύπτουν πρότυπα (περιγραφές), κάθε ένα από τα οποία περιγράφουν ένα μέρος από τα δεδομένα. Στη μάθηση χωρίς επίβλεψη, δεν υπάρχει προκαθορισμένο σύνολο τιμών. Τα παραδείγματα εκπαίδευσης χωρίζονται σε, άγνωστες εκ των προτέρων, ομάδες με βάση τα χαρακτηριστικά τους. Διακρίνονται τα εξής είδη προβλημάτων:

1. Οι κανόνες συσχέτισης (association rules)
2. Ομάδες (clusters), οι οποίες προκύπτουν από τη διαδικασία της ομαδοποίησης (clustering)

1.3.1 Κανόνες Συσχέτισης

Η ανακάλυψη των κανόνων συσχέτισης εμφανίστηκε αρκετά αργότερα από τη μηχανική μάθηση και έχει περισσότερες επιρροές από την ερευνητική περιοχή των βάσεων δεδομένων. Προτάθηκε στις αρχές της δεκαετίας του '90 από τον Bayardo et al. [2] ως τεχνική ανάλυσης καλαθιού αγορών (market basket analysis), που το ζητούμενο είναι η δημιουργία συσχετίσεων ανάμεσα στα αντικείμενα μιας βάσης δεδομένων. Στο συγκεκριμένο πρόβλημα υπάρχει ένας μεγάλος αριθμός αντικειμένων (items), για παράδειγμα ψωμί, γάλα κλπ. Οι πελάτες γεμίζουν τα καλάθια τους με κάποιο υποσύνολο αυτών των αντικειμένων και το ζητούμενο είναι να βρεθεί ποια από αυτά τα αντικείμενα αγοράζονται μαζί, χωρίς να ενδιαφέρει ποιος είναι ο αγοραστής. Οι κανόνες συσχέτισης είναι προτάσεις της μορφής $\{X_1, \cdots, X_n\} \longrightarrow Y$ που σημαίνει ότι αν βρεθούν όλα τα $X_1, \cdots, X_n$ στο καλάθι τότε είναι πολύ πιθανό να βρεθεί και το Y.

1.3.2 Ομαδοποίηση

Ομαδοποίηση είναι η διαδικασία του διαχωρισμού ενός συνόλου δεδομένων σε ομάδες ώστε σημεία που ανήκουν στην ίδια ομάδα να μοιάζουν όσο το δυνατόν περισσότερο και σημεία που ανήκουν σε διαφορετικές ομάδες να διαφέρουν όσο το δυνατόν περισσότερο.

Η ομαδοποίηση είναι μία συλλογή μεθόδων για την εξαγωγή συσχετίσεων σε ένα σύνολο δεδομένων. Συχνά η ομαδοποίηση χρησιμοποιείται για την ανακάλυψη φυσικών ομαδοποιήσεων σε ένα σύνολο δεδομένων. Ένα ιδανικό δείγμα για ομαδοποίηση θα πρέπει να τηρεί τις κατάλληλες προδιαγραφές έτσι ώστε να δημιουργηθούν οι ιδανικές ομάδες. Μία ιδανική ομάδα θα πρέπει να αποτελείται από ένα σύνολο παρόμοιων προτύπων. Η απόσταση μεταξύ των προτύπων που ανήκουν σε μία ομάδα θα πρέπει να είναι μικρότερη από την απόσταση μεταξύ των προτύπων που ανήκουν σε διαφορετικές ομάδες. Οι ομάδες αποτελούν συνδεμένες περιοχές στο χώρο προτύπων με σχετικά μεγάλη πυκνότητα προτύπων και διαχωρίζονται από τις άλλες ομάδες με περιοχές με χαμηλή πυκνότητα προτύπων.

Αλγόριθμοι ομαδοποίησης χρησιμοποιούνται σε πολλές εφαρμογές όπως εξόρυξη δεδομένων Judd et al. [24], μηχανική μάθηση Carpineto and Romano [5], συμπίεση Mohamed and Fahmy [31], διανυσματική και χρωματική ανάλυση εικόνας Kaukoranta et al. [25]. Κύριο χαρακτηριστικό μιας ομάδας είναι το κέντρο της (Centroid).

Το πρόβλημα της ομαδοποίησης περιγράφεται αυστηρά ως εξής Veenman et al. [37]:
Έστω ένα σύνολο δεδομένων $Z = \{z_1, z_2, \cdots, z_p, \cdots, z_{N_p}\}$ όπου z_p είναι ένα πρότυπο στον N_d-διάστατο χώρο χαρακτηριστικών, και N_p είναι ο αριθμός των προτύπων του Z. Η ομαδοποίηση του Z είναι ο διαχωρισμός του Z σε K ομάδες (clusters) $\{C_1, C_2, \cdots, C_K\}$ που ικανοποιούν τις ακόλουθες συνθήκες:

- Κάθε πρότυπο θα πρέπει να ανήκει σε μία ομάδα

$$\cup_{k=1}^{K} C_k = Z$$

- Κάθε ομάδα θα πρέπει να έχει τουλάχιστον ένα πρότυπο

$$C_k \neq \oslash, \;\; k = 1, \cdots, K$$

- Κάθε πρότυπο ανήκει σε μία και μόνο μία ομάδα

$$C_k \cap C_{kk} = \oslash, \;\; k \neq kk$$

1.4 Βιολογικά Εμπνευσμένοι Υπολογισμοί

Οι βιολογικά εμπνευσμένοι υπολογισμοί (Bio-inspired Computing) είναι ένας τρόπος ανάπτυξης υπολογιστικών συστημάτων και αλγορίθμων εμπνευσμένοι από τον βιολογικό κόσμο. Οι περισσότερες προσεγγίσεις βασίζονται σε βασικές και απλοποιημένες τεχνικές και διαδικασίες της βιολογίας.

Το 1978 οι Paulien Hogeweg και Ben Hesper μελετώντας τις διαδικασίες πληροφορικής σε βιοτικά συστήματα εισήγαγαν τους όρους Βιοπληροφορική και Υπολογιστική Βιολογία. Ο ορισμός της βιοπληροφορικής όπως παρουσιάζεται στο [19] είναι ο εξής.

Ορισμός 2 *Βιοπληροφορική είναι ο επιστημονικός χώρος όπου η σύμπραξη της Βιολογίας με την Πληροφορική, την Στατιστική και τα Μαθηματικά εξερευνά νέους τρόπους για την προσέγγιση των βιολογικών προβλημάτων, καθώς και την αντίληψη βασικών αρχών της Βιολογίας. Πρόκειται για γνωστικό χώρο με συγκεκριμένο όσο και ευρύ πεδίο εφαρμογών και αλληλεπίδρασης με τη σύγχρονη δομική, μοριακή, πληθυσμιακή και περιβαλλοντική βιολογία.*

Παρόλο που ο τομέας αυτός βασίζεται κυρίως στην έρευνα των βιολογικών φαινομένων και την ενσωμάτωση τους σε υπολογιστικά συστήματα, συγχρόνως δημιουργεί βάσεις ανάπτυξης νέων υπολογιστικών εργαλείων για την λύση δύσκολων προβλημάτων. Είναι γεγονός ότι πολλές βιολογικά εμπνευσμένες τεχνικές παρουσιάζουν λύσεις σε προβλήματα που δεν μπορούν να λυθούν με τις κλασικές μεθόδους όπως με τον γραμμικό , μη γραμμικό και δυναμικό προγραμματισμό. Οι πιο γνωστές προσεγγίσεις που έχουν αναπτυχθεί από τους βιολογικά εμπνευσμένους υπολογισμούς είναι τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks), οι Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms), η Νοημοσύνη του Σμήνους (Swarm Intelligence) και τα Τεχνητά Ανοσοποιητικά Συστήματα.

1.4.1 Τεχνητά Νευρωνικά Δίκτυα

Τα Τεχνητά Νευρωνικά Δίκτυα αναπτύχθηκαν από McCulloch and Pitts [29] οι οποίοι διατύπωσαν το πρώτο μαθηματικό μοντέλο για ένα νευρώνα και έτσι δημιουργήθηκε ο κλάδος των Τεχνητών Νευρωνικών Δικτύων [3, 13, 26] . Τα Τεχνητά Νευρωνικά Δίκτυα μπορούν να χαρακτηριστούν ως συστήματα επεξεργασίας πληροφοριών εμπνευσμένα από το νευρικό σύστημα του ανθρώπινου εγκεφάλου. Κατά κύριο λόγο, δίνουν έμφαση στον τομέα της επίλυσης προβλημάτων. Οι νευρώνες είναι οι βασικές μονάδες που χρησιμοποιεί ο εγκέφαλος για την επεξεργασία της πληροφορίας. Τα Τεχνητά Νευρωνικά Δίκτυα μοντελοποιούν μία απλοποιημένη μορφή των νευρώνων και αποτελούν την βασική μονάδα επεξεργασίας του συστήματος. Οι Νευρώνες είναι συνδεδεμένοι μεταξύ τους με μικρές ενώσεις που καλούνται συνάψεις και η δυνατότητά τους να διαμορφώνονται αποτελεί τη ικανότητα του ανθρώπου για γνωστικές ικανότητες όπως αντίληψη, σκέψη, συμπερασμός. Οι συνάψεις αυτές συνδέουν τους Νευρώνες μεταξύ τους και έχει ως αποτέλεσμα τη δημιουργία Δίκτυα Νευρώνων που επικοινωνούν μεταξύ τους. Ένα ακόμη βασικό χαρακτηριστικό των Νευρωνικών Δικτύων είναι η αναπαράσταση της γνώσης με ένα κατανεμημένο τρόπο καθώς και η παράλληλη επεξεργασία αυτής.

Τα κύρια χαρακτηριστικά των Τεχνητών Νευρωνικών Δικτύων είναι τρία:

1. Ένα σύνολο τεχνητών νευρώνων
2. Ένα δίκτυο επικοινωνίας μεταξύ των νευρώνων που είναι η δομή του Τ.Ν.Δ.
3. Μία διαδικασία μάθησης που είναι μία μέθοδος για τον υπολογισμό των βαρών των νευρώνων.

Οι νευρώνες είναι τα δομικά στοιχεία του δικτύου. Κάθε νευρώνας δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε από το περιβάλλον), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου.

Έστω ένας νευρώνας που αποτελείται από d συνδέσεις εισόδου, κάθε μία από τις οποίες δέχεται ένα σήμα εισόδου x_i με $(i = 1, 2, 3, \cdots, d)$ και χαρακτηρίζεται από μια τιμή βάρους w_i $(i = 1, 2, 3, \cdots d)$. Η είσοδος x_0 είναι μια σύνδεση σταθερής διέγερσης με τιμή που ισούται μονίμως με 1 και βάρος w_0 που ονομάζεται πόλωση (bias) του νευρώνα. Η εισαγωγή της πόλωσης προσδίδει στο νευρώνα επιπλέον υπολογιστικές δυνατότητες μάθησης και προσαρμογής στα δεδομένα εκπαίδευσης. Ο υπολογισμός που επιτελεί ένας νευρώνας διακρίνεται σε δύο στάδια :

- Υπολογισμός της συνολικής εισόδου $u(x) = \sum_{i=1}^{d} w_i x_i + w_0$
- Υπολογισμός της εξόδου $o(x)$ του νευρώνα περνώντας τη συνολική είσοδο $u(x)$ από μία συνάρτηση ενεργοποίησης (activation function) $g : p = g(u)$

Διαδικασία Μάθησης

Το κύριο χαρακτηριστικό των νευρωνικών δικτύων είναι η ικανότητα μάθησης. Με το όρο μάθηση εννοούμε τη σταδιακή βελτίωση της ικανότητας του δικτύου να επιλύει κάποιο πρόβλημα. Η μάθηση επιτυγχάνεται μέσω της εκπαίδευσης, μίας επαναληπτικής διαδικασίας σταδιακής προσαρμογής των παραμέτρων του δικτύου (συνήθως των βαρών και της πόλωσης του) σε τιμές κατάλληλες ώστε να επιλύεται με επαρκή επιτυχία το προς εξέταση πρόβλημα. Η εκπαίδευση αυτή υλοποιείται μέσω της ανταλλαγής τιμών και βαρών που αποσκοπεί στη βαθμιαία σύλληψη της πληροφορίας η οποία στη συνέχεια θα είναι διαθέσιμη προς ανάκτηση. Η μάθηση αυτή κατηγοριοποιείται σε δύο κατηγορίες μάθηση με επίβλεψη και μάθηση χωρίς επίβλεψη.

Μάθηση με επίβλεψη: Η μάθηση αυτή είναι μια διαδικασία η οποία συνδυάζει έναν εξωτερικό εκπαιδευτή ο οποίος έχει την γνώση του περιβάλλοντος πάνω στο οποίο λειτουργεί το δίκτυο. Αυτή η γνώση διοχετεύεται στο δίκτυο στη μορφή ενός συνόλου σημάτων εισόδου-εξόδου. Οι ελεύθερες μεταβλητές του δικτύου προσαρμόζονται ανάλογα με την είσοδο και το σφάλμα του σήματος. Το σφάλμα είναι η διαφορά της επιθυμητής εξόδου d_j από την πραγματική έξοδο y_j.

Μάθηση χωρίς επίβλεψη: Οι αλγόριθμοι της μάθησης χωρίς επίβλεψη είναι διαδικασίες οι οποίες δεν απαιτούν να είναι παρών ένας εξωτερικός δάσκαλος ή επιβλέπων. Βασίζονται, μάλιστα, μόνο σε τοπική πληροφορία καθ' όλη τη διάρκεια της εκπαίδευσης του Τεχνητού Νευρωνικού Δικτύου. Οι συγκεκριμένοι αλγόριθμοι οργανώνουν τα δεδομένα και ανακαλύπτουν σημαντικές συλλογικές ιδιότητες.

1.4.2 Εξελικτικοί Αλγόριθμοι

Οι Εξελικτικοί Αλγόριθμοι (ΕΑ) αποτελούν ένα τομέα της Εξελικτικής Υπολογιστικής. Χρησιμοποιούν υπολογιστικά μοντέλα εξελικτικών διαδικασιών, σαν βασικά στοιχεία σχεδιασμού και υλοποίησης υπολογιστικών συστημάτων επίλυσης προβλημάτων. Οι ΕΑ είναι αλγόριθμοι ανίχνευσης-αναζήτησης, βασισμένοι στη μηχανική της φυσικής επιλογής και της φυσικής γενετικής. Συνδυάζουν την επιβίωση του ικανότερου με μία οργανωμένη ανταλλαγή πληροφοριών, με στόχο την διαμόρφωση ενός αλγόριθμου ανίχνευσης που να διαθέτει τη διαίσθηση της ανθρώπινης ανίχνευσης. Οι ΕΑ μιμούνται τις διαδικασίες βιολογικής εξέλιξης με την υλοποίηση των ιδεών της φυσικής επιλογής και της επικράτησης του ισχυρότερου, έτσι ώστε να παρέχουν αποτελεσματικές λύσεις σε προβλήματα αναζήτησης και βελτιστοποίησης.

Οι Εξελικτικοί αλγόριθμοι είναι στοχαστικοί αλγόριθμοι. Διαθέτουν ένα σύνολο από υποψήφιες λύσεις του προβλήματος, που ονομάζεται πληθυσμός. Πραγματοποιούν εξέλιξη του πληθυσμού αυτού χρησιμοποιώντας μηχανισμούς που χρησιμοποιούνται από τη φύση για την εξέλιξη των οργανισμών, όπως η διασταύρωση , η μετάλλαξη και η επιλογή.

Η διαδικασία που ακολουθείται για την ανάπτυξη ενός Εξελικτικού Αλγορίθμου είναι η ακόλουθη. Στην αρχή, ο πληθυσμός αρχικοποιείται συνήθως με τυχαίο τρόπο και αξιολογείται από την συνάρτηση καταλληλότητας (αντικειμενική συνάρτηση) που έχει σχηματιστεί για το υπό εξέταση πρόβλημα. Στη συνέχεια σε ένα επαναληπτικό κύκλο εφαρμόζονται οι μηχανισμοί της διασταύρωσης και της μετάλλαξης που μεταβάλλουν τον πληθυσμό και παράγουν νέα άτομα. Τα άτομα που ανήκουν στον πληθυσμό στην τρέχουσα επανάληψη αυτού του σχήματος αποτελούν τη γενιά του πληθυσμού. Η διαδικασία της επιλογής σε κάθε επανάληψη αποφασίζει ποια άτομα θα συνεχίσουν στην επόμενη γενιά. Ο επαναληπτικός κύκλος θα σταματήσει όταν ικανοποιηθεί το κριτήριο τερματισμού. Αυτό μπορεί να είναι ένας προκαθορισμένος αριθμός επαναλήψεων ή η επίτευξη μιας τιμής-στόχου της αντικειμενικής συνάρτησης.

Οι Εξελικτικοί Αλγόριθμοι απαιτούν μόνο τις τιμές της αντικειμενικής συνάρτησης για να εφαρμοστούν και δεν απαιτούν κάποια πληροφορία για τις παραγώγους της. Το γεγονός αυτό μαζί με την απλότητα και

την ευκολία που τους χαρακτηρίζει στην υλοποίηση τους στον υπολογιστή, τους έχει κάνει πολύ δημοφιλείς για πολύπλοκα προβλήματα βελτιστοποίησης. Όταν η υπό εξέταση συνάρτηση είναι ομαλή και οι κλασικές μέθοδοι που έχουν σχεδιαστεί για τέτοιες συναρτήσεις είναι αποτελεσματικές τότε η χρήση των Εξελικτικών Αλγορίθμων δεν θα προσδώσει κάτι περισσότερο. Αντίθετα όμως στις περιπτώσεις όπου οι γνωστές κλασικές μέθοδοι αποτυγχάνουν ή δεν μπορούν να εφαρμοστούν έχει αποδειχτεί στην πράξη ότι οι Εξελικτικοί Αλγόριθμοι μπορούν να επιτύχουν. Τέτοιες περιπτώσεις είναι όταν η αντικειμενική συνάρτηση είναι μη παραγωγίσιμη, υπάρχουν ασυνέχειες ή υπάρχουν πάρα πολλά τοπικά ακρότατα. Επίσης ένα σημαντικό χαρακτηριστικό των Εξελικτικών Αλγορίθμων είναι ότι λόγω της δομής τους μπορούν πολύ εύκολα να παραλληλοποιηθούν. Έτσι, μπορεί να γίνει αποτελεσματική αξιοποίηση των υπαρχόντων παράλληλων υπολογιστικών συστημάτων και να επιλυθούν αποδοτικά, πολύπλοκα και απαιτητικά προβλήματα βελτιστοποίησης.

Οι Εξελικτικοί Αλγόριθμοι μοιράζονται ορισμένα κοινά χαρακτηριστικά. Η γενική μορφή των εξελικτικών αλγορίθμων είναι η εξής:

1. *Ένα πληθυσμό ατόμων (individual) που αναπαράγονται*
   Οι Εξελικτικοί Αλγόριθμοι χρησιμοποιούν την συλλογική μάθηση ενός πληθυσμού. Συνήθως, κάθε άτομο του πληθυσμού αυτού παριστάνει ένα σημείο στο χώρο λύσεων ενός προβλήματος. Τα άτομα αυτά αναπαράγονται δημιουργώντας απογόνους που κληρονομούν στοιχεία από τους γονείς τους.

2. *Γενετική Διαφοροποίηση*
   Οι απόγονοι των ατόμων διαφοροποιούνται γενετικά από τους γονείς τους, μέσω της διαδικασίας της μετάλλαξης και της διασταύρωσης.

3. *Φυσική Επιλογή*
   Χρησιμοποιείται η αντικειμενική συνάρτηση για την αξιολόγηση κάθε ατόμου του πληθυσμού. Βάση αυτής της συνάρτησης υλοποιείται η επιλογή καλύτερων ατόμων ενός πληθυσμού.

Μερικές διαφοροποιήσεις των τριών κανόνων αυτών έχουν οδηγήσει στη δημιουργία τριών κατηγοριών των εξελικτικών αλγορίθμων, τους γενετικούς αλγορίθμους, εξελικτικές στρατηγικές και στους εξελικτικό προγραμματισμό.

1. Οι γενετικοί αλγόριθμοι (όπως έχει περιγραφεί από τον Holland [20]) δίνουν έμφαση περισσότερο στην διαδικασία της διασταύρωσης, ως τον πιο σημαντικό παράγοντα και λιγότερο στον παράγοντα της μετάλλαξης. Ενώ χρησιμοποιούν ένα πιθανοθεωρητικό παράγοντα επιλογής.

2. Οι εξελικτικές στρατηγικές (που έχουν αναπτυχθεί από τον Rechenberg [34]και τον Schwefel [35] από το Technical University of Berlin) χρησιμοποιούν μεταλλάξεις από την κανονική κατανομή για να μεταλλάξουν διανύσματα πραγματικών αριθμών και δίνουν έμφαση στη μετάλλαξη και τη διασταύρωση ως διαδικασίες αναζήτησης στο χώρο αναζήτησης. Ο παράγοντας επιλογής είναι ντετερμινιστικός καθώς και οι γονείς και οι απόγονοι ενός πληθυσμού διαφέρουν ως προς το πλήθος.

3. Εξελικτικός Προγραμματισμός (αναπτύχθηκε από τον Fogel [14] από το University of California in San Diego) δίνει έμφαση στην μετάλλαξη του πληθυσμού και δεν χρησιμοποιεί καθόλου τη διαδικασία της διασταύρωσης. Τέλος ο παράγοντας της επιλογής δίδεται από μία συνάρτησης πιθανότητας.

1.4.3 Νοημοσύνη του Σμήνους

Τα τελευταία χρόνια η Νοημοσύνη του Σμήνος (Swarm Intelligence), που θεωρείται τομέας της Τεχνητής Νοημοσύνης, έχει προσελκύσει το ενδιαφέρον πολλών ερευνητών και έχει εφαρμοστεί σε διάφορα υπολογιστικά προβλήματα. Η Νοημοσύνη του Σμήνους είναι "η συλλογική συμπεριφορά αυταρκών μεν, αποκεντρωμένων δε, τεχνητών συστημάτων". Με άλλα λόγια, είναι η συμπεριφορά ενός συνόλου ατόμων κατά την επαφή τους με το περιβάλλον τους αλλά και η αλληλεπίδρασή τους. Υπάρχει ένα πλήθος μοντέλων που έχουν προταθεί και ερευνηθεί στον τομέα του Swarm Intelligence, τα πιο γνωστά είναι αυτά της αποικίας των μυρμηγκιών, αποικίας των μελισσών και η τεχνική σμήνους σωματιδίων. Γενικά, η Νοημοσύνη του Σμήνους ασχολείται με την μοντελοποίηση συλλογικών συμπεριφορών απλών πρακτόρων που αλληλεπιδρούν μεταξύ τους και με το περιβάλλον τους. Αυτά τα μοντέλα εμπνέονται από έντομα ή άλλα ζώα. Από υπολογιστική άποψη, η Νοημοσύνη του Σμήνους είναι υπολογιστικοί αλγόριθμοι που χρησιμοποιούνται για την επίλυση κατανεμημένων προβλημάτων. Ένα σμήνος μπορεί να θεωρηθεί σαν μια ομάδα πρακτόρων οι οποίοι συνεργάζονται για την επίτευξη κάποιου στόχου. Οι πράκτορες αυτοί ακολουθούν απλούς κανόνες

και δεν υπάρχουν γενικοί κανόνες ως προς το πώς θα αλληλεπιδράσουν με τους άλλους ή με το περιβάλλον τους, με αυτό τον τρόπο επιτυγχάνεται μία συλλογική νοημοσύνη.

Η συλλογική νοημοσύνη που επιτυγχάνεται, οφείλεται κατά κύριο λόγο στην αυτο-οργάνωση των εντόμων. Η αυτο-οργάνωση είναι ένα σύνολο δυναμικών μηχανισμών που δημιουργούνται από αλληλεπιδράσεις των συνιστωσών του. Οι αλληλεπιδράσεις αυτές οφείλονται σε τοπικές πληροφορίες χωρίς να λαμβάνουν υπόψη τους τη συνολική εικόνα. Τα βασικά στοιχεία που χαρακτηρίζουν την αυτο-οργάνωση είναι η θετική ανάδραση, η αρνητική ανάδραση, η ενίσχυση των τυχαίων διακυμάνσεων και η αλληλεπίδραση μεταξύ των συνιστωσών του συστήματος.

- **Η θετική ανάδραση** είναι ένα σύνολο κανόνων το οποίο είναι υπεύθυνο για τη δημιουργία των βασικών δομών. Κατά τη συγκεκριμένη διεργασία ένα έντομο αντιδρά σε ένα ερέθισμα φροντίζοντας για την προσέλκυση και άλλων εντόμων σε αυτό το ερέθισμα. Για παράδειγμα όταν ένα μυρμήγκι εντοπίζει ίχνος φερομόνης τότε το ακολουθεί και φροντίζει να το ενισχύσει με επιπλέον φερομόνη.

- **Η Αρνητική Ανάδραση** δρα ανταγωνιστικά στη θετική αποτρέποντας πολύ ριζοσπαστικές αλλαγές και σταθεροποιώντας τις υπάρχουσες δομές. Επιπλέον μειώνει τον κίνδυνο εγκλωβισμού αφού ενισχύει την εξερεύνηση της διαδικασίας.

- **Ενίσχυση των Τυχαίων Διακυμάνσεων** είναι ένας σημαντικός μηχανισμός αναζήτησης σε ένα σύστημα με αυτό-οργάνωση αφού γεννά καινοτομίες. Ουσιαστικά βασίζεται στις ατέλειες του κάθε συστήματος οι οποίες ενισχύονται λόγω της θετικής ανάδρασης.

- **Αλληλεπίδραση μεταξύ των συνιστωσών του συστήματος** επιτυγχάνεται όταν κάθε έντομο είναι σε θέση να δημιουργήσει κάποια σταθερή δομή και να ανιχνεύσει τις δομές που δημιούργησαν άλλα έντομα της κοινότητάς του. Για να έχει αποτέλεσμα η αλληλεπίδραση θα πρέπει να υπάρχει ένας ελάχιστος πληθυσμός εντόμων, διαφορετικά η εμφάνιση οργανωμένων δομών μέσω της συλλογικής δράσης είναι αδύνατη διότι υπάρχουν φαινόμενα αρνητικής ανάδρασης.

Τέλος, όλα τα μοντέλα Νοημοσύνης Σμήνους ακολουθούν ένα αριθμό από γενικούς κανόνες. Ο πρώτος κανόνας είναι ότι κάθε οντότητα του μοντέλου είναι ένας απλός πράκτορας. Η επικοινωνία ανάμεσα στους πράκτορες είναι έμμεση. Η συνεργασία των πρακτόρων γίνεται με κατανεμημένο τρόπο και χωρίς την παρουσία κάποιου κεντρικού ελέγχου.

1.4.4 Τεχνητά Ανοσοποιητικά Συστήματα

Τα Τεχνητά Ανοσοποιητικά Συστήματα είναι ένα ξεχωριστό πεδίο έρευνας που αναφέρεται σε υπολογιστικά συστήματα εμπνευσμένα από την ανοσολογία με κύριο στόχο την επίλυση προβλημάτων. Είναι ένα νέο πεδίο έρευνας που αναπτύχθηκε στα μέσα της δεκαετίας του 1980 και οι εφαρμογές του κινούνται σε ένα ευρύ φάσμα από τη βιολογία ως τη ρομποτική.

Το Ανοσοποιητικό Σύστημα των σπονδυλωτών οργανισμών εκτελεί ιδιαίτερα πολύπλοκους υπολογισμούς με έναν εξαιρετικά παράλληλο και κατανεμημένο τρόπο. Χρησιμοποιεί διάφορους μηχανισμούς μάθησης και μνήμης για την επίλυση προβλημάτων αναγνώρισης και ταξινόμησης. Σκοπός του ανοσοποιητικού μας συστήματος είναι η προστασία του οργανισμού μας από εξωγενής παθογόνους οργανισμούς όπως ιοί και βακτήρια. Κύριος μηχανισμός είναι η αναγνώριση των κυττάρων του οργανισμού μας και η διάκριση τους σε κύτταρα εαυτού και μη εαυτού. Τα κύρια επίπεδα του ανοσοποιητικού συστήματος είναι το επίπεδο της μη ειδικής ανοσίας και το ειδικής ανοσίας. Τα κύτταρα του συστήματος μη ειδικής ανοσίας αναγνωρίζουν και επιτίθενται στους παθογόνους οργανισμούς. Από την άλλη πλευρά, το σύστημα ειδικής ανοσίας είναι υπεύθυνο για την παραγωγή αντισωμάτων για την αντιμετώπιση παθογόνων οργανισμών που έχουν ήδη αναγνωριστεί από το σύστημα μη ειδικής ανοσίας.

Σύμφωνα με τα προηγούμενα, είναι φανερό ότι το ανοσοποιητικό μας σύστημα εκτελεί διάφορες διεργασίες όπως η αναγνώριση εαυτού από μη εαυτό, η εξαγωγή πληροφορίας, η μνήμη, η μάθηση οι οποίες από υπολογιστική άποψης έχουν μοντελοποιηθεί και χρησιμοποιηθεί σε πολλές εφαρμογές. Μερικές από αυτές είναι εφαρμογές ανίχνευσης ιών, εφαρμογές αναγνώρισης προτύπων, μηχανικής μάθησης, βιοπληροφορικής, βελτιστοποίησης και άλλες. Τα τεχνητά Ανοσοποιητικά Συστήματα είναι ένα από τα κύρια θέματα της παρούσας εργασίας και θα αναλυθούν λεπτομερώς στη συνέχεια.

Κεφάλαιο 2

Γενετικοί Αλγόριθμοι

2.1 Γενικά περί Γενετικών αλγορίθμων

Οι Γενετικοί Αλγόριθμοι στηρίζονται στην Δαρβινική Θεωρία και πιο συγκεκριμένα στην αρχή της εξέλιξης των ειδών. Ένα βασικό συστατικό στη Δαρβινική θεωρία είναι ότι ανάμεσα σε οργανισμούς που ανήκουν στο ίδιο είδος, δεν υπάρχει κάποιος διαχωρισμός σε κατώτερους ή ανώτερους. Οι απόγονοι ενός πληθυσμού καθορίζονται από το περιβάλλον, έτσι η βελτίωση των συνθηκών διαβίωσης αυξάνει την πιθανότητα οι απόγονοι να κληρονομήσουν κάποια χαρακτηριστικά των γονιών τους.

Ένα ακόμη βασικό στοιχείο της θεωρίας είναι η δομή των ατόμων ενός είδους. Μία βασική δομή είναι τα χρωμοσώματα τα οποία κωδικοποιούν τα χαρακτηριστικά των ατόμων ενός είδους. Τα χρωμοσώματα αποτελούνται από κάποιες μικρότερες δομικές μονάδες τα γονίδια (gene). Το σύνολο της πληροφορίας που αποθηκεύεται στα γονίδια ονομάζεται γενότυπος (genotype). Τα χαρακτηριστικά ενός οργανισμού δημιουργούνται με αποκωδικοποίηση του γενότυπου. Το σύνολο των χαρακτηριστικών που βλέπουμε με αυτή την αποκωδικοποίηση είναι ο φαινότυπος. Οι βασικές λειτουργίες των οργανισμών είναι η αναπαραγωγή και η μετάλλαξη. Στη διαδικασία της αναπαραγωγής δύο μέλη του οργανισμού ανταλλάσσουν γενετικό υλικό με στόχο την αναπαραγωγή. Η μετάλλαξη στα είδη λαμβάνει χώρα σε πολύ αραιά χρονικά διαστήματα που προκαλείται είτε από γενετικούς παράγοντες είτε από παράγοντες του περιβάλλοντος.

Οι Γενετικοί Αλγόριθμοι δημιουργήθηκαν ως μια διαδικασία ευρεστικής αναζήτησης που μιμείται την διαδικασία της φυσικής εξέλιξης όπως αυτή έχει διατυπωθεί από την Δαρβινική θεωρία. Οι Γενετικοί Αλγόριθμοι, ανήκουν στον τομέα των εξελικτικών αλγορίθμων και αποτελούν μία κατηγορία συστημάτων επίλυσης προβλημάτων βελτιστοποίησης χρησιμοποιώντας τεχνικές εμπνευσμένες από την φυσική εξέλιξη όπως την κληρονομικότητα (inheritance) , την μετάλλαξη (mutation), την επιλογή (selection), την διασταύρωση (crossover).

Η πρώτη εμφάνιση των εξελικτικών συστημάτων, χρονολογείται στην περίοδο 1950 και 1960, όταν διάφοροι επιστήμονες προσπάθησαν να αναπτύξουν εξελικτικά συστήματα με την ιδέα ότι η εξέλιξη μπορεί να χρησιμοποιηθεί σαν εργαλείο βελτιστοποίησης. Η βασική ιδέα ήταν η δημιουργία ενός πληθυσμού από υποψήφιες λύσεις για ένα πρόβλημα, χρησιμοποιώντας την φυσική επιλογή και την φυσική γενετική μετάλλαξη.

Το 1960 ο Rechenberg [33] εισήγαγε τις “Εξελικτικές στρατηγικές” (“evolution strategies”) μία μέθοδο για βελτιστοποίηση παραμέτρων. Στη συνέχεια το 1966 ο Fogel, Owens και Walsh [12, 15] δημιούργησαν τον “Εξελικτικό προγραμματισμό” (“evolutionary programming”) μία μέθοδο με την οποία οι υποψήφιες λύσεις ενός δεδομένου προβλήματος αναπαρίστανται ως πεπερασμένα αυτόματα τα οποία μετάλλασσαν τυχαία τα διαγράμματα μετάβασης καταστάσεων επιλέγοντας το καλύτερο.

Οι Γενετικοί Αλγόριθμοι εφευρέθηκαν από τον John Holland το 1960 και αναπτύχθηκαν από αυτόν και τους φοιτητές του στο Πανεπιστήμιο του Michigan το 1960 και 1970. Ο κύριος στόχος του Holland ήταν να μελετήσει το φαινόμενο της εξελικτικής προσαρμογής όπως αυτή παρουσιάζεται στη φύση και να δημιουργήσει μηχανισμούς φυσικής προσαρμογής που να μπορούν να εισαχθούν στα υπολογιστικά συστήματα. Ο Γενετικός Αλγόριθμος του Holland ήταν μία μέθοδος δημιουργίας πληθυσμών χρωμοσωμάτων (συμβολοσειρές από 0 και 1, δηλαδή bits) μέσα από τη διαδικασία της φυσικής επιλογής, της διασταύρωσης και της μετάλλαξης. Κάθε χρωμόσωμα αποτελείται από γονίδια (bits) και κάθε γονίδιο είναι μία παράσταση από αλληλόμορφα (0 και 1). Η διαδικασία της επιλογής διαλέγει τα καλύτερα χρωμοσώματα του πληθυσμού για να παράγει απογόνους. Η διαδικασία της διασταύρωσης ανταλλάσσει υποσύνολα των δύο χρωμοσωμάτων

ενώ η διαδικασία της μετάλλαξης αλλάζει τυχαία τα αλληλόμορφα των χρωμοσωμάτων σε κάποιες τυχαίες θέσεις. Η εισαγωγή ενός πληθυσμού που υπόκειται σε διαδικασίες επιλογής διασταύρωσης και μετάλλαξης ήταν μία πρωτοποριακή μέθοδος και ήταν μία πρώτη προσπάθεια εισαγωγής της έννοιας του εξελικτικού υπολογισμού σε ένα αυστηρό θεωρητικό υπόβαθρο.

2.1.1 Διαδικασία Γενετικών Αλγορίθμων

Οι Γενετικοί Αλγόριθμοι διατηρούν έναν πληθυσμό πιθανών λύσεων, του προβλήματος που μας ενδιαφέρει, πάνω στον οποίο δουλεύουν. Έτσι ένας Γενετικός Αλγόριθμος πραγματοποιεί αναζήτηση σε πολλές κατευθύνσεις και υποστηρίζει καταγραφή και ανταλλαγή πληροφοριών μεταξύ αυτών των κατευθύνσεων. Ο πληθυσμός υφίσταται μια προσομοιωμένη γενετική εξέλιξη. Σε κάθε γενιά, οι σχετικά "καλές" λύσεις αναπαράγονται, ενώ οι σχετικά "κακές" απομακρύνονται. Ο διαχωρισμός και η αποτίμηση των διαφόρων λύσεων γίνεται με την βοήθεια μιας αντικειμενικής συνάρτησης, η οποία παίζει το ρόλο του περιβάλλοντος μέσα στο οποίο εξελίσσεται ο πληθυσμός. Πιο συγκεκριμένα μπορούμε να πούμε ότι ένας Γενετικός Αλγόριθμος για ένα συγκεκριμένο πρόβλημα πρέπει να αποτελείται από τα παρακάτω πέντε συστατικά:

- *Μια γενετική αναπαράσταση των πιθανών λύσεων του προβλήματος*

  Για την αναζήτηση πιθανών λύσεων ενός προβλήματος από ένα γενετικό αλγόριθμο, είναι αναγκαία η κωδικοποίηση των πιθανών λύσεων σε μία ακολουθία από χαρακτήρες που ανήκουν σε ένα αλφάβητο $A = a_1, a_2, \cdots a_L$. Οι χαρακτήρες αυτοί συχνά είναι δυαδικοί αλλά και πραγματικοί αριθμοί. Η ακολουθία χαρακτήρων (string) $a_1, \cdots, a_i$ ονομάζεται χρωμόσωμα ενώ οι χαρακτήρες a_i ονομάζονται γονίδια. Το σύνολο των τιμών που ένα γονίδιο μπορεί να λάβει ονομάζεται αλληλόμορφα. Στην πιο απλή περίπτωση όταν οι χαρακτήρες είναι δυαδικοί, τα δύο πιθανά αλληλόμορφα είναι το 0 και το 1. Έτσι στο χρωμόσωμα A=11110000 τα τέσσερα πρώτα γονίδια έχουν αλληλόμορφο 1 και τα τέσσερα τελευταία έχουν αλληλόμορφο 0.

- *Μια αντικειμενική συνάρτηση αξιολόγησης των μελών του πληθυσμού*

  Στόχος ενός γενετικού αλγορίθμου είναι να εντοπίσει την καλύτερη λύση σε ένα πρόβλημα βελτιστοποίησης δηλαδή να εντοπίσει την καλύτερη ακολουθία χαρακτήρων (χρωμόσωμα). Το πόσο καλή είναι η ακολουθία χαρακτήρων (χρωμόσωμα) το υποδεικνύει η αντικειμενική συνάρτηση.

- *Έναν τρόπο δημιουργίας ενός αρχικού πληθυσμού από πιθανές λύσεις*

  Ένας γενετικός αλγόριθμος δημιουργεί ένα αρχικό πληθυσμό χρωμοσωμάτων και στη συνέχεια τον εξελίσσει σε έναν με καλύτερα χαρακτηριστικά. Για την εξέλιξη αυτή λαμβάνουν μέρος ορισμένες διαδικασίες, οι οποίες λαμβάνουν τον πληθυσμό σε μία συγκεκριμένη γενιά και από αυτόν παράγουν τον επόμενο με τέτοιο τρόπο έτσι ώστε ο μέσος βαθμός όλου του πληθυσμού να είναι καλύτερος από αυτόν του προηγούμενου. Αυτές οι διαδικασίες επαναλαμβάνονται μέχρι ένα κριτήριο τερματισμού να εκπληρωθεί. Οι διαδικασίες αυτές είναι η επιλογή, η διασταύρωση, η μετάλλαξη.

- *Γενετικούς τελεστές για τη δημιουργία νέων μελών*

  Οι γενετικοί τελεστές είναι οι εξής:

  1. **Επιλογή**: Αυτός ο τελεστής επιλέγει χρωμοσώματα από έναν πληθυσμό με τέτοιον τρόπο έτσι ώστε να επιλεχθούν τα καλύτερα χρωμοσώματα. Η διαδικασία αυτή δεν προσθέτει καινούργια χρωμοσώματα στον πληθυσμό αλλά τον ανακατασκευάζει έτσι ώστε οι επόμενοι πληθυσμοί να περιέχουν όλο και καλύτερα χρωμοσώματα.
  2. **Διασταύρωση**: Αυτός ο τελεστής παράγει δύο απογόνους από ένα ζευγάρι γονέων διασταυρώνοντας τα γονίδια τους. Έτσι κάθε απόγονος περιέχει γονίδια και από τους δύο γονείς. Για παράδειγμα αν έχουμε τις δυαδικές συμβολοσειρές

     10000100

     και

     11111111

     μία πιθανή διασταύρωση στην τρίτη θέση θα δημιουργούσε τους εξής δύο απογόνους τον

     10011111

και τον

11100100

3. **Μετάλλαξη**: Αυτός ο τελεστής αλλάζει σε τυχαίες θέσεις τα γονίδια ενός απογόνου. Κάθε γονίδιο αλλάζει ή όχι ανάλογα με μία πιθανότητα. Για παράδειγμα η δυαδική συμβολοσειρά

00000100

έπειτα από μετάλλαξη στην δεύτερη θέση γίνεται

01000100

2.1.2 Δομή Γενετικών Αλγορίθμων

Η δομή ενός γενετικού αλγορίθμου παρουσιάζει κοινά χαρακτηριστικά με την διαδικασία της γενετικής εξέλιξης ενός οργανισμού. Αρχικά ένας γενετικός αλγόριθμος για ένα συγκεκριμένο πρόβλημα περιλαμβάνει έξι συστατικά.

1. **Αρχικοποίηση -Initialization:** Δημιουργία με τυχαίο τρόπο, ενός αρχικού πληθυσμού δυνατών λύσεων.
2. **Συνάρτηση Καταλληλότητας:** Αξιολόγηση κάθε λύσης χρησιμοποιώντας τη συνάρτηση f σαν αντικειμενική συνάρτηση. Δηλαδή ο υπολογισμό του $f(x)$ για κάθε χρωμόσωμα x που ανήκει στον πληθυσμό.
3. Επανάληψη των παρακάτω μέχρι να δημιουργηθούν n απόγονοι
   - **Επιλογή-Selection:** Επιλογή ενός νέου πληθυσμού με βάση την απόδοση κάθε μέλους του προηγούμενου πληθυσμού. Η διαδικασία της επιλογής γίνεται με αντικατάσταση δηλαδή τα ίδια χρωμοσώματα μπορούν να επιλεχθούν παραπάνω από μία φορά.
   - **Διασταύρωση-Crossover:** Η διαδικασία της διασταύρωσης εφαρμόζεται με πιθανότητα p_c σε ένα ζευγάρι χρωμοσωμάτων σε μία τυχαία θέση των χρωμοσωμάτων για την παραγωγή δύο απογόνων. Αν δεν εφαρμοστεί η διαδικασία αυτή τότε οι απόγονοι θα είναι ένας ακριβής κλώνος των γονέων.
   - **Μετάλλαξη-Mutation:** Η διαδικασία της μετάλλαξης εφαρμόζεται με πιθανότητα p_m σε κάθε θέση ενός απογόνου.
4. Αντικατάσταση του συγκεκριμένου πληθυσμού με τον καινούργιο που παράχθηκε μέσω των διαδικασιών της επιλογής, της διασταύρωσης και της μετάλλαξης.
5. Με την ολοκλήρωση του προηγούμενου βήματος επιστρέφουμε στο βήμα 2 για την δημιουργία της νέας γενιάς.
6. Μετά από κάποιο αριθμό γενιών, και αφού καμιά βελτίωση δεν παρατηρείται πλέον, ο Γενετικός Αλγόριθμος τερματίζεται.

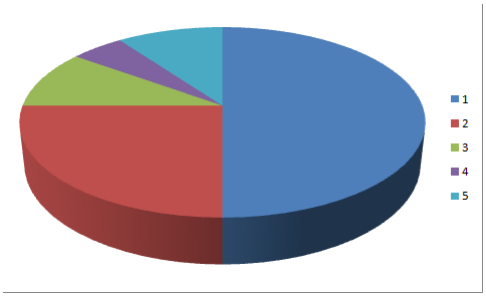
Τα πιο πάνω συστατικά περιγράφονται πιο αναλυτικά.

- **Αρχικοποίηση:**
  Στη φάση της αρχικοποίησης δημιουργούμε έναν αρχικό πληθυσμό από δυνατές λύσεις. Αυτό γίνεται παράγοντας τυχαία $(n \times m)$ στοιχεία, που μπορεί να είναι δυαδικά ψηφία ή πραγματικοί αριθμοί και n είναι το μέγεθος του πληθυσμού που θα επεξεργαστεί ο Γ.Α. Το μέγεθος του πληθυσμού παραμένει σταθερό καθ' όλη τη διάρκεια λειτουργίας του Γ. Α.

- **Επιλογή**
  Η διαδικασία της επιλογής επιλέγει χρωμοσώματα για να δημιουργήσει τον νέο πληθυσμό. Η επιλογή γίνεται με βάση την απόδοση κάθε μέλους (ατόμου - δυνατής λύσης) του πληθυσμού. Έτσι όσο καλύτερο είναι κάποιο μέλος, τόσο μεγαλύτερη πιθανότητα έχει να επιλεγεί και να περάσει στην επόμενη γενεά. Υπάρχουν διάφοροι μηχανισμοί επιλογής Roulette Wheel, Stochastic Uniform, Remainder, Uniform, Tournament. Πριν την εφαρμογή των διαδικασιών αυτών υλοποιούμε το ακόλουθο σχήμα.

- Υπολογίζουμε την απόδοση κάθε μέλους $f(v_i)$ με $i = 1, \cdots, n$
- Η συνολική απόδοση μας δίδεται από τον τύπο $F = \sum_{i=1}^{n} f(v_i)$
- Υπολογίζουμε την πιθανότητα επιλογής p_i κάθε μέλους v_i με $i = 1, \cdots, n$ που είναι $p_i = \frac{f(v_i)}{F}$
- Η αθροιστική πιθανότητα κάθε μέλους v_i είναι $q_i = \sum_{j=1} ip_j$

- **Roulette wheel** Δημιουργούμε μία επιφάνεια ρουλέτας σε σχήμα πίτας όπου κάθε μέλος απεικονίζεται στη ρουλέτα ανάλογα με το ποσοστό της αθροιστικής του πιθανότητας. Για την επιλογή των μελών του νέου πληθυσμού εκτελούμε n περιστροφές της ρουλέτας. Ανάλογα με το ποσοστό που έχει κάθε μέλος πάνω στη ρουλέτα τόσο πιθανό είναι να επιλεγεί. Προφανώς, με αυτή τη μέθοδο επιλογής είναι δυνατόν κάποια μέλη του πληθυσμού να επιλεχθούν περισσότερες από μία φορές. Τα μέλη που είχαν την καλύτερη απόδοση στην προηγούμενη γενιά έχουν τις περισσότερες πιθανότητες να επιλεχθούν.



Σχήμα 2.1: RouletteWheel

- **Stochastic Uniform**
  Ο μηχανισμός αυτός δημιουργεί μία γραμμή στην οποία κάθε γονέας έχει το δικό του μέρος (διάστημα) ανάλογα με το ποσοστό της αθροιστικής του πιθανότητας. Ο αλγόριθμος αυτός κινείται πάνω στην γραμμή με βήματα ίσου μεγέθους και παίρνει τυχαία ένα γονέα πάνω στον οποίο βρέθηκε στο i-στο βήμα.
- **Remainder**
  Η διαδικασία αυτή λαμβάνει το ακέραιο μέρος ενός χρωμοσώματος και έπειτα χρησιμοποιεί τη Roulette Wheel για την επιλογή απογόνων.
- **Tournament**
  Η μέθοδος αυτή πραγματοποιεί ένα είδος αγώνα ανάμεσα σε n χρωμοσώματα και έπειτα επιλέγει το καλύτερο για να δημιουργήσει απογόνους.

► **Διασταύρωση**
Κύριος σκοπός της διασταύρωσης είναι η ανταλλαγή γενετικής πληροφορίας μεταξύ τυχαίων επιλεγμένων γονέων. Η διαδικασία αυτή συνδυάζει μέρη των χρωμοσωμάτων για την παραγωγή νέων απογόνων για την επόμενη γενιά. Υπάρχουν διάφορες διαδικασίες διασταύρωσης όπως η Scattered, Single Point, Two Point, Intermediate, Heuristic, Arithmetic. Αρχικά θεωρούμε ότι η πιθανότητα κάθε μέλους του πληθυσμού να επιλεγεί για διασταύρωση είναι p_c. Για κάθε μέλος του πληθυσμού επιλέγουμε τυχαία έναν πραγματικό αριθμό r όπου $r \in [0, 1]$. Αν $r < p_c$ επιλέγουμε το τρέχον μέλος του πληθυσμού για διασταύρωση και στη συνέχεια πραγματοποιούμε μία από τις επόμενες τεχνικές διασταύρωσης.

- **Single Point**:
  Η διαδικασία Single Point αποτελεί την πιο διαδεδομένη διαδικασία διασταύρωσης. Μετά την επιλογή μελών του πληθυσμού για διασταύρωση , σχηματίζουμε ζευγάρια από μέλη και για κάθε ζευγάρι επιλέγεται τυχαία ένας ακέραιος αριθμός pos στο διάστημα $[1, m-1]$, όπου m είναι

το μήκος σε δυαδικά ψηφία του χρωμοσώματος κάθε μέλους. Ο αριθμός pos προσδιορίζει το σημείο διασταύρωσης. Τα επιλεγμένα ζευγάρια διασταυρώνονται και την θέση τους στον πληθυσμό την παίρνουν οι απόγονοί τους. Για παράδειγμα η διασταύρωση των παρακάτω ατόμων είναι η εξής:

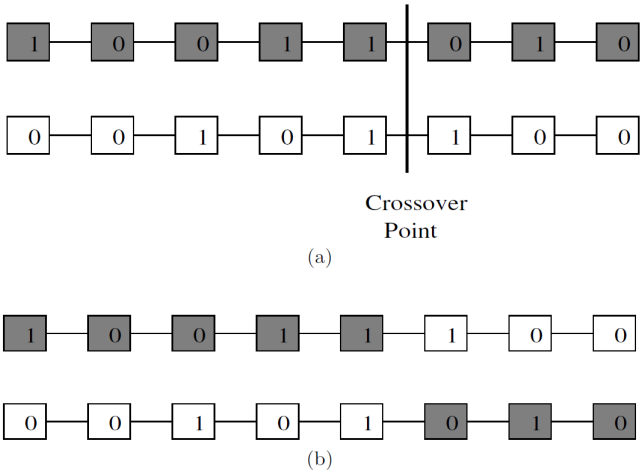
$$(b_1 b_2 \cdots b_{pos} b_{pos+1} \cdots b_m)$$

$$(c_1 c_2 \cdots c_{pos} c_{pos+1} \cdots c_m)$$

δημιουργεί το ακόλουθο ζευγάρι απογόνων:

$$(b_1 b_2 \cdots b_{pos} c_{pos+1} \cdots c_m)$$

$$(c_1 c_2 \cdots c_{pos} b_{pos+1} \cdots b_m)$$



Σχήμα 2.2: Διασταύρωση

- **Scattered:**
  Η διαδικασία Scattered δημιουργεί τυχαία ένα δυαδικό πίνακα μεγέθους ίσου με το μέγεθος των χρωμοσωμάτων και επιλέγει για διασταύρωση τα γονίδια από το πρώτο γονέα για τιμή 1 και τα γονίδια του δεύτερου γονέα για τιμή 0. Για παράδειγμα αν

  $$p_1 = [a\ b\ c\ d\ e\ f\ g\ h]$$

  είναι ο πρώτος γονέας,

  $$p_2 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

  ο δεύτερος γονέας και ο δυαδικός πίνακας είναι ο

  $$[1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]$$

  τότε το παιδί έπειτα από τη διασταύρωση θα είναι το

  $$child = [a\ b\ 3\ 4\ e\ 6\ 7\ 8]$$

- **Two Point:**
  Η διαδικασία αυτή επιλέγει δύο τυχαίους ακεραίους m και n στο διάστημα $[1, l]$ όπου l είναι το μέγεθος των χρωμοσωμάτων. Στη συνέχεια η συνάρτηση επιλέγει m γονίδια από τον πρώτο γονέα, $m + 1$ ως n από τον δεύτερο, και n ως l από τον πρώτο ξανά για να δημιουργήσει το παιδί. Για παράδειγμα έστω

  $$p_1 = [a\ b\ c\ d\ e\ f\ g\ h]$$

είναι ο πρώτος γονέας,

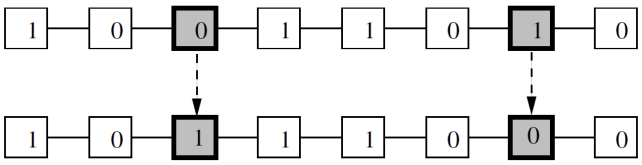
$$p_2 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

ο δεύτερος γονέας και $m = 3$ $n = 6$ τότε δημιουργείται το ακόλουθο παιδί

$$child = [a\ b\ c\ 4\ 5\ 6\ g\ h]$$

- **Intermediate:**
  Η διασταύρωση με αυτή την τεχνική δημιουργεί παιδιά σε συνάρτηση με το μέσο βάρος των γονέων.
- **Arithmetic:**
  Η διασταύρωση με αυτή την τεχνική δημιουργεί παιδιά σε συνάρτηση με το μέσο αριθμητικό βάρος των γονέων.

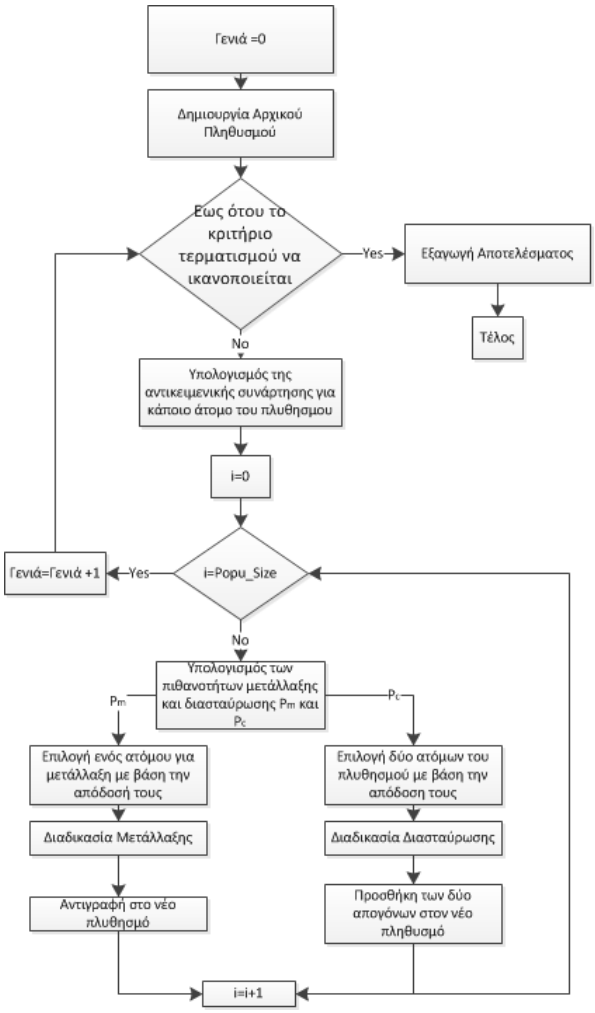
► **Μετάλλαξη**
Η μετάλλαξη είναι μία διαδικασία κατά την οποία εφαρμόζεται μία τυχαία εναλλαγή στην γενετική δομή του χρωμοσώματος. Κύριος σκοπός της είναι να δημιουργήσει μία γενετική διαφοροποίηση στον πληθυσμό των χρωμοσωμάτων και να συμβάλει στην αναζήτηση σε ένα μεγαλύτερο διάστημα λύσεων. Η μετάλλαξη επιλέγει με τυχαίο τρόπο γονίδια από τα χρωμοσώματα των μελών του πληθυσμού και μεταβάλει την τιμή τους. Αν τα γονίδια παίρνουν μόνο δυαδικές τιμές ο τελεστής της μετάλλαξης απλώς τα αντιστρέφει. Κάθε δυαδικό ψηφίο έχει την ίδια πιθανότητα να επιλεγεί προκειμένου να μεταλλαχθεί. Η πιθανότητα αυτή ισούται με την πιθανότητα μετάλλαξης p_m. Η διαδικασία είναι η ακόλουθη.Επιλέγουμε τυχαία έναν πραγματικό αριθμό r όπου $r \in [0, 1]$. Αν $r < p_m$, τότε μεταλλάσσουμε το γονίδιο



Σχήμα 2.3: Μετάλλαξη

Στην περίπτωση που τα χρωμοσώματα λαμβάνουν τιμές από το σύνολο των πραγματικών αριθμών τότε υπάρχουν διάφορες τεχνικές με τις οποίες δημιουργείται η μετάλλαξη.

- **Gausian**:
  Με πιθανότητα p_m η μετάλλαξη Gaussian αλλάζει ένα γονίδιο με έναν αριθμό από την κανονική κατανομή με μέση τιμή 0 και τυπική απόκλιση καθορισμένη από τον χρήστη.
- **Uniform**:
  Η μετάλλαξη Uniform αλλάζει ένα γονίδιο με πιθανότητα p_m με μία τιμή από ένα διάστημα που καθορίζεται από τον χρήστη.
- **Adaptive Feasible**:
  Η διαδικασία αυτή μεταλλάσσει τυχαία τα γονίδια των χρωμοσωμάτων ενός πληθυσμού ανάλογα με την επιτυχία ή την αποτυχία της μετάλλαξης της προηγούμενης γενιάς.



Σχήμα 2.4: Διάγραμμα Γενετικού Αλγόριθμου

2.1.3 Πλεονεκτήματα των Γενετικών Αλγορίθμων

Μερικά από τα σημαντικότερα πλεονεκτήματα που έχει η χρήση Γενετικού Αλγόριθμου για την επίλυση προβλημάτων είναι τα εξής:

- Μπορούν να επιλύουν δύσκολα προβλήματα γρήγορα και αξιόπιστα
- Μπορούν εύκολα να συνεργαστούν με τα υπάρχοντα μοντέλα και συστήματα
- Είναι εύκολα επεκτάσιμοι και εξελίξιμοι

- Μπορούν να συμμετέχουν σε υβριδικές μορφές με άλλες μεθόδους
- Εφαρμόζονται σε πολύ περισσότερα πεδία από κάθε άλλη μέθοδο
- Δεν απαιτούν περιορισμούς στις συναρτήσεις που επεξεργάζονται
- Δεν ενδιαφέρει η σημασία της υπό εξέταση πληροφορίας. Η μόνη "επικοινωνία" του Γενετικού Αλγόριθμου με το περιβάλλον του είναι η αντικειμενική συνάρτηση
- Έχουν από τη φύση τους το στοιχείο του παραλληλισμού
- Είναι μία μέθοδος που κάνει ταυτόχρονα εξερεύνηση του χώρου αναζήτησης και εκμετάλλευση της ήδη επεξεργασμένης πληροφορίας
- Επιδέχονται παράλληλη υλοποίηση

2.1.4 Εφαρμογές των Γενετικών Αλγορίθμων

Αρχικά οι Γενετικοί Αλγόριθμοι αποτέλεσαν αντικείμενο μελέτης αποκλειστικά σε πανεπιστήμια και ερευνητικά κέντρα. Τα τελευταία χρόνια η αυξανόμενη ζήτηση αποδοτικών εφαρμογών βελτιστοποίησης οδήγησε στη χρήση των Γενετικών Αλγόριθμων σε ένα ευρύ φάσμα πεδίων με εντυπωσιακά αποτελέσματα. Τα παρακάτω πεδία είναι κάποια ενδεικτικά που χρησιμοποιούνται οι Γενετικοί Αλγόριθμοι.

- **Βελτιστοποίηση**: Οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί σε διάφορα προβλήματα βελτιστοποίησης όπως αριθμητική βελτιστοποίηση ή βελτιστοποίηση προβλημάτων συνδιαστικής όπως σχεδίαση κυκλωμάτων και προγραμματισμός job-shop.
- **Automatic Programming**: Οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί για την εξέλιξη των προγραμμάτων υπολογιστών σε συγκεκριμένους τομείς.
- **Machine Learning**: Οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί σε πολλές Machine Learning εφαρμογές όπως προβλήματα ταξινόμησης και πρόβλεψης.
- **Οικονομικά Μοντέλα**: Οι γενετικοί αλγόριθμου έχουν χρησιμοποιηθεί στην μοντελοποίηση διαδικασιών καινοτομίας, στην ανάπτυξη στρατηγικών προσφορών και στη συγχώνευση αγορών.
- **Μοντέλα Ανοσοποιητικού Συστήματος**: έχουν χρησιμοποιηθεί στην ανάπτυξη διάφορων μοντέλων του ανοσοποιητικού συστήματος όπως τη σωματική μετάλλαξη ενός ατόμου κατά τη διάρκεια ζωής του.
- **Οικολογικά Μοντέλα**: Έχουν χρησιμοποιηθεί στη μοντελοποίηση διάφορων οικολογικών μοντέλων.

2.2 Γενετικοί Αλγόριθμοι Ταξινόμησης

Το πρόβλημα της ταξινόμησης όπως έχει αναφερθεί προηγουμένως είναι η διαδικασία της ταξινόμησης ενός προτύπου που παρουσιάζει ορισμένα χαρακτηριστικά σε μία ή περισσότερες κλάσεις. Στην περίπτωση που ο ταξινομητής έχει σχεδιαστεί σε ένα σύνολο δεδομένων που οι κλάσεις είναι γνωστές τότε αυτό είναι ένα πρόβλημα μάθησης με επίβλεψη.

Η διαδικασία της ταξινόμησης έχει μοντελοποιηθεί με διάφορους τρόπους και τεχνικές όπως τα Δέντρα Αποφάσεων, τα Τεχνητά Νευρωνικά Δίκτυα, τα Τεχνητά Ανοσοποιητικά Συστήματα και άλλα. Στον τομέα αυτό οι Γενετικοί Αλγόριθμοι δεν έχουν ερευνηθεί σε βάθος ως προς την ικανότητα τους στην προσέγγιση προβλημάτων ταξινόμησης. Υπάρχουν ορισμένες υλοποιήσεις που χρησιμοποιούν τους γενετικούς αλγορίθμους ως βοηθητικό εργαλείο για την υλοποίηση ενός μέρους της ταξινόμησης αλλά δεν αποτελούν τον πυρήνα της μεθόδου.

Μερικές από τις προσπάθειες αυτές είναι του James and Lawrence [22] (A hybrid Genetic Algorithm) με τον υβριδικό αλγόριθμο GA-WKNN που είναι μία βελτιωμένη έκδοση του γνωστού αλγόριθμου των k

πλησιέστερων γειτόνων με βάρη χρησιμοποιώντας την διαδικασία του γενετικού αλγορίθμου. Μία ακόμη προσπάθεια βρίσκεται στο Bandyopadhyay and Pal [1] στο οποίο περιγράφονται αλγόριθμοι ταξινόμησης χρησιμοποιώντας γενετικούς αλγορίθμους. Ο αλγόριθμος που έχει εξαιρετικό ενδιαφέρον είναι ο GA-Classifier που είναι ένας γενετικός αλγόριθμος που παράγει υπερ-επίπεδα για τον διαχωρισμό των στοιχείων ενός συνόλου εκπαίδευσης σε κλάσεις.

2.2.1 Αλγόριθμος GA-WKNN

Ο αλγόριθμος GA-WKNN [22] βασίζεται σε μία παραλλαγή του αλγορίθμου των κ πλησιέστερων γειτόνων τον WKNN. Ο WKNN προσθέτει ένα πίνακα βαρών στην διαδικασία του K-NN έτσι ώστε να επιφέρει μία βελτιστοποίηση στα παραγόμενα αποτελέσματα. Πιο συγκεκριμένα ο K-NN ταξινομεί ένα αντικείμενο του συνόλου δεδομένων ανάλογα με την απόσταση ως προς τα αντικείμενα κάθε κλάσης. Ο αλγόριθμος αυτός είναι εξαιρετικά αποτελεσματικός όταν τα χαρακτηριστικά των αντικειμένων που επεξεργάζεται είναι εξίσου σημαντικά ως προς την ταξινόμηση. Όταν αυτό δεν ισχύει τότε ο συγκεκριμένος αλγόριθμος δεν συμπεριφέρεται καλά. Από την άλλη πλευρά, ο WKNN προτείνει την δημιουργία ενός πίνακα ο οποίος περιέχει βάρη ανάλογα με τη συνεισφορά κάποιου χαρακτηριστικού στη συνολική απόδοση της ταξινόμησης. Ο Γενετικός Αλγόριθμος χρησιμοποιείται στην εύρεση ενός βέλτιστου πίνακα βαρών.

Αρχικά ο WKNN αντιστοιχεί σε κάθε χαρακτηριστικό ένα βάρος ανάλογα με τη συνεισφορά του. Για παράδειγμα, αν ένα χαρακτηριστικό είναι καθοριστικής σημασίας, στην εύρεση της κλάσης στην οποία ανήκει τότε το βάρος του θα είναι μεγαλύτερο από κάποιο άλλο.

Η μετρική που χρησιμοποιείται στον WKNN βασίζεται σε μία διαφοροποίηση της Ευκλείδιας μετρικής έτσι ώστε να περιέχει και την πληροφορία των βαρών.

$$d_{ij} = \{\sum_{a=1}^{n} w_a (x_{ia} - x_{ja})^2\}^{\frac{1}{2}} \tag{2.1}$$

Επιπλέον διαφοροποιείται η διαδικασία κατά την οποία ο KNN προσδιορίζει την κλάση ενός στοιχείου. Η επιλογή της κλάσης ενός στοιχείου βασίζεται στην παρατήρηση ότι θα πρέπει να υπάρχει ένας γείτονας από τους k, ο οποίος θα έχει μεγαλύτερη επιρροή στην ταξινόμηση του σε κάποια κλάση. Έτσι είναι πιθανό ο γείτονας που είναι πιο κοντά στο στοιχείο να έχει μεγαλύτερη επιρροή στην ταξινόμηση του από το k-στο γείτονα. Με αυτό τον τρόπο συσχετίζουμε τα βάρη των στοιχείων που ορίσαμε προηγουμένως με την επιλογή της κλάσης στην οποία ταξινομούνται. Ορίζουμε την ακόλουθη παράμετρο την οποία ονομάζουμε Vote

$$V_i = \sum_{i=1}^{k} \begin{cases} W_j & \text{if } C_j = i \\ 0 & \text{if } C_j \neq i \end{cases} \tag{2.2}$$

με W_j είναι το βάρος του j-στου γείτονα και C_j η κλάση του j-στου γείτονα.

Ο GA-WKNN συνδυάζει τις δυνατότητες βελτιστοποίησης των γενετικών αλγορίθμων με τις δυνατότητες ταξινόμησης του αλγόριθμου των k πλησιέστερων γειτόνων. Με αυτό τον τρόπο ο γενετικός αλγόριθμος χρησιμοποιείτε για την εύρεση του βέλτιστου πίνακα βαρών δηλαδή του πίνακα αυτού που προκαλεί τη μεγαλύτερη ακρίβεια ταξινόμησης.

Η δομή του αλγορίθμου σκιαγραφείται παρακάτω:

- Τα χρωμοσώματα παριστάνονται με διανύσματα πραγματικών τιμών στο διάστημα $[0, 1]$ που αντιπροσωπεύουν τα βάρη. Οι τιμές αυτές παράγονται τυχαία για κάθε γενιά.
- Η απόδοση των χρωμοσωμάτων υπολογίζεται μέσω τις σχέσης 2.2 για τον υπολογισμό των βαρών και προσδιορίζονται οι κ πλησιέστεροι γείτονες μέσω της σχέσης 2.1 .

► Τέλος χρησιμοποιείται μία αντικειμενική συνάρτηση f που αθροίζει όλα τα δεδομένα που έχουν ταξινομηθεί σε λάθος κλάση. Ο γενετικός αλγόριθμος χρησιμοποιείται για την ελαχιστοποίηση της f δηλαδή ελαχιστοποιεί τα δεδομένα που ταξινομούνται λάθος.

2.2.2 GA-Classifier

Ο GA-Classifier [1] υλοποιεί ένα σύνολο H από υπερεπίπεδα στον χώρο αναζήτησης έτσι ώστε να ελαχιστοποιήσει τον αριθμό των σημείων που ταξινομούνται λανθασμένα. Η εξίσωση ενός υπερεπιπέδου στον N-διάστατο χώρο $(X_1, X_2, \cdots, X_n)$ δίνεται από την ακόλουθη εξίσωση.

$$x_N \cos\alpha_{N-1} + \beta_{N-1} \sin\alpha_{N-1} = d \tag{2.3}$$

με

$$\begin{aligned}
&\beta_{N-1} = x_{N-1} \cos\alpha_{N-2} + \beta_{N-2} \sin\alpha_{N-2},\\
&\beta_{N-2} = x_{N-2} \cos\alpha_{N-3} + \beta_{N-3} \sin\alpha_{N-3}\\
&\vdots\\
&\beta_1 = x_1 \cos\alpha_0 + \beta_0 \sin\alpha_0
\end{aligned}$$

με παραμέτρους
X_i: το i-οστό χαρακτηριστικό (ή i-οστή διάσταση)
$(x_1, x_2, \cdots, x_N)$: ένα σημείο στο υπερεπίπεδο
α_{N-1}: Η γωνία που σχηματίζει η κανονικοποιημένη μονάδα του υπερεπιπέδου με τον άξονα των X_N
α_{N-2}: Η γωνία που σχηματίζει η προβολή της κανονικοποιημένης μονάδας στον $(X_1 - X_2 - \cdots X_{N-1})$ χώρο με τον άξονα X_{N-1}
$\vdots$
α_1: Η γωνία που σχηματίζει η προβολή της κανονικοποιημένης μονάδας στον $(X_1 - X_2)$ χώρο με τον άξονα X_2
α_0: Η γωνία που σχηματίζει η προβολή της κανονικοποιημένης μονάδας στον (X_1) επίπεδο με τον άξονα X_1
d: Η κάθετη απόσταση του υπερεπιπέδου από την αρχή των αξόνων
Με αυτόν τον τρόπο η N-άδα $< \alpha_1, \alpha_2, \cdots, \alpha_N - 1, d >$ ορίζει ένα επίπεδο N-διαστάσεων.
Κάθε γωνία a_j με $j = 1, 2, \cdots, N-1$ κινείται στο διάστημα $[0, 2\pi]$. Έτσι αν θεωρήσουμε ότι b_1 bits χρησιμοποιούνται για την αναπαράσταση μίας γωνίας τότε οι επιτρεπόμενες τιμές των a_j είναι

$$0, \delta * 2\pi, 2\delta * 2\pi, 3\delta * 2\pi, \cdots, (2^{b_1} - 1) * 2\pi$$

με $\delta = \frac{1}{2^{b_1}}$. Συνεπώς αν b_1 bits χρησιμοποιούνται για την αναπαράσταση μίας γωνίας στη δεκαδική του μορφή είναι v_1 και τότε η γωνία δίνεται από τον τύπο $v_1 * \delta * 2\pi$.

Όταν οι γωνίες σταθεροποιούνται τότε και το υπερεπίπεδο σταθεροποιείται σε ένα σημείο. Για τον καθορισμό του υπερεπιπέδου μένει να καθορίσουμε το d. Για το λόγο αυτό θεωρούμε ένα υπερορθογώνιο το οποίο περικλείει τα σημεία του συνόλου εκπαίδευσης. Έστω $({x_i}^{\min}, {x_i}^{\max})$ η μέγιστη και η ελάχιστη τιμή των χαρακτηριστικών του συνόλου εκπαίδευσης X_i. Τότε τα διανύσματα που καθορίζουν το υπερορθογώνιο είναι τα

$${x_1}^{ch1}, {x_2}^{ch2}, \cdots, {x_N}^{chN}$$

με ch_i, με $i = 1, 2, \cdots, N$ μπορούν να είναι ή ${x_i}^{\min}$ ή ${x_i}^{\max}$ δηλαδή το πλήθος των διανυσμάτων θα είναι 2^N.
Έστω $diag$ η διαγώνιος του υπερορθογώνιου με

$$diag = \sqrt{({x_1}^{\min} - {x_1}^{\max})^2 + ({x_2}^{\min} - {x_2}^{\max})^2 + \cdots + ({x_N}^{\min} - {x_N}^{\max})^2}$$

Ένα υπερεπίπεδο είναι υπερεπίπεδο βάσης με ένα δεδομένο προσανατολισμό όταν:

1. Έχει ίδιο προσανατολισμό
2. Περνάει από ένα διάνυσμα του υπερορθογωνίου
3. Η κάθετη απόσταση από την αρχή των αξόνων είναι η ελάχιστη. Έστω η απόσταση αυτή $d_{\min}$

Αν b_2 bits χρησιμοποιούνται για την αναπαράσταση του d και έστω η δεκαδική τιμή του d η v_2. Τότε η v_2 καθορίζει ένα υπερεπίπεδο με αυτόν τον προσανατολισμό και για τον οποίο το d δίνεται από την σχέση $d_{\min} + \frac{diag}{2^{b_2}} * v_2$. Με αυτό τον τρόπο κάθε χρωμόσωμα έχει μέγεθος $l = H((N-1)b_1 + b_2$ με H ο αριθμός των υπερεπιπέδων. Κάθε χρωμόσωμα χαρακτηρίζεται από μια δυαδική συμβολοσειρά μεγέθους l. Η απόδοση του κάθε χρωμοσώματος εξαρτάται από τον βαθμό ταξινόμησής του. Μια συμβολοσειρά str_i με $i = 1, 2, \cdots, Pop$ και Pop το σύνολο του πληθυσμού των χρωμοσωμάτων αναπαριστά H υπερεπίπεδα που συμβολίζονται ως ${pln_j}^i$ στον N-διάστατο χώρο και $j = 1, 2, \cdots, H$.
Για κάθε σημείο από το σύνολο εκπαίδευσης ${x_1}^m, {x_2}^m, \cdots, {x_N}^m$ υπολογίζουμε το πρόσημό του σε σχέση με το υπερεπίπεδο ${pln_j}^i$. Το πρόσημο αυτό προκύπτει από την παρακάτω παράσταση.

$$x_m \cos {\alpha_{N-1}}^{ij} + {\beta_{N-1}}^{ij} \sin {\alpha_{N-1}}^{ij} - d^{ij}$$

Το πρόσημο αυτό είναι θετικό ή αρνητικό (ανάλογα 0 ή 1) ανάλογα αν το σημείο βρίσκεται στην θετική πλευρά του υπερεπιπέδου ή όχι. Αυτή η διαδικασία επαναλαμβάνεται για όλα τα υπερεπίπεδα και στο τέλος παίρνουμε μια συμβολοσειρά μήκους Η την $sign\_string$. Η $sign\_string$ αναφέρεται στην ταξινόμηση που προέκυψε στο σημείο m, δηλαδή σε ποια τάξη ανήκει.

Για τον υπολογισμό των σημείων που ταξινομήθηκαν λάθος χρησιμοποιούμε ένα βοηθητικό πίνακα τον $class\_matrix$ με $class\_matrix = 2 * k$. Κάθε γραμμή του πίνακα περιέχει την δεκαδική τιμή της δυαδικής συμβολοσειράς του εκάστοτε $sign\_string$ και κάθε στήλη έχει ένα αριθμό από το 1 ως το K δηλαδή τις κλάσεις του συνόλου δεδομένων μας. Αν η κλάση του σημείου είναι η i-οστή, τότε η τιμή στη θέση $class\_matrix[dec(sign\_string), i]$ αυξάνεται κατά 1. Αυτή η διαδικασία επαναλαμβάνεται για κάθε σημείο του συνόλου εκπαίδευσης.

Τέλος για να συνδέσουμε τις τοποθεσίες του χώρου αναζήτησης με μια συγκεκριμένη κλάση λαμβάνουμε ως σημείο αναφοράς την πρώτη στήλη του πίνακα $class\_matrix$ που έχει 1 και 0. Με άλλα λόγια

$$Class(Reg_i) = argmax_{p=1,2\cdots,K}\ class\_matrix[dec(Reg_i), p]$$

έτσι ώστε $class\_matrix[dec(Reg_i), 0] \neq 0$

Κεφάλαιο 3

Τεχνητά Ανοσοποιητικά Συστήματα

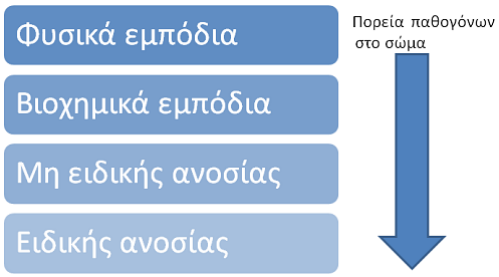
3.1 Εισαγωγή

Τεχνητά Ανοσοποιητικά Συστήματα (ΤΑΣ) είναι ένα ξεχωριστό πεδίο έρευνας που προσπαθεί να συνδέσει την ανοσολογία με την μηχανική, και την πληροφορική μέσω της υπολογιστικής μοντελοποίησης. Είναι υπολογιστικά συστήματα τα οποία, αναπτύχθηκαν εμπνευσμένα από τις αρχές και τη λειτουργία του ανοσοποιητικού συστήματος. Το ανοσοποιητικό σύστημα των σπονδυλωτών οργανισμών αποτελείται από ένα πολύπλοκο δίκτυο από εξειδικευμένα μέλη όπως οι ιστοί, τα όργανα, τα κύτταρα και τα χημικά μόρια. Ορισμένες δυνατότητες του φυσικού ανοσοποιητικού συστήματος είναι η αναγνώριση, η καταστροφή και η διατήρηση ιστορικού ενός σχεδόν ακαθόριστου πλήθους ξένων σωματιδίων καθώς και η προστασία του οργανισμού από παθογόνους οργανισμούς. Μια από τις πιο βασικές διεργασίες του ανοσοποιητικού συστήματος είναι η αναγνώριση του εαυτού από τον μη εαυτό. Τα Τεχνητά Ανοσοποιητικά Συστήματα είναι μία προσπάθεια μοντελοποίησης αυτών των μηχανισμών του ανοσοποιητικού συστήματος. Για αυτό το λόγο, έχουν αναπτυχθεί πολλές θεωρίες σχετικά με τον τρόπο που το ανοσοποιητικό σύστημα υλοποιεί τους μηχανισμούς αυτούς. Μερικές από αυτές είναι, η θεωρία της επιλογής των κλώνων, η θεωρία της αρνητική επιλογής, η θεωρία ανοσοποιητικού δικτύου, η θεωρία κινδύνου και πολλές άλλες. Πριν προχωρήσουμε στην περιγραφή του τομέα των Τεχνητών Ανοσοποιητικών Συστημάτων, είναι απαραίτητο να περιγράψουμε την βασική δομή και λειτουργία του ανοσοποιητικού συστήματος.

Δομή του Ανοσοποιητικού Συστήματος

Το ανοσοποιητικό σύστημα είναι ο αμυντικός μηχανισμός του ανθρώπου και γενικά των σπονδυλωτών οργανισμών ενάντια σε παθογόνους μικροοργανισμούς. Περιλαμβάνει ένα μεγάλο πλήθος από εξειδικευμένα κύτταρα και μόρια τα οποία παράγονται και ωριμάζουν με σκοπό την αντιμετώπιση εξωγενών βλαβερών μικροοργανισμών. Η αλληλεπίδραση ανάμεσα στα ανοσοποιητικά κύτταρα και άλλα κύτταρα του οργανισμού, συνθέτουν ένα πλούσιο και πολύπλοκο περιβάλλον που έχουν ως αποτέλεσμα την άμυνα του οργανισμού μας. Το ανοσοποιητικό σύστημα μπορεί να χωριστεί σε τέσσερα επίπεδα (3.1).

- Φυσικά εμπόδια: δέρμα, τρίχες
- Βιοχημικά εμπόδια: βιολογικά υγρά, ιδρώτας
- Μη ειδικής ανοσίας
- Ειδικής ανοσίας

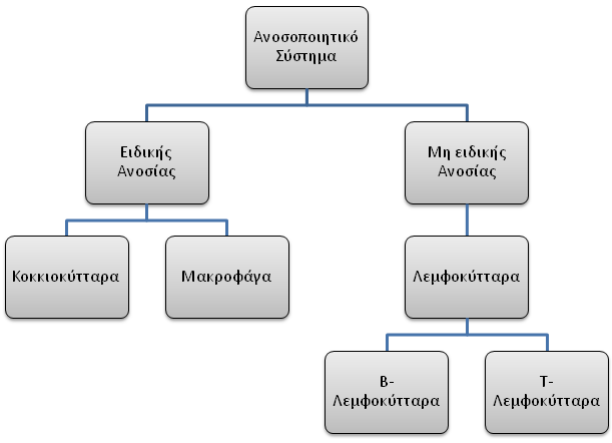


Σχήμα 3.1: Επίπεδα Ανοσοποιητικού Συστήματος

Τα δύο τελευταία επίπεδα αποτελούν τις κύριες γραμμές άμυνας του Ανοσοποιητικού Συστήματος.

Το σύστημα μη ειδικής ανοσίας είναι η πρώτη γραμμή άμυνας των σπονδυλωτών οργανισμών από παθογόνους μικροοργανισμούς. Αυτός ο μηχανισμός είναι ο βασικός αμυντικός μηχανισμός, ο οποίος δεν μεταβάλετε κατά τη διάρκεια της ζωής ενός ανθρώπου και περιλαμβάνει την φλεγμονώδη αντίδραση και την φαγοκυττάρωση (δηλαδή την κατάποση παθογόνων οργανισμών από εξειδικευμένα κύτταρα). Από την άλλη πλευρά, το σύστημα ειδικής ανοσίας αναπτύσσεται όσο ο οργανισμός αναπτύσσεται και έχει τη δυνατότητα να μεταβάλλεται. Αυτό έχει ως αποτέλεσμα, να αναγνωρίζει παθογόνους οργανισμούς και να τους θυμάται για μελλοντική αντιμετώπισή τους. Τα συστήματα ειδικής και μη ειδικής ανοσίας λειτουργούν με διαφορετική χρονική διάρκεια το καθένα. Το σύστημα μη ειδικής ανοσίας λειτουργεί αρχικά για ένα μικρό χρονικό διάστημα μόλις εντοπίσει τον παθογόνο οργανισμό, ενώ το σύστημα ειδικής ανοσίας μπορεί να διαρκέσει μέρες.

Τα συστήματα ειδικής και μη ειδικής ανοσίας περιέχουν ορισμένα εξειδικευμένα κύτταρα τα οποία επικοινωνούν μέσω συνδέσεων, είτε κύτταρο με κύτταρο, είτε μέσω κυτοκίνων. Αυτή η επικοινωνία γίνεται μέσω ειδικών πρωτεϊνικών μορίων που λέγονται υποδοχείς και υπάρχουν στην επιφάνεια των κυττάρων. Όταν υπάρχει μία ισχυρή χημική επικοινωνία μεταξύ δύο υποδοχέων τότε παράγεται ένα σήμα στο κύτταρο το οποίο του επιτρέπει την αναγνώρισή του. Οι υποδοχείς αυτοί ονομάζονται αντιγονικοί υποδοχείς ενώ τα μόρια που τους συνδέουν ονομάζονται αντιγόνα.

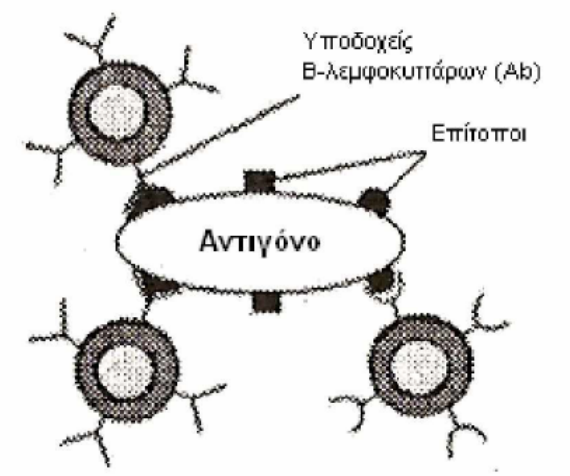


Σχήμα 3.2: Δομή Ανοσοποιητικού Συστήματος

Τα βασικά συστατικά του συστήματος ειδικής ανοσίας είναι τα Τ-λεμφοκύτταρα και τα Β-λεμφοκύτταρα (3.2). Γενικά, τα λεμφοκύτταρα είναι τα μόνα κύτταρα, τα οποία παράγουν αντιγονικούς υποδοχείς, με κάθε λεμφοκύτταρο να παράγει διαφορετικό αντιγονικό υποδοχέα από τους άλλους. Τα λεμφοκύτταρα παράγο-

νται στον μυελό των οστών και διαφοροποιούνται στα διάφορα όργανα του λεμφικού συστήματος. Όταν ο οργανισμός δεν είναι μολυσμένος, τα λεμφοκύτταρα βρίσκονται σε κατάσταση ηρεμίας και απλά κυκλοφορούν στο αίμα και στους λεμφαδένες.

Κύριος ρόλος των Β-λεμφοκυττάρων είναι η παραγωγή και η έκκριση αντισωμάτων. Τα αντισώματα είναι πρωτεΐνες που συνδέονται σε συγκεκριμένα αντιγόνα, που βρίσκονται στην επιφάνεια του παθογόνου οργανισμού. Με αυτό τον τρόπο είτε εξασθενούν τον παθογόνο οργανισμό είτε ενεργοποιούν άλλα κύτταρα του ανοσοποιητικού συστήματος τα οποία εν συνεχεία τα καταστρέφουν. Κάθε Β-λεμφοκύτταρο παράγει μόνο ένα είδος αντισώματος, το οποίο μπορεί να αναγνωρίσει μόνο ένα συγκεκριμένο είδος αντιγόνου. Τα αντισώματα μπορούν είτε να προσδεθούν στην επιφάνεια των Β-λεμφοκυττάρων είτε σε μόρια ελεύθερα σε διάλυμα (3.3). Όταν γίνεται η σύνδεση ενός αντιγόνου με ένα αντίσωμα τότε το Β-λεμφοκύτταρο ενεργοποιείται και πολλαπλασιάζεται. Κάποιοι κλώνοι αυτού του Β-λεμφοκυττάρου μπορεί να είναι τα λεμφοκύτταρα μνήμης τα οποία παρέχουν προστασία από μελλοντικές επιθέσεις του ίδιου παθογόνου.



Σχήμα 3.3: Πρόσδεση Β-λεμφοκυττάρων σε αντιγόνο

Τα Τ-λεμφοκύτταρα ρυθμίζουν την συμπεριφορά άλλων κυττάρων και παράλληλα επιτίθενται άμεσα στα μολυσμένα κύτταρα του οργανισμού. Ονομάζονται έτσι γιατί ωριμάζουν στο θύμο αδένα. Γενικά, τα Τ-λεμφοκύτταρα ρυθμίζουν την ανοσολογική αντίδραση εκκρίνοντας ουσίες, τις κυτοκίνες ή πιο συγκεκριμένα τις λεμφοκίνες, οι οποίες ειδοποιούν και ενεργοποιούν άλλα κύτταρα, ενώ παράλληλα μπορούν να εξοντώσουν και τα κύτταρα στόχους.

Όταν ένας αντιγονικός οργανισμός εισβάλει για πρώτη φορά στο ανοσοποιητικό σύστημα τότε θα υπάρξει μία αντίδραση του οργανισμού παράγοντας ένα μεγάλο πλήθος αντισωμάτων. Αφού η μόλυνση εξαλειφθεί, μια μνήμη των επιτυχών υποδοχέων διατηρείται προκειμένου να επιτραπεί η γρηγορότερη απόκριση σε περίπτωση που ίδια ή παρόμοια παθογόνα εισβάλλουν στο σώμα. Σε περίπτωση που ο οργανισμός ξαναμολυνθεί από τον ίδιο παθογόνο οργανισμό ή από μια μεταλλαγμένη εκδοχή του τότε το ανοσοποιητικό σύστημα παράγει γρηγορότερα ένα πληθυσμό από σχετικά αντισώματα με αποτέλεσμα την απομάκρυνση του αντιγόνου αποτελεσματικότερα. Επομένως εάν μια μεταλλαγμένη έκδοση του αρχικού παθογόνου αντιμετωπιστεί, το ανοσοποιητικό σύστημα είναι ήδη εν μέρει προσαρμοσμένο έτσι ώστε να το αντιμετωπίσει, βασισμένο στην προηγούμενη γνώση του. Αν και υπάρχει διαμάχη γύρω από τον ακριβή τρόπο με τον οποίο η μνήμη του ανοσοποιητικού συστήματος διατηρείται, γενικά θα μπορούσαμε να πούμε ότι το ανοσοποιητικό σύστημα διατηρεί έναν πληθυσμό μακρόβιων λεμφοκυττάρων ή κυττάρων μνήμης. Τα Τ και Β κύτταρα έχουν διαφορετικές εκδοχές μνήμης. Η δημιουργία των κυττάρων μνήμης εξασφαλίζουν ότι τα

αποτελέσματα της εκμάθησης του παρελθόντος κωδικοποιούνται στον τρέχοντα πληθυσμό των λεμφοκυττάρων.

3.2 Βασικά Στοιχεία των Τεχνητών Ανοσοποιητικών Συστημάτων

Σύμφωνα με τον ορισμό του de Castro and Timmis [8]:

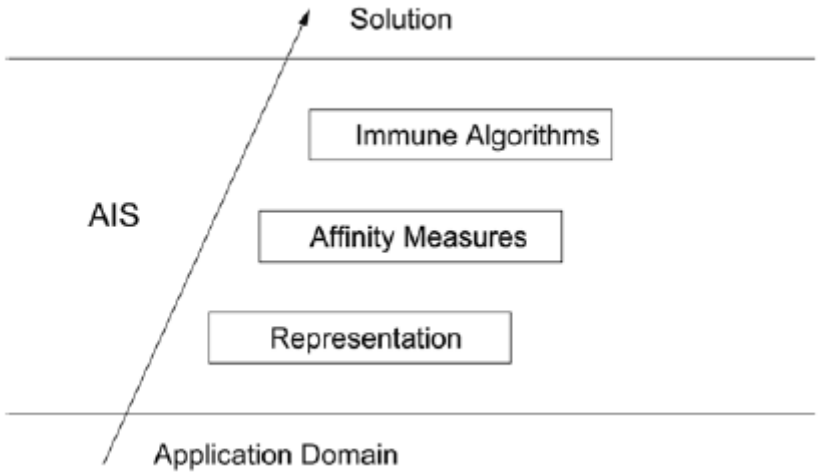
Ορισμός 3 *Τα Τεχνητά Ανοσοποιητικά Συστήματα, είναι προσαρμοστικά συστήματα εμπνευσμένα από τη θεωρία της ανοσολογίας και από τις αρχές, τους μηχανισμούς και τα μοντέλα του ανοσοποιητικού συστήματος, τα οποία εφαρμόζονται στην επίλυση προβλημάτων.*

Τις δυνατότητες και τα χαρακτηριστικά που έχουν δανειστεί τα Τεχνητά Ανοσοποιητικά Συστήματα από το φυσικό ανοσοποιητικό είναι τα ακόλουθα:

- Το γεγονός ότι χρειάζεται να γνωρίζει μόνο τα φυσιολογικά κύτταρα (self)
- Τη δυνατότητα αναγνώρισης των κυττάρων εαυτού (self) από τα ξένα κύτταρα (non self)
- Τη δυνατότητα αναγνώρισης προσδιορισμού ενός ξένου κυττάρου ως επιβλαβές ή μη επιβλαβές
- Την ικανότητα των λεμφοκυττάρων να κλωνοποιούνται και να μεταλλάσσονται προκειμένου να προσαρμόζονται στα ξένα κύτταρα που αντιμετωπίζει ο οργανισμός
- Την ικανότητα γρήγορης αντίδρασης σε αντιγόνα που ο οργανισμός έχει ήδη αντιμετωπίσει που οφείλεται στα κύτταρα μνήμης
- Τη δημιουργία δικτύων λεμφοκυττάρων σαν αποτέλεσμα της συνεργασίας μεταξύ των λεμφοκυττάρων

Στη συνέχεια έχει προταθεί ότι για την υλοποίηση των Τεχνητών Ανοσοποιητικών Συστημάτων είναι απαραίτητη η θεμελίωση ενός πλαισίου, το οποίο θα πρέπει να ακολουθούν όλα τα Τεχνητά Ανοσοποιητικά Συστήματα. Έτσι τα βασικά στοιχεία τα οποία πρέπει να πληρεί η δομή ενός Τεχνητού Ανοσοποιητικού Συστήματος είναι τα εξής:

- **Αναπαράσταση Συστατικών** : Η αναπαράσταση των συστατικών του συστήματός μας
- **Μέτρα έλξης** : Ένα σύνολο μηχανισμών για τον υπολογισμό της αλληλεπίδρασης των συστατικών μας με άλλα
- **Διαδικασίες προσαρμογής** : Η διαδικασία κατά την οποία τα συστατικά μας αλληλεπιδρούν με τη δυναμική του συστήματος



Η αναπαράσταση συστατικών σε ένα Τεχνητό Ανοσοποιητικό Σύστημα μπορεί να αναπαρασταθεί ως εξής. Ένα αντιγόνο ή ένα αντίσωμα μπορεί να αναπαρασταθεί σαν ένα διάνυσμα στον N-διάστατο χώρο. Έτσι αν θεωρήσουμε ένα αντιγόνο ως Ab και ένα αντίσωμα ως Ag, η κατάσταση κατά την οποία το αντιγόνο και το παθογόνο ταιριάζουν τέλεια περιγράφεται ως εξής $Ab = Ag$. Το συμπληρωματικό του Ab ή του Ag μπορεί να μετρηθεί ως η απόσταση του Ab και του Ag χρησιμοποιώντας μία κατάλληλη μετρική. Η μετρική αυτή, εκφράζει το μέτρο συγγένειας των δύο αυτών στοιχείων δηλαδή το μέτρο έλξης. Για την περιγραφή του μεγέθους αυτού χρησιμοποιήθηκε η έννοια του σχηματο-χώρου.

Η βασική ιδέα της έννοιας του σχηματο-χώρου είναι ότι ανάλογα με τον βαθμό που ταιριάζουν οι υποδοχείς του ανοσοποιητικού κυττάρου με το επίτοπο, τόσο ισχυρή είναι η έλξη μεταξύ τους. Αυτό οφείλεται στον τρόπο με τον οποίο τα μόρια τους, ταιριάζουν μεταξύ τους. Δηλαδή, αν ταιριάζουν ή όχι αυτά τα μόρια οφείλεται στο σχήμα τους. Το μέγεθος της έλξης έχει σχέση με το μέγεθος και το είδος των κλώνων που θα παραχθούν για την αντιμετώπισή ενός παθογόνου οργανισμού.

Για να περιγράψουμε το μέγεθος έλξης μπορούμε να χρησιμοποιήσουμε διάφορες μετρικές. Μία μετρική που χρησιμοποιείται είναι η ευκλείδεια μετρική. Αν χρησιμοποιήσουμε n πραγματικούς αριθμούς για να περιγράψουμε την έλξη μεταξύ κάθε σύνδεσης, τότε ο σχηματο-χώρος είναι ένα υποσύνολο του $\mathbb{R}^n$. Έστω ένας υποδοχέας με συντεταγμένες $x = (x_1, x_2, \cdots, x_n)$ και ένα επίτοπο με συντεταγμένες $y = (y_1, y_2, \cdots, y_n)$ η Ευκλείδεια μετρική θα μας δώσει ένα ποσό έλξης μεταξύ τους που υπολογίζεται από τον τύπο της απόστασης στον Ευκλείδειο χώρο.

$$D(x, y) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2}$$

Ένας άλλος τρόπος για την μέτρηση της έλξης είναι να θεωρήσουμε μία μπάλα ακτίνας ϵ με ϵ οσοδήποτε μικρό και κέντρο x. Τα στοιχεία που βρίσκονται στο εσωτερικό της μπάλας παρουσιάζουν μεγάλη έλξη, ώστε να ενεργοποιήσουν το ανοσοποιητικό σύστημα, ενώ αυτά που ανήκουν στο συμπλήρωμα δεν παρουσιάζουν μεγάλη έλξη. Θεωρούμε την μπάλα με κέντρο x και ακτίνα ϵ και τα στοιχεία που ανήκουν σε αυτή, y για τα οποία ισχύει $d(x, y) < \epsilon$.

$$B_\epsilon(x) = \{y | D(x, y) < \epsilon\}$$

Σε αυτή την περίπτωση η μετρική έλξης είναι η εξής:

$$D(x,y) = \begin{cases} 1 & : y \in B_\epsilon(x) \\ 0 & : y \notin B_\epsilon(x) \end{cases}$$

Ένας τελευταίος και πιο αποτελεσματικός τρόπος μέτρησης της έλξης είναι χρησιμοποιώντας τον σχηματοχώρο Hamming. Ο σχηματοχώρος Hamming Σ^L αποτελείται από L στοιχεία ενός πεπερασμένου αλφαβήτου Σ. Όπως παρουσιάζεται και στο παρακάτω σχήμα το αριστερό σκέλος αποτελείται από ένα σχηματοχώρο Hamming μεγέθους L με αλφάβητο το $\{0,1\}$. Ενώ το δεξί σκέλος αποτελείται από ένα σχηματοχώρο Hamming μεγέθους L με αλφάβητο το $\{A,C,G,T\}$.

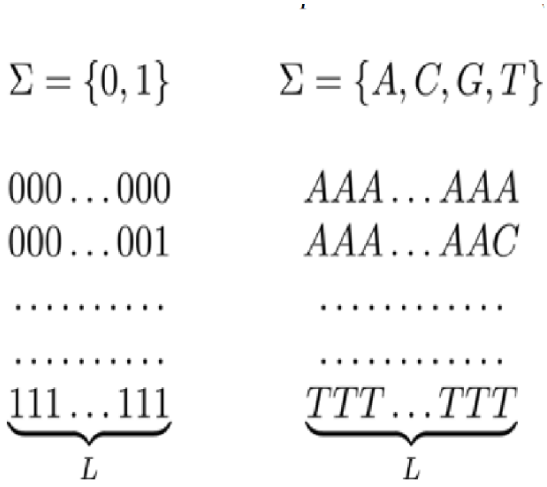


Fig. 2. Different Hamming shape spaces.

Ο Percus et. al [32] πρότεινε μία μέθοδο μέτρησης της έλξης γνωστή ως r-contiguous.

Ορισμός 4 *Έστω ένα στοιχείο* $e = (e_1, e_2, \cdots, e_L)$ *με* $e \in \Sigma^L$ *και έστω ένας ανιχνευτής* $d = (d_1, d_2, \cdots, d_L)$ *με* $d \in \Sigma^L$. *Σύμφωνα με τον r-contiguous κανόνα τα e και d ταιριάζουν αν υπάρχει φυσικός p ώστε* $e_i = d_i$ *για* $i = p, \cdots, p + r - 1$ *όπου* $p < L - r + 1$

Ο πιο παραπάνω ορισμός δηλώνει ότι δύο στοιχεία ταιριάζουν αν r συνεχόμενοι χαρακτήρες είναι ίδιοι.

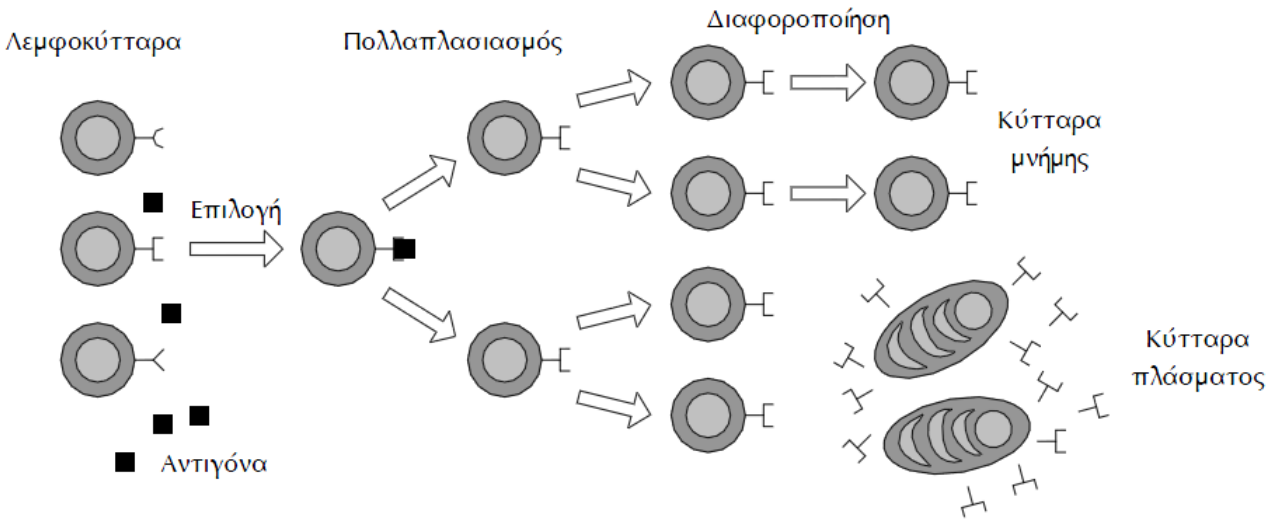
Η συντριπτική πλειοψηφία των ερευνών πάνω στον τομέα των Τεχητών Ανοσοποιητών Συστημάτων βασίζεται σε τρεις βασικές θεωρίες της ανοσολογίας: την αρχή της επιλογής των κλώνων, τη θεωρία ανοσοποιητικού δικτύου, και την αρχή της αρνητικής επιλογής. Παρακάτω παρουσιάζουμε βασικά χαρακτηριστικά των ερευνών αυτών καθώς και αλγορίθμων που έχουν αναπτυχθεί.

3.2.1 Ο Αλγόριθμος επιλογής των κλώνων

Ο αλγόριθμος επιλογής των κλώνων βασίστηκε στην αρχή της επιλογής των κλώνων. Η αρχή της επιλογής των κλώνων ενέπνευσε του θεμελιωτές των Τεχνητών Ανοσοποιητικών Συστημάτων στη δημιουργία ενός αλγόριθμου ο οποίος μπορεί να εφαρμοστεί στον τομέα της Αναγνώρισης Προτύπων. Ο αλγόριθμος βασίζεται στο γεγονός ότι μόνο τα λεμφοκύτταρα που αναγνωρίζουν καλύτερα τα αντιγόνα πολλαπλασιάζονται. Πιο συγκεκριμένα όταν τα αντισώματα Β-λεμφοκύτταρου αναγνωρίσουν ένα αντιγόνο τότε ενεργοποιούνται και αρχίζουν να κλωνοποιούνται καινούργια Β-λεμφοκύτταρα. Αρχικά αυτά παρουσιάζουν τα ίδια χαρακτηριστικά με τους γονείς τους, αλλά στη συνέχεια υπόκεινται σε μια διαδικασία υπερμετάλλαξης κατά την οποία παράγουν αντισώματα που είναι ικανά να αντιμετωπίσουν το συγκεκριμένο αντιγόνο.

Συμπερασματικά η αρχή της επιλογής των κλώνων (3.4) είναι η διαδικασία επιλογής του αριθμού των Β-λεμφοκυττάρων που θα παραχθούν και διέπεται από τους παρακάτω κανόνες.

1. Τα νέα κύτταρα είναι κλώνοι των γονέων τους, και υπόκεινται σε μεταλλάξεις
2. Τα λεμφοκύτταρα τα οποία έχουν αυτενεργούς υποδοχείς καταστρέφονται
3. Πολλαπλασιασμός και διαφοροποίηση σε επαφή ώριμων κυττάρων με αντιγόνα



Σχήμα 3.4: Η αρχή της επιλογής των κλώνων

Οι de Castro and Timmis [9] παρατήρησαν δύο σημαντικές ιδιότητες των Β-λεμφοκυττάρων που μπορούν να αξιοποιηθούν υπολογιστικά.

Η πρώτη παρατήρηση είναι ότι ο πολλαπλασιασμός των Β λεμφοκυττάρων είναι ανάλογος με την μέγεθος της έλξης. Δηλαδή όσο μεγαλύτερη έλξη παρουσιάζει ένα Β-λεμφοκύτταρο τόσο περισσότεροι κλώνοι παράγονται.

Η δεύτερη παρατήρηση είναι ότι οι μεταλλάξεις ενός αντισώματος είναι αντιστρόφως ανάλογη από την έλξη. Δηλαδή αν η έλξη μεταξύ ενός αντιγόνου και ενός αντισώματος είναι μεγάλη τότε οι μεταλλάξεις που θα προκληθούν στα Β-λεμφοκύτταρα θα είναι μικρές.

Αυτές οι παρατηρήσεις οδήγησαν τους de Castro and Von Zuben [10] στην ανάπτυξη του αλγορίθμου της αρχής της επιλογής των κλώνων που τον ονόμασαν CLONALG και έχει χρησιμοποιηθεί σε πληθώρα εφαρμογών.

Όταν αυτός ο αλγόριθμος εφαρμόζεται στην Αναγνώριση Προτύπων περιέχει ένα σύνολο S προτύπων που παίζουν τον ρόλο των αντιγόνων. Η λειτουργία του αλγορίθμου είναι να παράγει ένα σύνολο M από αντισώματα μνήμης, τα οποία ταιριάζουν με τα πρότυπα (αντιγόνα) του συνόλου S. Η σύγκριση αυτή υλοποιείται υπολογίζοντας τη μετρική του σχηματοχώρου Hamming. Ο αλγόριθμος είναι ο εξής :

Input: **S** = Σύνολο S προτύπων προς αναγνώριση.
n = Ο αριθμός χειρότερης περίπτωσης των στοιχείων a $\in S$ όπου θα μετακινηθούν.
Output: **M**= Σύνολο που περιέχει αντισώματα μνήμης που μπορούν να ταξινομήσουν καινούργια πρότυπα.
begin
Δημιούργησε ένα τυχαίο σύνολο A που περιέχει αντισώματα. Δηλαδή στοιχεία που μπορούν να αναγνωρίσουν τα αντιγόνα (πρότυπα)
foreach *στοιχείο του S* **do**
1. Υπολόγισε την έλξη κάθε στοιχείου του A με κάθε στοιχείο του S.
2. Δημιούργησε κλώνους των αντισωμάτων ενός υποσυνόλου του A με την μεγαλύτερη έλξη. Όσο μεγαλύτερη η έλξη τόσο περισσότεροι κλώνοι.
3. Μετάλλαξε στοιχεία των κλώνων αυτών ανάλογα με την έλξη τους. Πρόσθεσε αυτούς τους κλώνους στο σύνολο A, και αντέγραψε τα στοιχεία του A με τη μεγαλύτερη έλξη στο M
4. Αντικατέστησε n στοιχεία με την χαμηλότερη έλξη στο A με καινούργια αντισώματα τυχαίως επιλεγμένα.

end
end

Algorithm 1: Αλγόριθμος Επιλογής των Κλώνων

3.2.2 Αλγόριθμος Αρνητικής Επιλογής

Ο αλγόριθμος αρνητικής επιλογής αναπτύχθηκε από τους Forrest et al. [16] και αρχικά χρησιμοποιήθηκε για την αναγνώριση ιών σε υπολογιστές. Ο αλγόριθμος αυτός εμπνεύστηκε από τον κύριο μηχανισμό που πραγματοποιείται στον θύμο αδένα και παράγει Τ-λεμφοκύτταρα τα οποία έχουν την δυνατότητα να επιτελούν μία διάκριση του εαυτού με τον μη εαυτό. Αυτός ο μηχανισμός είναι ζωτικής σημασίας για τους σπονδυλωτούς οργανισμούς διότι αν καταρρεύσει οδηγεί στην εμφάνιση αυτοάνοσων ασθενειών. Στον θύμο αδένα τα Τ-λεμφοκύτταρα εκτίθενται σε πρωτεΐνες εαυτού. Τα Τ-λεμφοκύτταρα που αντιδρούν στις πρωτεΐνες αυτές καταστρέφονται ενώ αυτά που δεν αντιδρούν παραμένουν έτοιμα για δράση. Αυτά τα Τ-λεμφοκύτταρα αφού έχουν ωριμάσει στον θύμο αδένα και έχουν μηχανισμούς αναγνώρισης του εαυτού από τον μη εαυτό κυκλοφορούν στο σώμα του οργανισμού και τον προστατεύουν από εξωγενή αντιγόνα. Ο αλγόριθμος αυτός είναι πολύ σημαντικός για τη δημιουργία τεχνητών συστημάτων που θα μπορούν να κάνουν διάκριση μεταξύ μίας αποδεκτής κατάστασης και μίας μη αποδεκτής.

Αρχικά ο αλγόριθμος όπως αναπτύχθηκε από τον Forrest et al. [16] δημιουργεί ένα σύνολο από συμβολοσειρές S που αντιπροσωπεύουν τη κανονική κατάσταση του συστήματος. Δηλαδή το σύνολο S περιέχει στοιχεία που αντιπροσωπεύουν την φυσιολογική κατάσταση του συστήματος. Η έξοδος είναι ένα σύνολο D που τα στοιχεία του είναι ανιχνευτές τα οποία ανιχνεύουν στοιχεία που δεν ανήκουν στο S. Αυτοί οι ανιχνευτές έπειτα μπορούν να εφαρμοστούν σε διάφορα δεδομένα με σκοπό να τα ταξινομήσουν σε κανονικά

δεδομένα και ανώμαλα. Ο αλγόριθμος είναι ο παρακάτω:

Input: **S**= ένα σύνολο με στοιχεία που εκπροσωπούν την κανονική κατάσταση ενός συστήματος.
Output: **D**= ένα σύνολο από ανιχνευτές της ανώμαλης κατάστασης.
begin
  repeat
    1. Δημιούργησε τυχαία ανιχνευτές και πρόσθεσε τους σε ένα σύνολο P.
    2. Υπολόγισε το μέτρο έλξης κάθε στοιχείου του P με κάθε στοιχείου του συνόλου S.
    3. Αν τουλάχιστον ένα στοιχείο του S αναγνωριστεί από τους ανιχνευτές του P βάση ενός κατωφλιού θ τότε ο ανιχνευτής αυτός απορρίπτεται, αλλιώς προστίθεται στο D.
  until *κάποιο κριτήριο να ικανοποιηθεί*;
end

Algorithm 2: Αλγόριθμος Αρνητικής Επιλογής

Σε αυτή την υλοποίηση του αλγορίθμου οι ανιχνευτές παράγονται με τυχαίο τρόπο και έπειτα εξετάζεται αν ταιριάζουν με στοιχεία του εαυτού (ομαλής κατάστασης). Αν κάποιος ανιχνευτής ταιριάζει με κάποιο από τα στοιχεία εαυτού τότε απορρίπτεται. Αυτή η παραγωγή των ανιχνευτών επαναλαμβάνεται μέχρις ότου παραχθεί ένα ικανοποιητικός αριθμός ανιχνευτών. Ο αριθμός των ανιχνευτών που χρειαζόμαστε για να υπάρχει ένα ικανοποιητικό επίπεδο αξιοπιστίας διαφέρει. Μία προσέγγιση είναι μία πιθανοθεωρητική ανάλυση του αριθμού των ανιχνευτών, η οποία όμως αυξάνει εκθετικά με το πλήθος του συνόλου εαυτού. Έχουν προταθεί και άλλοι μέθοδοι που τρέχουν σε γραμμικό χρόνο Helman and Forrest [18].

3.2.3 Θεωρία Ανοσοποιητικού Δικτύου

Το 1974, ο Jerne [23] πρότεινε μία θεωρία πού βασίζεται στις ιδιότητες των κυττάρων και των μορίων του ανοσοποιητικού μας συστήματος να αναγνωρίζονται το ένα με το άλλο ακόμα και κατά την απουσία εξωγενών αντιγονικών παραγόντων. Η θεωρία ανοσοποιητικού δικτύου ερευνά την ικανότητα μάθησης και μνήμης των κυττάρων αυτών. Η θεωρία αυτή βασίζεται στο γεγονός ότι κάθε υποδοχέας ενός λεμφοκύτταρου μπορεί να αναγνωριστεί από συγκεκριμένο ρεπερτόριο άλλων υποδοχέων. Εν γένει μπορούμε να διακρίνουμε τα παρακάτω συστατικά που παίρνουν μέρος στην αναγνώριση των αντιγόνων από τα λεμφοκύτταρα.

- Επίτοπο : είναι ο υποδοχέας που έχει το αντιγόνο και μπορεί να προσδεθεί με το αντίσωμα
- Παράτοπο : είναι ο υποδοχέας ενός αντισώματος που μπορεί να προσδεθεί με ένα επίτοπο
- Ιδιότυπο: είναι το σύνολο όλων των επιτόπων

Η θεωρία του Ανοσοποιητικού Δικτύου βασίζεται στο γεγονός ότι το ανοσοποιητικό σύστημα περιέχει ένα σύνολο από ιδιότυπα τα οποία συνδέουν μεταξύ τους ένα σύνολο από Β-λεμφοκύτταρα. Τα κύτταρα

αυτά αλληλεπιδρούν μεταξύ τους και αυτό έχει ως αποτέλεσμα την σταθεροποίηση του δικτύου. Δύο Β-λεμφοκύτταρα παρουσιάζουν μία σύνδεση μεταξύ τους αν και μόνο αν η έλξη τους ξεπερνά ένα κατώφλι. Στην περίπτωση που ένα αντιγόνο αναγνωριστεί από ένα αντίσωμα αυτό συνεπάγεται την ενεργοποίηση του ανοσοποιητικού συστήματος και τον πολλαπλασιασμό ανοσοποιητικών κυττάρων.

Στο Τεχνητό Ανοσοποιητικό Δίκτυο ο πληθυσμός των Β-λεμφοκυττάρων αποτελείται από δύο υποπληθυσμούς: τον αρχικό πληθυσμό και τον κλωνοποιημένο. Ο αρχικός πληθυσμός παράγεται από ένα υποσύνολο του συνόλου εκπαίδευσης για να δημιουργήσει το δίκτυο. Τα υπόλοιπα στοιχεία του συνόλου εκπαίδευσης χρησιμοποιούνται για την αναπαράσταση των αντιγόνων. Αν η έλξη είναι μεγάλη τότε τα Β-λεμφοκύτταρα κλωνοποιούνται και μεταλλάσσονται. Η μετάλλαξη αυτή δημιουργεί ένα διαφοροποιημένο σύνολο αντισωμάτων. Από την άλλη πλευρά όταν ένα Β-λεμφοκύτταρο παραχθεί γίνεται μία προσπάθεια για την προσθήκη του στο Δίκτυο. Στην περίπτωση που αυτό δεν μπορεί να προστεθεί τότε αφαιρείται από τον πληθυσμό.

Ο αλγόριθμος που εμπνεύστηκε από την θεωρία του Jerne αναπτύχθηκε από τους de Castro and von Zuben [7] [6]. Ο αλγόριθμος αυτός ονομάστηκε aiNet και είναι παρόμοιος με τον αλγόριθμο επιλογής των κλώνων. Η διαφορά του aiNet από τον CLONALG έγκειται στο γεγονός ότι στον aiNet υπάρχουν αλληλεπιδράσεις μεταξύ του πληθυσμού των ανιχνευτών. Ο αλγόριθμος παρουσιάζεται ποιο κάτω :

Input: **G**= ένα σύνολο με πρότυπα προς αναγνώριση
N= ένα σύνολο ανιχνευτών
n=αριθμός των αντισωμάτων καλύτερης περίπτωσης
Output: **M**= ένα σύνολο από ανιχνευτές που αναγνωρίζουν τα πρότυπα που περιέχει το σύνολο G
begin
Δημιούργησε έναν αρχικό σύνολο Β με τυχαία στοιχεία **foreach** *πρότυπο* $\in G$ **do**
1. Υπολόγισε την έλξη κάθε στοιχείου του Β με κάθε στοιχείο του Ν των ανιχνευτών
2. Διάλεξε n στοιχεία του Β με την καλύτερη έλξη
3. Κλωνοποίησε και μετάλλαξε κάθε στοιχείο από τα n ανάλογα με το μέγεθος της έλξης τους
4. Πρόσθεσε αυτά τα n στοιχεία στον Μ
5. Διέγραψε δυναμικά τα χειρότερα στοιχεία του Μ ως προς την έλξη
6. Δημιούργησε b τυχαία στοιχεία και πρόσθεσε τα στο σύνολο Β

end
end

Algorithm 3: Αλγόριθμος Ανοσοποιητικού Δικτύου

Κεφάλαιο 4

Αλγόριθμος AIRS

4.1 Εισαγωγή

Ο αλγόριθμος AIRS (Artificial Immune Recognition System) είναι ένας αλγόριθμος ταξινόμησης που αναπτύχθηκε από τον Andrews Watkins (2001) [41] και υλοποιήθηκε με τεχνικές εμπνευσμένες από τα Τεχνητά Ανοσοποιητικά Συστήματα. Ο αλγόριθμος αυτός δεν προσπαθεί να μοντελοποιήσει κάποιο συγκεκριμένο μηχανισμό του ανοσοποιητικού συστήματος, αλλά δανείζεται ορισμένα χαρακτηριστικά του. Πιο συγκεκριμένα δανείστηκε ορισμένα συστατικά της θεωρίας της επιλογής των κλώνων και της θεωρίας του ανοσοποιητικού δικτύου. Από την θεωρία επιλογής των κλώνων και πιο συγκεκριμένα από τον Clonalg [10] χρησιμοποιεί το χαρακτηριστικό του συνόλου των κυττάρων μνήμης, που αποτελεί μία αναπαράσταση του περιβάλλοντος μάθησης, καθώς και την ιδιότητα των κυττάρων για ωρίμανση και υπερμετάλλαξη. Από την άλλη μεριά, από τη θεωρία ανοσοποιητικού δικτύου και τον αλγόριθμο AiNet [7] χρησιμοποιεί τον μηχανισμό ελέγχου του πληθυσμού των κυττάρων καθώς και την έννοια των τεχνητών σφαιρών αναγνώρισης (Artificial Recognition Ball). Τα κύτταρα ARB είναι ένα σύνολο Β-λεμφοκυττάρων που παρουσιάζουν τα ίδια χαρακτηριστικά μεταξύ τους. Τέλος, το κατώφλι διέγερσης (Affinity Threshold) είναι μία έννοια εμπνευσμένη από τον AiNet.

Στο [39] παρουσιάζονται τα ακόλουθα επιθυμητά χαρακτηριστικά που χαρακτηρίζουν τον αλγόριθμο AIRS.

- **Αυτορύθμιση (Self-regulation)**
  Ο αλγόριθμος έχει την ικανότητα της αυτορύθμισης, δηλαδή δεν χρειάζεται η επιλογή κάποιας αρχιτεκτονικής αλλά μαθαίνει την κατάλληλη αρχιτεκτονική του προβλήματος μέσω της διαδικασίας της εκπαίδευσης.

- **Επίδοση (Performance)**
  Η εφαρμογή του αλγορίθμου σε προβλήματα ταξινόμησης που πραγματοποιήθηκαν στο University of California, Irvine [4] έδειξαν ότι ο AIRS μπορεί να ανταγωνιστεί πολλούς άλλους αλγόριθμους ταξινόμησης. Τα αποτελέσματα δείχνουν ότι η ακρίβεια ταξινόμησης του AIRS είναι ανάμεσα στις πέντε καλύτερες ενώ στο [39] παρατηρούμε ότι σε ορισμένα σύνολα ο AIRS παρουσιάζει την μεγαλύτερη ακρίβεια.

- **Γενίκευση (Generalization)**
  Ο AIRS πραγματοποιεί μία διαδικασία γενίκευσης, μέσω της μεθόδου της μείωσης δεδομένων (Data Reduction). Αυτό σημαίνει ότι ο αλγόριθμος χρησιμοποιεί μικρότερο πλήθος δεδομένων [40] για την παραγωγή καλύτερων αποτελεσμάτων από άλλους αλγορίθμους.

► **Σταθερότητα παραμέτρων (Parameter Stability)**
Ο αλγόριθμος έχει τη δυνατότητα ρύθμισης των παραμέτρων έτσι ώστε να πετύχει καλύτερη ακρίβεια σε διαφορετικά δεδομένα.

Μερικές έννοιες που χρησιμοποιείται ο AIRS είναι οι ακόλουθες:

► **Έλξη (Affinity)**
Η έλξη περιγράφει το βαθμό ομοιότητας μεταξύ ενός αντιγόνου και ενός κυττάρου ανίχνευσης. Η μετρική που χρησιμοποιεί για την έλξη είναι η Ευκλείδια μετρική.

► **Ωρίμανση της έλξης (Affinity maturation)**
Είναι η ικανότητα του συστήματος να προσαρμόζεται και να ωριμάζει.

► **Κλωνοποίηση**
Η κλωνοποίηση είναι η διαδικασία παραγωγής των κλώνων.

► **Υπερμετάλλαξη(Hypermutation)**
Είναι η διαδικασία κατά την οποία τα κλωνοποιημένα κύτταρα υφίστανται μετάλλαξη, δηλαδή αλλαγή του γενετικού τους υλικού. Ο βαθμός υπερμετάλλαξης καθορίζεται από τον βαθμό έλξης του αντιγόνου από ένα κύτταρο ανιχνευτή.

Οι κλώνοι που δημιουργούνται με την διαδικασία της Υπερμετάλλαξης παρουσιάζουν διαφορετικά χαρακτηριστικά από τους γονείς τους. Με αυτόν τον τρόπο εφαρμόζεται μια εξελικτική διαδικασία επιλογής του ισχυρότερου, δηλαδή του κυττάρου ανιχνευτή, που παρουσιάζει μεγαλύτερη έλξη ως προς κάποιο αντιγόνο. Τέλος, το σύστημα του αλγορίθμου περιέχει ένα είδος μνήμης με την έννοια ότι αναγνωρίζει ένα αντιγόνο που το έχει ξανά-αντιμετωπίσει.

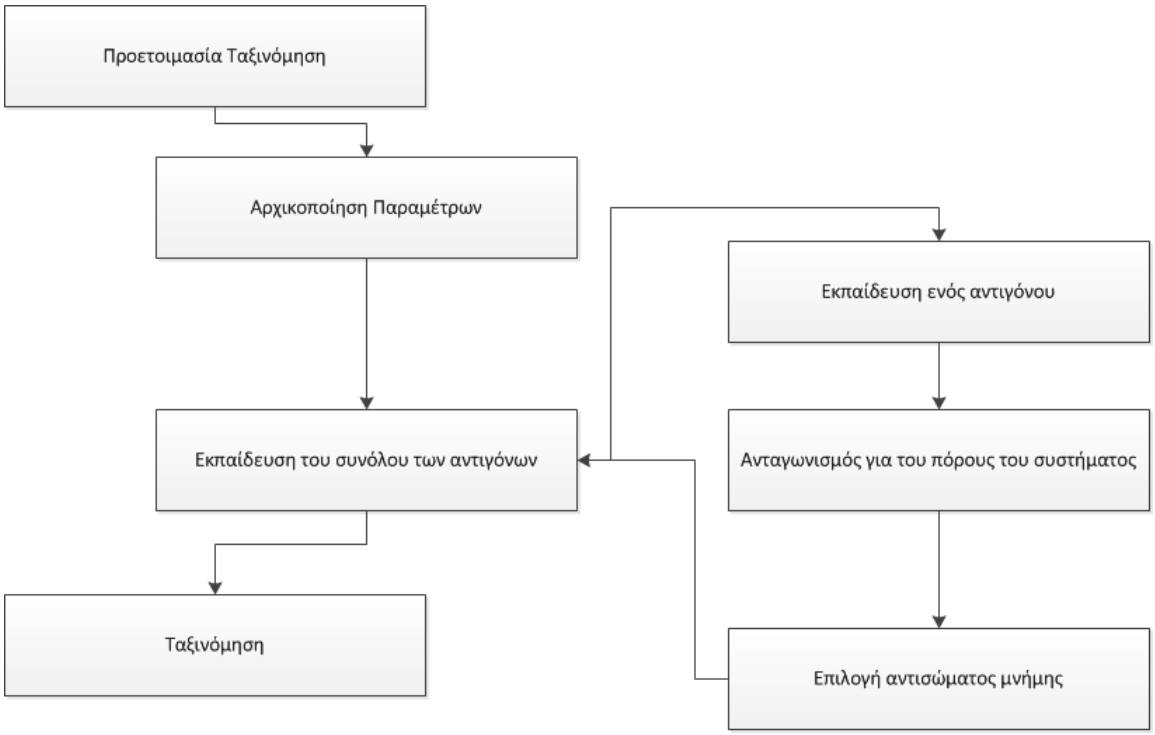
4.2 Περιγραφή Αλγορίθμου

Ο αλγόριθμος δέχεται ως είσοδο ένα σύνολο αντιγόνων, ένα προς ένα, το οποίο είναι το σύνολο εκπαίδευσης (Training Set) μίας κλάσης δεδομένων και παράγει ως έξοδο ένα σύνολο αντισωμάτων μνήμης που αναγνωρίζει τα στοιχεία (αντιγόνα) της κλάσης αυτής.

Ο αλγόριθμος AIRS αποτελείται από δύο σύνολα. Το σύνολο των Αντισωμάτων Μνήμης που περιέχει τα αντισώματα που είναι εκπαιδευμένα προς αναγνώριση των αντιγόνων. Καθώς επίσης το σύνολο των Διαθέσιμων Αντισωμάτων στο οποίο εκπαιδεύονται τα αντισώματα για την καλύτερη αναγνώριση των αντιγόνων. Σκοπός του αλγορίθμου είναι να παράγει ένα σύνολο Αντισωμάτων Μνήμης που να αναγνωρίζει όσο το δυνατόν καλύτερα τα αντιγόνα μίας κλάσης.

► **Αρχικοποίηση**
Κατά την αρχικοποίηση όλα τα στοιχειά του συνόλου δεδομένων που δέχεται ως είσοδο ο αλγόριθμος κανονικοποιούνται στο διάστημα $[0, 1]$. Με αυτόν τον τρόπο η Ευκλείδεια απόσταση μεταξύ δύο οποιονδήποτε στοιχείων του συνόλου δεδομένων να είναι στο διάστημα $[0, 1]$. Έστω D το σύνολο που περιέχει τα δεδομένα προς ταξινόμηση και $x, y \in D$ όπου $x, y \in \mathbb{R}$ τότε η απόσταση

$$d(x, y) = \|x - y\|_{norm} \leq 1 \qquad \forall x, y \in D$$



Σχήμα 4.1: Διάγραμμα διαδικασίας του αλγορίθμου AIRS

Το σύνολο $D \in \mathbb{M}_{N \times K}$ είναι το σύνολο των δεδομένων και αποτελείται από x κλάσεις μεγέθους $C \in \mathbb{M}_{\frac{N}{x} \times K}$. Το σύνολο εκπαίδευσης T (Training Set) είναι ένα υποσύνολο μίας τάξης του D και χρησιμοποιείται για την εκπαίδευση των στοιχείων αυτής της κλάσης. Γενικά ισχύει ότι

$$T_i \subseteq C_i \subseteq D \qquad i \in \{1, 2 \cdots x\}$$

Στη συνέχεια, ο αλγόριθμος υπολογίζει την μέση έλξη μεταξύ των στοιχείων του Training Set το Affinity Threshold. Το Affinity Threshold είναι η μέση τιμή των αποστάσεων μεταξύ των στοιχείων του Training Set και δίνεται από τον παρακάτω τύπο.

$$affinity\ threshold = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{affinity(ag_i, ag_j)}{\frac{n(n-1)}{2}}$$

όπου

$$affinity(x, y) = \|x - y\|_{norm}$$

Το τελευταίο στάδιο της αρχικοποίησης είναι η αρχικοποίηση του συνόλου Αντισωμάτων Μνήμης και του συνόλου Διαθέσιμων Αντισωμάτων.

► **Αρχικοποίηση Συνόλου Αντισωμάτων**
Το σύνολο των αντισωμάτων μνήμης για κάθε κλάση αρχικοποιείται με το τρέχον αντιγονικό πρότυπο από την ίδια κλάση προτύπων ή από ένα σύνολο αντιγονικών προτύπων από την ίδια κλάση προτύπων.

► **Αρχικοποίηση Συνόλου Διαθέσιμων Αντισωμάτων**
Το σύνολο διαθέσιμων αντισωμάτων (ARB pool) για κάθε κλάση αρχικοποιείται με ένα τυχαίο διά-

νυσμα χαρακτηριστικών στο θεωρούμενο χώρο σχημάτων.

► **Φάση Εκπαίδευσης**
Για κάθε κλάση προτύπων και κάθε αντιγονικό πρότυπο.

► **Προσδιορισμός συμβατού Αντισώματος Μνήμης**
Ο αλγόριθμος είναι, one -shot δηλαδή εξετάζει ένα στοιχείο (αντιγόνο) τη φορά. Το πρώτο βήμα είναι ο προσδιορισμός συμβατού αντισώματος μνήμης από το σύνολο των Αντισωμάτων Μνήμης. Έστω ag ένα αντιγόνο από το σύνολο Training Set. Στο βήμα αυτό προσδιορίζουμε το αντίσωμα μνήμης mc_{match} το οποίο παρουσιάζει το μεγαλύτερο βαθμό διέγερσης ως προς το τρέχον αντιγόνο ag.

$$mc_{match} = argmax_{mc \in MC_{ag.c}} stimulation(ag, mc)$$

όπου $stimulation(x, y)$ είναι η μεταξύ τους απόσταση δηλαδή

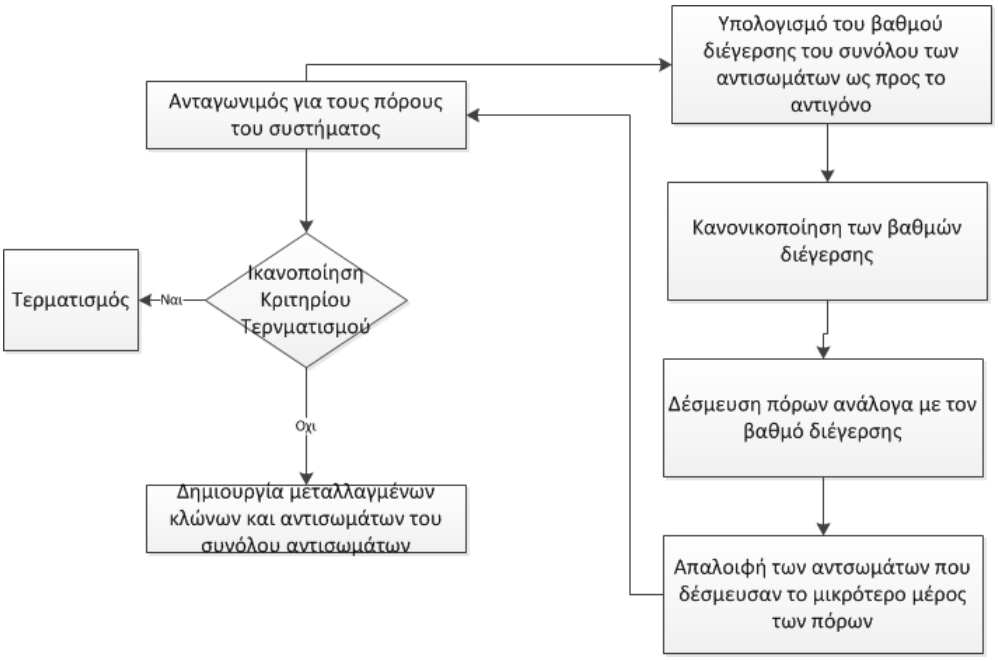
$$stimulation(x, y) = 1 - \|x - y\|_{norm}$$

Με άλλα λόγια το mc_{match} είναι εκείνο το αντίσωμα μνήμης που απέχει λιγότερο από το αντιγόνο ag.
Στην περίπτωση που το σύνολο των αντισωμάτων μνήμης αυτής της κλάσης προτύπων είναι κενό δηλαδή $MC_{ag.c} \equiv \emptyset$ τότε το $mc_{match} \leftarrow ag$ δηλαδή το mc_{match} είναι το ίδιο το αντιγονικό πρότυπο και έτσι το τοποθετούμε μέσα στο σύνολο αντισωμάτων μνήμης.

► **Παραγωγή αντισωμάτων**
Το αντίσωμα μνήμης mc_{match} που παρουσιάζει το μεγαλύτερο βαθμό διέγερσης προς το τρέχον αντιγονικό πρότυπο ag χρησιμοποιείται ως το αρχέτυπο για την παραγωγή ενός συνόλου από μεταλλαγμένες εκδοχές του αρχικού. Αυτά τα αντισώματα θα συμπεριληφθούν στο σύνολο των διαθέσιμων αντισωμάτων. Ο ρυθμός μετάλλαξης είναι αντιστρόφως ανάλογος του βαθμού διέγερσης προς το τρέχον αντιγονικό πρότυπο.

► **Διαδικασία εκπαίδευσης**
Η διαδικασία εκπαίδευσης επαναλαμβάνεται όσο ο μέσος βαθμός διέγερσης του συνόλου των διαθέσιμων αντισωμάτων είναι μικρότερος από κάποια προκαθορισμένη τιμή. Αυτό το βήμα του αλγόριθμου έχει ως στόχο να δημιουργήσει αντισώματα που να αναγνωρίζουν όσο καλύτερα το τρέχον αντίσωμα.

1. **Κατανομή πόρων**
   Για κάθε στοιχειο του συνόλου των διαθέσιμων αντισωμάτων δεσμεύεται ένα μέρος των συνολικών πόρων του συστήματος ανάλογα με το βαθμό διέγερσής του ως προς το τρέχον αντιγονικό πρότυπο.
2. **Καταστολή διαθέσιμων Αντισωμάτων**
   Απαλοιφή εκείνων των αντισωμάτων που δέσμευσαν το μικρότερο μέρος από τους συνολικού πόρους του συστήματος.
3. **Παραγωγή Μεταλλαγμένων Απογόνων**
   Το υποσύνολο των διαθέσιμων αντισωμάτων που έχουν εξασφαλίσει το μεγαλύτερο μέρος των πόρων του συστήματος έχουν μία επιπλέον ευκαιρία για την παραγωγή μεταλλαγμένων απογόνων.



Σχήμα 4.2: Διάγραμμα διαδικασίας κατανομής των πόρων του συστήματος

► **Προσδιορισμός Υποψήφιου Αντισώματος**
Ως υποψήφιο αντίσωμα μνήμης επιλέγεται εκείνο το διάνυσμα χαρακτηριστικών που παρουσιάζει το μεγαλύτερο βαθμό διέγερσης ως προς το τρέχον αντιγονικό πρότυπο. Το οποίο ονομάζουμε $mc_{candidate}$

► **Εισαγωγή αντισωμάτων μνήμης**
Σε αυτό το βήμα χρησιμοποιούμε το Affinity Threshold ως κριτήριο για την τοποθέτηση του $mc_{candidate}$ στο σύνολο των Αντισωμάτων Μνήμης. Το $mc_{candidate}$ προστίθεται στο σύνολο των αντισωμάτων μνήμης αν ο βαθμός διέγερσής του, ως προς το τρέχον αντιγονικό πρότυπο, είναι μεγαλύτερος από αυτόν του mc_{match}. Στην περίπτωση που αυτό ισχύει τότε αν το

$$affinity(mc_{candidate}, mc_{match}) < AT \cdot ATS$$

τότε το $mc_{candidate}$ τοποθετείται στο σύνολο αντισωμάτων μνήμης και αντικαθίσταται από το mc_{match} .

► **Ταξινόμηση**
Για την ταξινόμηση χρησιμοποιούμε τον αλγόριθμο των κ πλησιέστερων γειτόνων (knn-algorithm).

Κεφάλαιο 5

Προτεινόμενος Αλγόριθμος

5.1 Εισαγωγή

Η πρόταση της παρούσας εργασίας είναι μία μέθοδος ταξινόμησης εμπνευσμένη από τους Γενετικούς Αλγορίθμους και τον αλγόριθμο AIRS. Η υλοποίηση της μεθόδου Genetic AIRS βασίζεται στο γεγονός ότι έχει γίνει ελάχιστη προσπάθεια για την αξιοποίηση των Γενετικών Αλγορίθμων σε προβλήματα ταξινόμησης. Καθώς και στο ότι ο αλγόριθμος AIRS παρουσιάζει θετικά αποτελέσματα σε τέτοιου είδους προβλήματα. Ενώ τέλος, γίνεται μία προσπάθεια προσέγγισης του προβλήματος ταξινόμησης σφαιρικά (globally), αφού εκ φύσεως οι γενετικοί αλγόριθμοι υλοποιούνται σφαιρικά, σε αντίθεση με τη μέθοδο AIRS που εκτελείται τοπικά. Στόχος του αλγορίθμου αυτού είναι να παράγει λύσεις παρόμοιες ή καλύτερες από αυτές που παράγει ο αλγόριθμος AIRS, έτσι ώστε να αυξηθεί η ακρίβεια ταξινόμησης του αλγορίθμου σε διάφορα σύνολα δεδομένων.

Βασικό συστατικό του Genetic AIRS είναι οι ανιχνευτές που παίζουν το ρόλο των αντισωμάτων μνήμης. Οι ανιχνευτές εντοπίζουν τα αντικείμενα ενός συνόλου εκπαίδευσης. Με αυτό τον τρόπο, το σύνολο των ανιχνευτών δημιουργεί μία διαφορετική κατανομή του συνόλου εκπαίδευσης. Αυτό έχει ως αποτέλεσμα, την εξαγωγή συμπερασμάτων για την καλύτερη ταξινόμηση. Κάθε ανιχνευτής χαρακτηρίζεται από μία μπάλα. Έστω x ένας ανιχνευτής τότε ο ανιχνευτής αυτός ορίζει μία μπάλα που έχει την εξής μορφή

$$B_r p = \{d(x, p) \leq r\}$$

όπου p είναι κάποιο αντικείμενο στο σύνολο εκπαίδευσης και r είναι η ακτίνα της μπάλας. Η ακτίνα της μπάλας προσδιορίζει πόσο κοντά επιτρέπεται να είναι ο ανιχνευτής με το αντικείμενο του συνόλου εκπαίδευσης που προσεγγίζει. Επομένως, το πρόβλημα μας μοντελοποιείται ως ένα πρόβλημα βελτιστοποίησης της απόστασης του συνόλου των ανιχνευτών με τα αντικείμενα του συνόλου εκπαίδευσης. Η συνάρτηση προς βελτιστοποίηση είναι μια συνάρτηση της απόστασης των ανιχνευτών με τα αντικείμενα του συνόλου εκπαίδευσης.

5.2 Μαθηματική Διατύπωση

Έστω $Data$ το σύνολο που περιέχει τα δεδομένα προς ταξινόμηση με $Data \in \mathbb{M}_{N \times L}$ το οποίο αποτελείται από N αντικείμενα L διάστασης. Το σύνολο $Data$ χωρίζεται σε S κλάσεις μεγέθους $\frac{N}{S}$ οι οποίες είναι L διάστασης, δηλαδή $C \in \mathbb{M}_{\frac{N}{S} \times L}$. Το σύνολο των ανιχνευτών είναι το D το οποίο χρησιμοποιείται

για την ανίχνευση των αντικειμένων της κλάσης αυτής. Γενικά ισχύει ότι

$$C_i \subseteq Data \qquad i \in \{1, 2 \cdots S\}$$

και

$$\{C_1, C_2, \cdots, C_S\} = Data$$

Έστω $C_1, C_2, \cdots C_S$ είναι οι κλάσεις που προέρχονται από το σύνολο δεδομένων Data. Τα σύνολα εκπαίδευσης των κλάσεων αυτών είναι τα $\widehat{C_1}, \widehat{C_2}, \cdots \widehat{C_S}$ που είναι υποσύνολα των κλάσεων $C_1, C_2, \cdots C_S$ δηλαδή $\widehat{C_i} \subseteq C_i$ για $i = 1, 2, \cdots S$ που είναι της εξής μορφής:

$$\begin{aligned}
\widehat{C_1} &= \{\overrightarrow{x_1}^1, \overrightarrow{x_2}^1, \cdots, \overrightarrow{x_n}^1\} \\
\widehat{C_2} &= \{\overrightarrow{x_1}^2, \overrightarrow{x_2}^2, \cdots, \overrightarrow{x_n}^2\} \\
&\vdots \\
\widehat{C_S} &= \{\overrightarrow{x_1}^S, \overrightarrow{x_2}^S, \cdots, \overrightarrow{x_n}^S\}
\end{aligned}$$

με

$$\overrightarrow{x_j}^\kappa \in U_n = [0,1]^L, \quad \forall \kappa \in \{1, 2, \cdots, S\} \quad \forall j \in [1, 2, \cdots n]$$

Δηλαδή το $\overrightarrow{x_j}^\kappa$ είναι ένα διάνυσμα χαρακτηριστικών του j-οστού στοιχείου της κ κλάσης με τιμές που ανήκουν στο [0,1]. Στη συνέχεια ορίζουμε S σύνολα ανιχνευτών ένα για κάθε κλάση προτύπων.

$$\begin{aligned}
\widehat{D_1} &= \{\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1\} \\
\widehat{D_2} &= \{\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2\} \\
&\vdots \\
\widehat{D_S} &= \{\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S\}
\end{aligned}$$

με

$$\overrightarrow{d_j}^\kappa \in U_n = [0,1]^L, \quad \forall \kappa \in \{1, 2, \cdots, S\} \quad \forall j \in [1, 2, \cdots m]$$

Ενώ το $m \leq n$ δηλαδή το σύνολο των ανιχνευτών έχει λιγότερα χαρακτηριστικά L διάστασης από το σύνολο εκπαίδευσης.

Κάθε ανιχνευτής στο σύνολο $\widehat{D_i}$ με $i = 1, 2, \cdots, m$ ορίζει μία μπάλα ακτίνας $emax$ η οποία ορίζεται από τον χρήστη. Η ακτίνα αυτή ορίζει το πόσο κοντά θα βρίσκεται ο ανιχνευτής από το στοιχείο που ανιχνεύει. Αν ένας ανιχνευτής βρεθεί $emax$ κοντά σε ένα στοιχείο, τότε λέμε ότι ο ανιχνευτής προσδιορίζει αρκετά καλά το στοιχείο αυτό και έτσι αφαιρείται από το σύνολο των ανιχνευτών. Το πρόβλημα της βελτιστοποίησης έγκειται στην εύρεση ενός συνόλου ανιχνευτών έτσι ώστε να ελαχιστοποιείται η συνολική απόσταση των ανιχνευτών από τα αντικείμενα του συνόλου εκπαίδευσης. Συνεπώς, θα πρέπει να υλοποιήσουμε μία συνάρτηση f η οποία να αναπαριστά τη συνολική απόσταση των ανιχνευτών από τα στοιχεία του συνόλου εκπαίδευσης. Πιο αυστηρά το πρόβλημα βελτιστοποίησης είναι το εξής:

Πρώτη Προσέγγιση

$$minimize \qquad f_\kappa(\overrightarrow{d_1}^\kappa, \overrightarrow{d_2}^\kappa, \cdots, \overrightarrow{d_m}^\kappa) = \sum_{i=1}^{n} {\xi_i}^\kappa$$

Έτσι ώστε

$${\xi_i}^\kappa = \min_{j \in [m]} \{{\phi_{ij}}^\kappa\}, \quad \forall i \in [n]$$

με

$${\phi_{ij}}^\kappa = \max\{0, \|\overrightarrow{x_i}^\kappa - \overrightarrow{d_j}^\kappa\|_{norm} - emax\}$$

$$\overrightarrow{d_j}^\kappa \in U_n = [0,1]^L, \ \ \forall j \in [m]$$

και

$$\kappa \in [1, 2, \cdots, S] \tag{5.1}$$

Το κ είναι η κ-στη κλάση, άρα για τις S κλάσεις θα έχουμε S προβλήματα βελτιστοποίησης. Συνεπώς θα έχουμε τα εξής S προβλήματα τα οποία υλοποιούνται παράλληλα:

$$
\begin{aligned}
&minimize \quad f_1(\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1) = \sum_{i=1}^{n} {\xi_i}^1 \\
&minimize \quad f_2(\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2) = \sum_{i=1}^{n} {\xi_i}^2 \\
&\vdots \\
&minimize \quad f_S(\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S) = \sum_{i=1}^{n} {\xi_i}^S
\end{aligned}
$$

Έτσι ώστε

$$
\begin{aligned}
&{\xi_i}^1 = \min_{j \in [m]} \{{\phi_{ij}}^1\}, \quad \forall i \in [n] \\
&{\xi_i}^2 = \min_{j \in [m]} \{{\phi_{ij}}^2\}, \quad \forall i \in [n] \\
&\vdots \\
&{\xi_i}^S = \min_{j \in [m]} \{{\phi_{ij}}^S\}, \quad \forall i \in [n]
\end{aligned}
$$

με

$$
\begin{aligned}
&{\phi_{ij}}^1 = \max\{0, \|\overrightarrow{x_i}^1 - \overrightarrow{d_j}^1\|_{norm} - emax\} \\
&{\phi_{ij}}^2 = \max\{0, \|\overrightarrow{x_i}^2 - \overrightarrow{d_j}^2\|_{norm} - emax\} \\
&\vdots \\
&{\phi_{ij}}^S = \max\{0, \|\overrightarrow{x_i}^S - \overrightarrow{d_j}^S\|_{norm} - emax\} \\
&\overrightarrow{d_j}^{\kappa} \in U_n = [0,1]^L, \ \ \forall j \in [m]
\end{aligned}
$$

και

$$
\kappa \in [1, 2, \cdots, S]
$$

Δεδομένου $\overrightarrow{d}, \overrightarrow{x} \in U_n = [0,1]^L$ η κανονικοποιημένη Ευκλείδεια απόσταση δίνεται από τον παρακάτω τύπο.

$$
\|\overrightarrow{d} - \overrightarrow{x}\|_{norm} = \frac{1}{\sqrt{L}} \|\overrightarrow{d} - \overrightarrow{x}\|_{norm}
$$

Έτσι οι συναρτήσεις $f_1, f_2, \cdots, f_s$ δέχονται ως είσοδο τους

$$\widehat{D_1} = \{\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1\}$$
$$\widehat{D_2} = \{\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2\}$$
$$\vdots$$
$$\widehat{D_S} = \{\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S\}$$

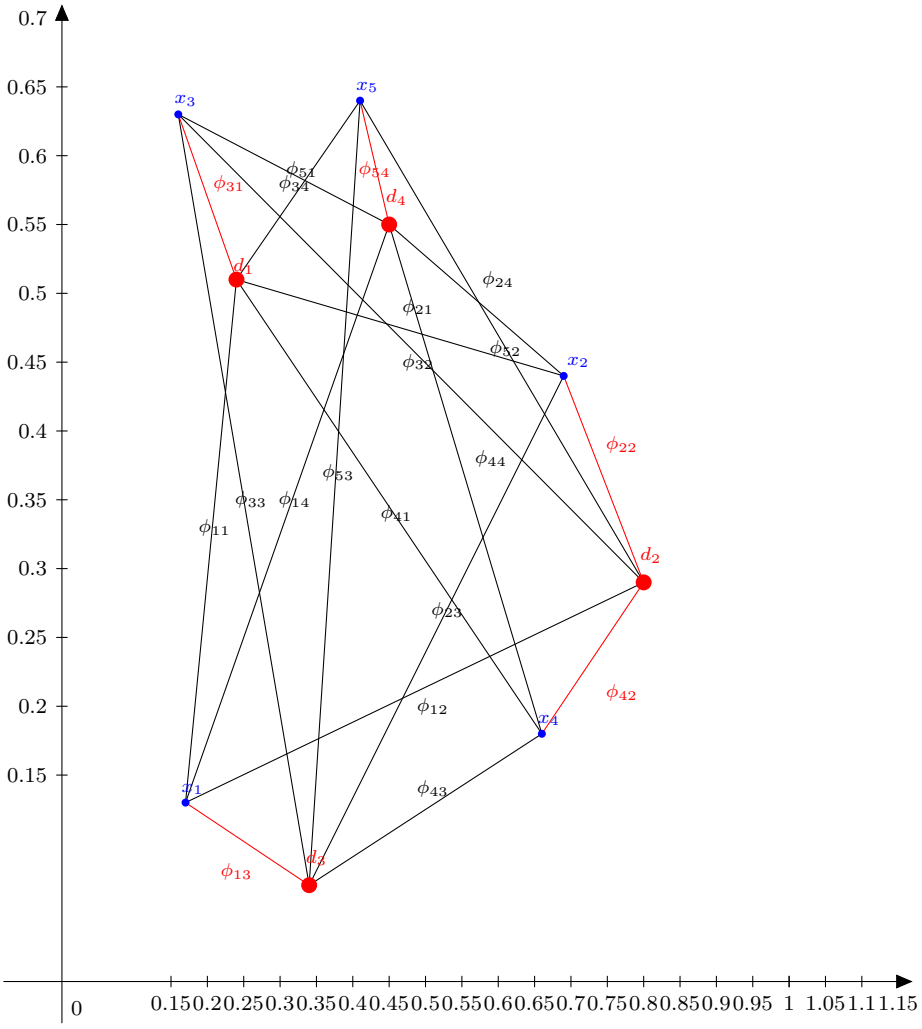
τυχαίους ανιχνευτές έτσι ώστε να αρχικοποιήσουν το πρόβλημα και παράγουν ως έξοδο του εκπαιδευμένους ανιχνευτές

$$\widehat{D_1^*} = \{\overrightarrow{d_1}^{1^*}, \overrightarrow{d_2}^{1^*}, \cdots, \overrightarrow{d_m}^{1^*}\}$$
$$\widehat{D_2^*} = \{\overrightarrow{d_1}^{2^*}, \overrightarrow{d_2}^{2^*}, \cdots, \overrightarrow{d_m}^{2^*}\}$$
$$\vdots$$
$$\widehat{D_S^*} = \{\overrightarrow{d_1}^{S^*}, \overrightarrow{d_2}^{S^*}, \cdots, \overrightarrow{d_m}^{S^*}\}$$

Το σύνολο των ανιχνευτών αυτών παρουσιάζει διαφορετική κατανομή και καλύτερα χαρακτηριστικά από τα αντικείμενα των κλάσεων $\widehat{C} = \{\widehat{C_1}, \widehat{C_2}, \cdots, \widehat{C_S}\}$ και αυτό έχει ως αποτέλεσμα να επιτυγχάνουν καλύτερη ακρίβεια ταξινόμησης.

Παράδειγμα

–



Σχήμα 5.1: Παράδειγμα

Στο σχήμα (5.1) αποτυπώνεται η παραπάνω διαδικασία στον χώρο των δύο διαστάσεων.

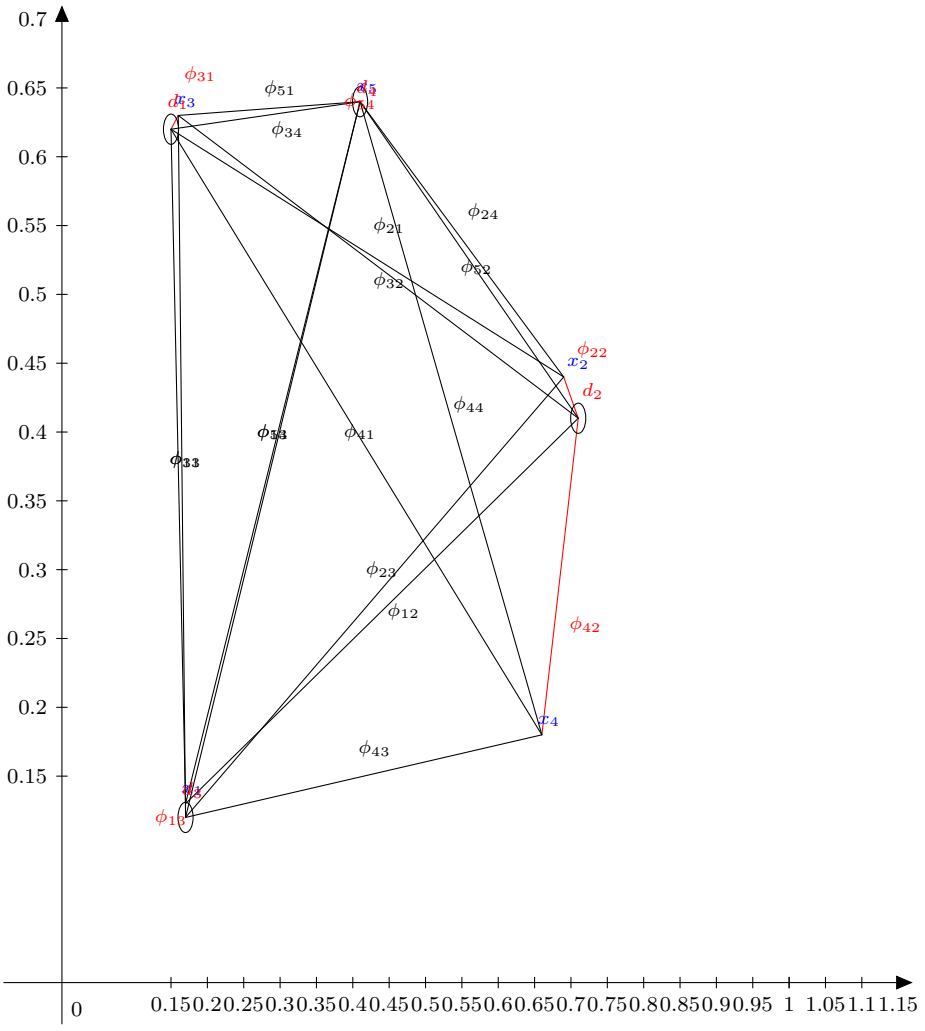
Τα $(x_1, x_2, x_3, x_4, x_5)$ είναι τα δεδομένα του συνόλου εκπαίδευσης ενώ τα (d_1, d_2, d_3, d_4) είναι οι ανιχνευτές που προσεγγίζουν τα σημεία αυτά. Η διαδικασία υπολογίζει όλες τις αποστάσεις μεταξύ των x_i και d_j που συμβολίζουμε ϕ_{ij}. Από αυτές τις αποστάσεις κρατάμε τις μικρότερες για κάθε i. Αυτές οι αποστάσεις

είναι η μεταβλητή

$$\xi_i = \min_{j \in [m]} \{\phi_{ij}\}, \quad \forall i \in [n]$$

Το πρόβλημα βελτιστοποίησης προσπαθεί να ελαχιστοποιήσει τις αποστάσεις αυτές έτσι ώστε να γίνουν οι μικρότερες δυνατές. Η ακτίνα emax είναι η ακτίνα των ανιχνευτών που ορίζει πόσο κοντά πρέπει να είναι ο ανιχνευτής για την επίτευξη της βέλτιστης ανίχνευσης. Στο παρακάτω σχήμα παρουσιάζεται μια ιδανική εξέλιξη της διαδικασίας βελτιστοποίησης.

–



Σχήμα 5.2: Παράδειγμα με ακτίνα

Δεύτερη Προσέγγιση

Η δεύτερη προσέγγιση είναι απολύτως ισοδύναμη με την πρώτη. Το μόνο που διαφοροποιείται, είναι η αντικειμενική συνάρτηση, η οποία υπολογίζει τις αποστάσεις ανάλογα με τον κοντινότερο γείτονα ανά σημείο.

$$minimize \qquad f_\kappa(\overrightarrow{d_1}^\kappa, \overrightarrow{d_2}^\kappa, \cdots, \overrightarrow{d_m}^\kappa) = \sum_{i=1}^{n} {\xi_i}^\kappa \tag{5.2}$$

$$\text{Έτσι ώστε} \qquad {\xi_i}^\kappa = \begin{cases} 0 & \|\overrightarrow{x_i}^\kappa - NN(D_1, \overrightarrow{x_i}^\kappa)\| \leq emax \\ 1 & otherwise \end{cases} \tag{5.3}$$

Με $NN(D_1, \overrightarrow{x_i}^\kappa)$ είναι η κοντινότερη απόσταση των x_i από το d_1.

5.3 Αλγόριθμος Genetic AIRS

Ο αλγόριθμος δέχεται ως είσοδο ένα σύνολο εκπαίδευσης (Training Set) μίας κλάσης δεδομένων και παράγει ως έξοδο ένα σύνολο ανιχνευτών το οποίο αναγνωρίζει τα στοιχεία της κλάσης αυτής. Σκοπός του αλγορίθμου είναι να παράγει ένα αριθμό ανιχνευτών για τις κλάσεις οι οποίοι θα αναγνωρίζουν όσο το δυνατόν καλύτερα τα στοιχεία που προέρχονται από τις κλάσεις αυτές.

Αρχικοποίηση

Κατά την αρχικοποίηση όλα τα στοιχειά του συνόλου δεδομένων που δέχεται ως είσοδο ο αλγόριθμος κανονικοποιούνται στο διάστημα $[0, 1]$. Δηλαδή τα στοιχεία του συνόλου $Data$ κανονικοποιούνται έτσι ώστε η Ευκλείδεια απόσταση μεταξύ δύο οποιονδήποτε στοιχείων του συνόλου δεδομένων να είναι στο διάστημα $[0, 1]$. Έστω $Data$ το σύνολο που περιέχει τα δεδομένα προς ταξινόμηση και $x, y \in Data$ όπου $x, y \in \mathbb{R}$ τότε η απόσταση

$$d(x, y) = \|x - y\|_{norm} \leq 1 \qquad \forall x, y \in Data$$

Η κανονικοποίηση του συνόλου $Data$ υλοποιείται με τον εξής τύπο

$$Normalized(Data) = \frac{Data_{ij} - \min(Data_j)}{\max(Data_j) - \min(Data_j)}$$

Το $\min(Data_j)$ και $\max(Data_j)$ είναι το ελάχιστο και μέγιστο στοιχείο j-στης στήλης του πίνακα $Data$ με $j = [1, 2, \cdots, L]$ με L οι στήλες του πίνακα. Η διαδικασία της κανονικοποίησης περιγράφεται στο (4).

Input: **Data** = Το σύνολο των δεδομένων
Output: **Normalized_ Data**= Το κανονικοποιημένο σύνολο δεδομένων στο διάστημα [0,1]
begin
  Min $\leftarrow$ min $(Data)$
  Max $\leftarrow$ max $(Data)$
  Collumns $\leftarrow Collumns(Data)$
  foreach $x_{ij} \in Data$ **do**
    $Normalized\_Data = \dfrac{x_{ij} - Min}{Max - Min}$
  end
end

Algorithm 4: Διαδικασία δημιουργίας αρχικού πληθυσμού

Αρχικοποίηση Συνόλου Ανιχνευτών

Στο βήμα αυτό αρχικοποιούμε το σύνολο των ανιχνευτών σε ένα διάστημα ϵ κοντά στη μέση τιμή κάθε χαρακτηριστικού της ανάλογης κλάσης, έτσι ώστε να εισάγουμε πληροφορία σχετικά με την κατανομή των δεδομένων της κλάσης αυτής. Συγκεκριμένα, το σύνολο των ανιχνευτών αρχικοποιείται για κάθε κλάση στο διάστημα

$$(mean(\overrightarrow{x_i^{\kappa}}) - \epsilon, mean(\overrightarrow{x_i^{\kappa}}) + \epsilon)$$

με $mean(\overrightarrow{x_i^{\kappa}})$ είναι η μέση τιμή του i χαρακτηριστικού της κ κλάσης δηλαδή

$$mean(\overrightarrow{x_i^{\kappa}}) = \frac{1}{L}\sum_{j=1}^{L} \overrightarrow{x_{i_j}^{\kappa}}$$

με L η διάσταση των χαρακτηριστικών και ϵ ένας μικρός αριθμός που εισάγεται από τον χρήστη. Η διαδικασία αρχικοποίησης του συνόλου των ανιχνευτών περιγράφεται στο 5.

Input:
Training Set = Το σύνολο των δεδομένων προς εκπαίδευση του Training Set
ϵ = Πόσο θα απέχει ο αρχικός πληθυσμός από τη μέση τιμή
Output:
Initial Population= Αρχικός πληθυσμός
begin
  $L \leftarrow |Training\_Set|$
  $mean\_value \leftarrow \sum_{j=1}^{L} \dfrac{x_{ij}}{L}$
  $a\_1 \leftarrow mean\_value + \epsilon$
  $a\_2 \leftarrow mean\_value - \epsilon$
  $rnd\_num \leftarrow random\_numbers \in [0, 1]$
  $Initial\_Population \leftarrow a\_1 + (b\_1 - a\_1) \cdot rnd\_num$
end

Algorithm 5: Διαδικασία δημιουργίας αρχικού πληθυσμού

Αντικειμενική Συνάρτηση

Η αντικειμενική συνάρτηση δέχεται ως είσοδο τρεις μεταβλητές, το σύνολο εκπαίδευσης κάποιας κλάσης $Training\_Set$, το σύνολο των χρωμοσωμάτων (ανιχνευτών) μίας γενιάς X καθώς και μια ακέραια τιμή $emax$ που είναι η ακτίνα των ανιχνευτών. Η αντικειμενική συνάρτηση υπολογίζει την ελάχιστη απόσταση κάθε στοιχείου του $Training\_Set$ από κάθε στοιχείο του X και στη συνέχεια αθροίζει όλες αυτές τις αποστάσεις σε μία μεταβλητή F. Αν η απόσταση ενός ή περισσότερων στοιχείων του $Training\_Set$ βρίσκεται $emax$ κοντά σε κάποιον ανιχνευτή του X, τότε αυτή η απόσταση μηδενίζεται. Αυτό έχει ως αποτέλεσμα να μην υπολογίζεται στο άθροισμα των αποστάσεων της F και έτσι η F να ελαττώνεται. Σκοπός της διαδικασίας του γενετικού αλγορίθμου είναι η ελαχιστοποίηση της F, δηλαδή η εύρεση ενός X^* έτσι ώστε η $F \longrightarrow 0$. Αν όμως $F \longrightarrow 0$ αυτό σημαίνει ότι όλα τα στοιχεία του συνόλου των ανιχνευτών X^* θα βρίσκονται $emax$ κοντά στο $Training\_Set$ δηλαδή $|X^* - Training\_Set| \longrightarrow 0$. Επειδή όμως, το πλήθος των στοιχείων του X^* είναι μικρότερο από το πλήθος των στοιχείων του $Training\_Set$ δηλαδή $|X^*| \leq |Training\_Set|$, αυτό έχει ως αποτέλεσμα το σύνολο X^* να παρουσιάζει διαφορετική κατανομή από το σύνολο $Training\_Set$. Συμπερασματικά, το X^* είναι ένα σύνολο ανιχνευτών το οποίο εκπαιδεύεται πάνω στο σύνολο $Training\_Set$ δηλαδή αντλεί πληροφορίες από το σύνολο αυτό και τελικά παράγεται ένα εκπαιδευμένο σύνολο ανιχνευτών κατάλληλο να εντοπίσει και να ταξινομήσει αντικείμενα κάποιας κλάσης. Η διαδικασία αυτή παρουσιάζεται στο (6).

Input:
Training _ Set = Το σύνολο των δεδομένων προς εκπαίδευση του Training Set
X = Ο σύνολο των χρωμοσωμάτων
emax= Ακέραιος
Output:
F= Το σύνολο των τιμών της αντικειμενικής συνάρτησης
begin
  $L \leftarrow |Training\_Set|$
  foreach $d_i \in Training\_Set$ **do**
    foreach $x_j \in X$ **do**
      $Distance \leftarrow \dfrac{\sqrt{(x_{j_1} - d_{i_1})^2 + (x_{j_2} - d_{i_2})^2 + \cdots + (x_{j_L} - d_{i_L})^2}}{\sqrt{L}}$
      if $Distance < emax$ **then**
        $Distance \leftarrow 0$
      end
    end
    $Minimum\_Dist \leftarrow \min(Distance)$
  end
  $F \leftarrow F + Minimum\_Dist$
end

Algorithm 6: Αντικειμενική Συνάρτηση

Εκπαίδευση

Στη φάση εκπαίδευσης χρησιμοποιούμε τον Γενετικό Αλγόριθμο για την εκπαίδευση των ανιχνευτών κάθε κλάσης. Χρησιμοποιούμε τον αρχικό πληθυσμό που έχουμε παράγει ο οποίος κωδικοποιεί μία υποψήφια λύση για την βελτιστοποίηση της αντικειμενικής συνάρτησης. Έστω m το πλήθος των ανιχνευτών που κυμαίνεται από το 80 % του Training Set ως το 100 % και emax η απόσταση μεταξύ του ανιχνευτή και

του αντικειμένου που ανιχνεύει. Οι ανιχνευτές εκπαιδεύονται με σκοπό το σύνολο των αποστάσεων τους από το Training Set να είναι το ελάχιστο. Ο αλγόριθμος δημιουργεί ακολουθίες νέων πληθυσμών σε κάθε βήμα. Χρησιμοποιεί το σύνολο των ανιχνευτών (χρωμοσωμάτων) της κάθε γενιάς για τη δημιουργία νέου πληθυσμού ως εξής:

1. Υπολογίζει τη διέγερση κάθε ανιχνευτή του πληθυσμού υπολογίζοντας την αντικειμενική συνάρτηση.
2. Επιλέγει μέλη του συνόλου των ανιχνευτών, τους γονείς, ανάλογα με το βαθμό διέγερσης τους.
3. Οι x καλύτεροι γονείς με την μεγαλύτερη διέγερση επιλέγονται και περνάνε στην επόμενη γενιά χωρίς μετάλλαξη και διασταύρωση.
4. Παραγωγή παιδιών από τους γονείς. Δημιουργείται ένα πλήθος παιδιών χρησιμοποιώντας τις διαδικασίες της μετάλλαξης και της διασταύρωσης. Δηλαδή, είτε προκαλώντας τυχαίες αλλαγές στον γονέα (μετάλλαξη) ή συνδυάζοντας στοιχεία των γονέων μεταξύ τους (διασταύρωση).
5. Παραγωγή της επόμενης γενιάς με τα παιδιά που δημιουργήθηκαν από τη διασταύρωση και τη μετάλλαξη.
6. Αντικατάσταση της παλιάς γενιάς με την νέα γενιά.

Input:
Training_ Set = Το σύνολο των δεδομένων προς εκπαίδευση
Output:
Detectors = Το σύνολο των ανιχνευτών
begin
  Δημιουργία αρχικού πληθυσμού
  while *Κριτήριο Τερματισμού* **do**
    ► Υπολογισμό της αντικειμενικής συνάρτησης σε κάθε χρωμόσωμα του πληθυσμού
    ► Διαδικασία Επιλογής
    ► Διαδικασία Ελιτισμού
    ► Διαδικασία Διασταύρωσης
    ► Διαδικασία Μετάλλαξης
    ► Δημιουργία Νέου Πληθυσμού
    ► Αντικατάσταση του παλιού πληθυσμού με τον νέο
  end
end

Algorithm 7: GA-AIRS

Διαδικασία Ελιτισμού

Η διαδικασία ελιτισμού είναι μία υπο-διαδικασία του Γενετικού Αλγορίθμου. Στη διαδικασία αυτή επιλέγουμε τους καλύτερους ανιχνευτές μιας γενιάς, δηλαδή αυτούς που παρουσιάζουν την καλύτερη τιμή στη συνάρτηση καταλληλότητας και τους τοποθετούμε αυτούσιους στην επόμενη γενιά χωρίς τη διαδικασία της διασταύρωσης και της μετάλλαξης.

Input:
F = Το σύνολο των τιμών της αντικειμενικής συνάρτησης
elite_ count= Ακέραιος για το πλήθος των τιμών
Output:
E = Το σύνολο των τιμών που επιλέχθηκαν
begin
  for *i=1:elite_ count* **do**
    $Maximum\_value \leftarrow \max(F)$
    $F \leftarrow F - Maximum\_value$
  end
end

Algorithm 8: Διαδικασία Ελιτισμού

Διαδικασία Ταξινόμησης

Τέλος, για την διαδικασία της ταξινόμησης χρησιμοποιούμε τον αλγόριθμο των k πλησιέστερων γειτόνων (k-nn algorithm). Ο αλγόριθμος αυτός δέχεται ως είσοδο το σύνολο των ανιχνευτών και το σύνολο των δεδομένων και ταξινομεί τα δεδομένα ανάλογα με τους k κοντινότερους γείτονες.

Input:
Detectors = $\{(x_1, c_1), (x_2, c_2), \cdots, (x_n, c_n)\}$ Το σύνολο των ανιχνευτών έπειτα από εκπαίδευση
Test_ Data=$\{y_1, y_2, \cdots, y_n\}$ Αντικείμενα άγνωστης κλάσης προς ταξινόμηση
Output: $\boldsymbol{D^K}$ = Οι κλάσεις των στοιχείων του Test_ Data
begin
  foreach $(x_i, c_i) \in Detectors$ **do**
    Υπολόγισε τις αποστάσεις $d(x_i, x)$
    $Order(d(x_i, x))$ Διέταξε τις αποστάσεις από τις μικρότερες στις μεγαλύτερες
    Επέλεξε τα k πιο κοντινά αντικείμενα x : ${D_x}^K$
    Ανέθεσε στο x την πιο συχνή κλάση του ${D_x}^K$
  end
end

Algorithm 9: Διαδικασία Ταξινόμηση k Πλησιέστερων Γειτόνων

5.4 Ομοιότητες και Διαφορές Με AIRS

Ομοιότητες

Συνοψίζοντας, οι δύο αλγόριθμοι παρουσιάζουν ορισμένα κοινά χαρακτηριστικά ως προς τις διαδικασίες και μεθοδολογίες που χρησιμοποιούν για την ταξινόμηση των χαρακτηριστικών σε διάφορες κατηγορίες. Βασίζονται και οι δύο στη θεωρία των Τεχνητών Ανοσοποιητικών Συστημάτων και για αυτό παρουσιάζουν ορισμένα κοινά χαρακτηριστικά.

Πιο συγκεκριμένα, καθώς συγκρίνουμε τα χαρακτηριστικά των δύο αλγορίθμων, παρατηρούμε αντιστοιχία των ανιχνευτών του Genetic Airs με τα αντισώματα μνήμης του Airs. Ένα ακόμη σημείο που οι δύο αλγόριθμοι παρουσιάζουν κοινά χαρακτηριστικά, έχει σχέση με τη μετρική που χρησιμοποιούμε για τον υπολογισμό της έλξης. Ο υπολογισμός της έλξης και στους δύο αλγορίθμους υπολογίζεται με την Ευκλείδεια μετρική, και είναι ο τρόπος μέτρησης της προσέγγισης ενός δεδομένου. Η μετάλλαξη είναι μία διαδικασία που παρουσιάζεται και στους δύο αλγορίθμους ως ένας τρόπος για την γενετική διαφοροποίηση των ανιχνευτών και των αντισωμάτων μνήμης. Στη συνέχεια, μία άλλη ομοιότητα παρουσιάζεται στη διαδικασία της επιλογής. Η συγκεκριμένη διαδικασία επιλέγει αντισώματα ή ανιχνευτές που παρουσιάζουν την μεγαλύτερη έλξη. Τέλος, οι δύο αλγόριθμοι εκτελούν αρχικοποίηση των δεδομένων εισόδου σε ένα διάστημα [0,1].

Διαφορές

Αν και οι δύο αλγόριθμοι παρουσιάζουν αρκετά κοινά χαρακτηριστικά, παρόλα αυτα δεν μπορούμε να παραβλέψουμε και τις ιδιαίτερες διαφοροποιήσεις τους. Πιο συγκεκριμένα, όσον αφορά το Data Reduction υπάρχουν έντονες διαφοροποιήσεις ανάμεσα στους δύο αλγορίθμους. Στον Airs, η διαδικασία Data Reduction υλοποιείται με ένα αυτοματοποιημένο τρόπο με στόχο να πετύχει μεγαλύτερη ακρίβεια ταξινόμησης. Με αυτό τον τρόπο ο Airs επιτυγχάνει καλύτερη ακρίβεια ταξινόμησης αλλά από την άλλη χρησιμοποιεί και περισσότερα δεδομένα για το σύνολο εκπαίδευσης. Ο Genetic Airs χρησιμοποιεί ένα σταθερό Data Reduction που ορίζεται από τον χρήστη. Έτσι μπορούμε να πετύχουμε (όπως παρουσιάζεται και στα παραδείγματα) μεγαλύτερη ακρίβεια ταξινόμησης με μικρότερο πλήθος ανιχνευτών.

Επιπλέον ένα άλλο σημείο διαφοροποίησης εντοπίζεται στο γεγονός ότι ο Airs εκτελείται τοπικά, χρησιμοποιώντας μία διαδικασία one-shot κατά την οποία εξετάζει ένα αντιγόνο την φορά και εκπαιδεύει σε σχέση με αυτό τα αντισώματα μνήμης. Ο Genetic Airs εκτελείται σφαιρικά δηλαδή εκπαιδεύει όλους τους ανιχνευτές παράλληλα ως προς το σύνολο προς ανίχνευση. Αυτή η σφαιρική αντιμετώπιση έχει ως αποτέλεσμα ο Genetic Airs να παρουσιάζει περισσότερο χρόνο εκτέλεσης από τον Airs.

Η παράμετρος Seed του αλγορίθμου Airs αποτελεί μία ακόμη διαφοροποίηση. Η παράμετρος αυτή έχει ως σκοπό την τοποθέτηση αντιγόνων στο σύνολο αντισωμάτων μνήμης έτσι ώστε να αρχικοποιηθεί. Μία παράμετρος του Genetic Airs που υλοποιεί μία αντίστοιχη διαδικασία είναι το Initial Population. Το Initial Population είναι μία συνάρτηση αρχικοποίησης που παράγει έναν αριθμό ανιχνευτών κοντά στη μέση τιμή του συνόλου δεδομένων.

Στη συνέχεια, μία παράμετρος που δεν παρουσιάζεται στον Airs, είναι η μεταβλητή Emax. Η ποσότητα αυτή είναι η ακτίνα του κάθε ανιχνευτή που καθορίζει πόσο κοντά μπορεί να βρίσκεται στο αντικείμενο προς ανίχνευση. Μπορούμε να πούμε ότι μία αντίστοιχη μεταβλητή που χρησιμοποιείται στον Airs είναι το Affinity Threshold Scalar αλλά δεν έχει την ίδια βαρύτητα και χρήση με αυτή του Emax.

Μία ακόμη διαδικασία που δεν υπάρχει στον αλγόριθμο Airs είναι αυτή της διασταύρωσης. Η διασταύρωση στον αλγόριθμο Genetic Airs είναι καθοριστικής σημασίας αφού αποτελεί μία επιπλέον διαδικασία, εκτός από αυτή της μετάλλαξης, διαφοροποιώντας το γενετικό υλικό του ανιχνευτή.

Τέλος, η διαδικασία κατανομής των πόρων του Airs δεν παρουσιάζεται στον Genetic Airs. Ο Genetic Airs υλοποιεί μια παρόμοια εσωτερική διαδικασία, αφού ως Γενετικός Αλγόριθμος βασίζεται και υλοποιεί

την αρχή της επικράτησης του ισχυρότερου.

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

6.1 Μεταβλητές του Genetic AIRS

Οι μεταβλητές του Genetic Airs καθώς και το πεδίο τιμών τους παρουσιάζονται στον παρακάτω πίνακα

Πίνακας 6.1: Μεταβλητές

Variables	Range
Detectors	[0.8,1]
Population Size	[20,200]
Elite Count	[1,10]
Crossover Fraction	[0,1]
Migration Fraction	[0,1]
k	[1,10]
emax	[0,1]
Crossover Function	Single Point Two Point Arithmetic Heuristic Intermediate
Selection Function	Stochastic Remainder Roulete Wheel Touranment

Επεξήγηση Μεταβλητών

- Detectors: Είναι το ποσοστό των δεδομένων που θα χρησιμοποιηθούν ως ανιχνευτές. Είναι το ποσοστό του Training Set το οποίο τελικά θα εκπαιδευτεί για την ανίχνευση των δεδομένων μας. Η μεταβλητή Detectors παίρνει τιμές από [0.8,1] δηλαδή από το 80 % του Training Set ως και το 100 % .
- Population Size: Το Population Size καθορίζει πόσες τιμές θα υπάρχουν σε κάθε γενιά εκτέλεσης του Γενετικού α Αλγορίθμου. Με μία μεγάλη τιμή του Population Size ο αλγόριθμος αναζητά λύσεις σε μεγαλύτερο χώρο αναζήτησης, αλλά από την άλλη πλευρά αυξάνεται ο χρόνος εκτέλεσης του αλγορίθμου. Θα χρησιμοποιήσουμε τιμές για το Population Size στο διάστημα [20-200]
- Individuals: Είναι ο πληθυσμός των πιθανών λύσεων.
- Elite Count: Είναι ο αριθμός των καλύτερων individuals που θα επιζήσουν και θα περάσουν στην επόμενη γενιά. Θα χρησιμοποιήσουμε τιμές για το Elite Count στο διάστημα [1-10]
- Crossover Fraction : Είναι το ποσοστό των individuals που παράγονται από την διαδικασία της διασταύρωσης. Θα χρησιμοποιήσουμε τιμές για το Crossover Fraction στο διάστημα [0-1].
- Mutation Fraction: Το Mutation Fraction είναι το ποσοστό των individuals που παράγονται από την διαδικασία της μετάλλαξης. Το Mutation Fraction εξαρτάται από το Crossover Fraction αλλά και από το Elite Count. Ο τύπος που μας δίνει το Mutation Fraction είναι ο ακόλουθος:

$$Mutation\_Fraction = Population\_Size - ((Crossover\_Fraction * Population\_Size) - Elite\_Count)$$

- Migration Fraction: Καθορίζει πόσα individuals θα ανταλλάσσονται μεταξύ των υποπληθυσμών. Με άλλα λόγια το Migration Fraction είναι το ποσοστό των individuals που θα μετακινηθούν στον υποπληθυσμό. Θα χρησιμοποιήσουμε τιμές για το Migration Fraction στο διάστημα [0-1] .
- Migration Interval: Καθορίζει σε πόσες γενιές θα γίνει η διαδικασία migration.
- k: Είναι η μεταβλητή που χρησιμοποιείται στον βοηθητικό αλγόριθμο k-Nearest Neighbor, ο οποίος λαμβάνει υπόψη τις αποστάσεις των k πλησιέστερων γειτόνων. Θα χρησιμοποιήσουμε τιμές για το k στο διάστημα [1,10] .
- emax: Είναι η μεταβλητή που καθορίζει πόσο κοντά θα είναι το training set με τα πραγματικά μας δεδομένα.Θα χρησιμοποιήσουμε τιμές για το emax στο διάστημα [0,1] .

6.2 Σύνολα Πειραματικών Δεδομένων

6.2.1 Iris Data

Τα Iris Data [17] είναι δεδομένα λουλουδιών που αποτελούνται από τρεις κλάσεις δεδομένων, την κλάση Setosa, την κλάση Versicolor και την κλάση Virginica. Αυτές οι κλάσεις έχουν 50 στοιχεία τεσσάρων διαστάσεων η κάθε μία. Πιο συγκεκριμένα, οι τρείς κλάσεις ανήκουν στο σύνολο M_{50x4}. Τα πειράματα γίνονται με την διαδικασία 10 fold-cross validation. Με τη μέθοδο αυτή χωρίζουμε τα δεδομένα μας για κάθε κλάση σε test-δεδομένα και σε training-δεδομένα. Τα test-δεδομένα είναι το 10 % των συνολικών δεδομένων της

κάθε κλάσης και τα training-δεδομένα είναι τα υπόλοιπα 90 % των δεδομένων μας. Συγκεκριμένα, για τα Iris Data από τα 50 δεδομένα της κάθε κλάσης έχουμε 5 δεδομένα για test και τα υπόλοιπα 45 δεδομένα για training. Στη συνέχεια πραγματοποιούμε knn Classification με Sample τα δεδομένα που δημιουργήθηκαν από τον αλγόριθμο Genetic Airs και Training τα test-δεδομένα που εξαιρέσαμε από το σύνολο. Η διαδικασία knn Classification πραγματοποιείται 10 φορές, δηλαδή μία για κάθε fold. Στο τέλος, υπολογίζουμε τον confusion-matrix που μας δείχνει πόσο επιτυχημένα ή όχι ο αλγόριθμος αναγνώρισε ότι ένα δεδομένο από το σύνολο των test-δεδομένων ανήκει στην κλάση από την οποία προήλθε.

Οι αρχικές τυχαίες μεταβλητές για την διεξαγωγή των πειραμάτων για το Iris Data είναι οι ακόλουθες

Πίνακας 6.2: Αρχικές Μεταβλητές Iris Data

PopulationSize	150
EliteCount	2
Crossover Fraction	0.9
Migration Interval	14
Migration Fraction	0.2
k	4
emax	0.1

- Στα παρακάτω πειράματα κρατάμε σταθερά όλα τα ορίσματα εκτός από ένα που το μεταβάλλουμε για να δούμε τη συμπεριφορά του αλγορίθμου.
- Αρχικά τα πειράματα γίνονται με τα Default ορίσματα του Γενετικού Αλγορίθμου ως προς τα Mutation Function, Selection Function, Crossover Function. Έπειτα από την επιλογή των κατάλληλων παραμέτρων διεξάγουμε πειράματα για τις υπόλοιπες μεταβλητές.
- Το Initial Population παράγεται με την Default συνάρτηση του γενετικού αλγόριθμου, παράγοντας έτσι τιμές στο διάστημα [0-1].
- Οι Detectors είναι το 80 % του training set
- Τα Test γίνονται με την διαδικασία 10-cross-validation.

6.2.2 Prima Indians Diabetes Data

Το Prima Indians Diabetes [17] είναι ένα σύνολο δεδομένων που αναπαριστά ένα δυαδικό πρόβλημα ταξινόμησης. Πιο συγκεκριμένα είναι ένα πρόβλημα καθορισμού αν ένας ασθενής παρουσιάζει διαβήτη (κλάση 1) ή όχι (κλάση 0).

Υπάρχουν 768 δεδομένα συνολικά με 8 χαρακτηριστικά το κάθε ένα. Τα 500 δεδομένα του συνόλου ανήκουν στην κλάση 1 ενώ τα 268 ανήκουν στην κλάση 0. Για την πραγματοποίηση των πειραμάτων πραγματοποιούμε knn Classification με Sample τα δεδομένα που δημιουργήθηκαν από τον αλγόριθμο Genetic Airs και Training τα test-δεδομένα που εξαιρέσαμε από το σύνολο. Η διαδικασία knn Classification πραγματοποιείται 10 φορές δηλαδή μία για κάθε fold. Στο τέλος, υπολογίζουμε τον confusion-matrix που μας δείχνει πόσο επιτυχημένα ή όχι ο αλγόριθμος αναγνώρισε ότι ένα δεδομένο από το σύνολο των test-δεδομένων ανήκει στην κλάση από την οποία προήλθε.

Οι αρχικές τυχαίες μεταβλητές για την διεξαγωγή των πειραμάτων για το Prima Indians Diabetes Data είναι οι ακόλουθες

- Στα παρακάτω πειράματα κρατάμε σταθερά όλα τα ορίσματα εκτός από ένα που το μεταβάλλουμε για να δούμε τη συμπεριφορά του αλγόριθμου.

Πίνακας 6.3: Αρχικές Μεταβλητές Prima Indians

PopulationSize	80
EliteCount	2
Crossover Fraction	1
Migration Interval	50
Migration Fraction	0.2
k	3
emax	0.1

- Αρχικά τα πειράματα γίνονται με τα Default ορίσματα του γενετικού αλγόριθμου ως προς τα Mutation Function Selection Function, Crossover Function.Έπειτα από την επιλογή των κατάλληλων παραμέτρων διεξάγουμε πειράματα για τις υπόλοιπες μεταβλητές.
- Το Initial Population παράγεται από μία συνάρτηση που παράγει τιμές ϵ κοντά στην μέση τιμή των δεδομένων κάθε χαρακτηριστικού όπου $\epsilon = 0.05$.
- Οι Detectors είναι το 100 % του training set
- Τα Test γίνονται με την διαδικασία 10-cross-validation.

6.2.3 Sonar Data

Το Sonar Data [17] αποτελείται από 208 δεδομένα που κάθε ένα έχει 60 χαρακτηριστικά. Τα 111 δεδομένα προέρχονται από σήματα μετάλλων που έχουν ληφθεί από διάφορες γωνίες και κάτω από διαφορετικές συνθήκες. Τα υπόλοιπα 97 δεδομένα από σήματα από πέτρες που έχουν ληφθεί με παρόμοιο τρόπο. Το σύνολο των δεδομένων περιέχει σήματα που λαμβάνονται από διαφορετικές γωνίες που κυμαίνονται από 90 μοίρες για τα μέταλλα και 180 για τις πέτρες. Κάθε δεδομένο είναι ένα σύνολο από 60 στοιχεία που οι τιμές τους κυμαίνονται στο διάστημα [0,1].

Για την διεξαγωγή των πειραμάτων τα Sonar Data έχουν χωριστεί σε δύο κλάσεις, την κλάση mines και την κλάση rocks. Η κλάση rocks αποτελείται από 100 δεδομένα με 60 χαρακτηριστικά το κάθε ένα. Ενώ η κλάση mines αποτελείται από 90 δεδομένα. Τα πειράματα υλοποιούνται με την διαδικασία 10-cross validation όπου τα δεδομένα χωρίζονται σε test δεδομένα και training δεδομένα. Συγκεκριμένα για την κλάση rocks θα έχουμε 10 δεδομένα test και 90 δεδομένα training set. Ενώ, για την κλάση mines θα έχουμε 81 δεδομένα για το training set και 9 δεδομένα για το test set. Στη συνέχεια, πραγματοποιούμε knn Classification με Sample τα δεδομένα που δημιουργήθηκαν από τον αλγόριθμο Genetic Airs και Training τα test-δεδομένα που εξαιρέσαμε από το σύνολο. Η διαδικασία knn Classification πραγματοποιείται 10 φορές δηλαδή μία για κάθε fold. Στο τέλος, υπολογίζουμε τον confusion-matrix που μας δείχνει πόσο επιτυχημένα ή όχι ο αλγόριθμος αναγνώρισε ότι ένα δεδομένο από το σύνολο των test-δεδομένων ανήκει στην κλάση από την οποία προήλθε.

Οι αρχικές τυχαίες μεταβλητές για την διεξαγωγή των πειραμάτων για το Sonar Data είναι οι ακόλουθες

- Στα παρακάτω πειράματα κρατάμε σταθερά όλα τα ορίσματα εκτός από ένα που το μεταβάλλουμε για να δούμε τη συμπεριφορά του αλγόριθμου.
- Αρχικά τα Test γίνονται με τα Default ορίσματα του γενετικού αλγόριθμου ως προς τα Mutation Function, Selection Function, Crossover Function. Έπειτα από την επιλογή των κατάλληλων παραμέτρων διεξάγουμε πειράματα για τις υπόλοιπες μεταβλητές.

Πίνακας 6.4: Αρχικές Μεταβλητές Sonar Data

PopulationSize	50
EliteCount	2
Crossover Fraction	1
Migration Interval	5
Migration Fraction	0.2
k	3
emax	0.1

- Το Initial Population παράγεται από μία συνάρτηση παράγει τιμές ϵ κοντά στην μέση τιμή των δεδομένων κάθε χαρακτηριστικού όπου $\epsilon = 0.05$.
- Οι Detectors είναι το 100 % του training set
- Τα Test γίνονται με την διαδικασία 10-cross-validation.

6.2.4 Music Feature Data

Το σύνολο δεδομένων αυτό αποτελείται από ένα σύνολο των 1000 τραγουδιών που προέρχονται από 10 κατηγορίες της δυτικής μουσικής. Πιο συγκεκριμένα περιέχει 100 τραγούδια των 30 δευτερολέπτων το καθένα από τις επόμενες 10 κατηγορίες δυτικής μουσικής.

Πίνακας 6.5: Κατηγορίες

Class ID	Label
1	Blues
2	Classical
3	Country
4	Disco
5	Hip-Hop
6	Jazz
7	Metal
8	Pop
9	Reggae
10	Rock

Κάθε στιγμιότυπο είναι ένα μουσικό αρχείο ποιότητας CD το οποίο περιέχει 44.100 16 bit δείγματα / δευτερόλεπτο, διάρκειας 30 δευτερολέπτων. Χρησιμοποιήθηκε το λογισμικό MARSYAS για τη μείωση της διάστασης του αρχικού χώρου προκειμένου από κάθε μουσικό αρχείο να εξαχθεί ένα 30-διάστατο πραγματικό διάνυσμα χαρακτηριστικών. Στη συνέχεια πραγματοποιούμε knn Classification με Sample τα δεδομένα που δημιουργήθηκαν από τον αλγόριθμο Airs και Training τα test-δεδομένα που εξαιρέσαμε από το σύνολο. Η διαδικασία knn Classification πραγματοποιείται 10 φορές δηλαδή μία για κάθε fold. Στο τέλος, υπολογίζουμε τον confusion-matrix που μας δείχνει πόσο επιτυχημένα ή όχι ο αλγόριθμος αναγνώρισε ότι ένα δεδομένο από το σύνολο των test-δεδομένων ανήκει στην κλάση από την οποία προήλθε. Έχουν διεξαχθεί 10 πειράματα για το συγκεκριμένο σύνολο δεδομένων Music Data. Έχουμε κάνει σύγκριση των ακόλουθων κατηγοριών:

Οι αρχικές τυχαίες μεταβλητές για την διεξαγωγή όλων των παραπάνω πειραμάτων είναι οι ακόλουθες

Πίνακας 6.6: Κλάσεις

C1vsC2
C1vsC2vsC3
C1vsC2vsC3vsC4
C1vsC2vsC3vsC4vsC5
C1vsC2vsC3vsC4vsC5vsC6
C1vsC2vsC3vsC4vsC5vsC6vsC7
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10

Πίνακας 6.7: Αρχικές Μεταβλητές Music Data

PopulationSize	80
EliteCount	5
Crossover Fraction	1
Migration Interval	50
Migration Fraction	0.2
k	3
emax	0.1

- Στα παρακάτω πειράματα κρατάμε σταθερά όλα τα ορίσματα εκτός από ένα που το μεταβάλλουμε για να δούμε τη συμπεριφορά του αλγόριθμου.
- Αρχικά τα Test γίνονται με τα Default ορίσματα του γενετικού αλγόριθμου ως προς τα Creation Function, Migration Function, Selection Function, Crossover Function. Έπειτα από την επιλογή των κατάλληλων παραμέτρων διεξάγουμε πειράματα για τις υπόλοιπες μεταβλητές.
- Το Initial Population παράγεται από μία συνάρτηση παράγει τιμές ϵ κοντά στην μέση τιμή των δεδομένων κάθε χαρακτηριστικού όπου $\epsilon = 0.05$.
- Οι Detectors είναι το 80 % του training set
- Τα Test γίνονται με την διαδικασία 10-cross-validation.

6.3 Μεταβλητή Elite Count

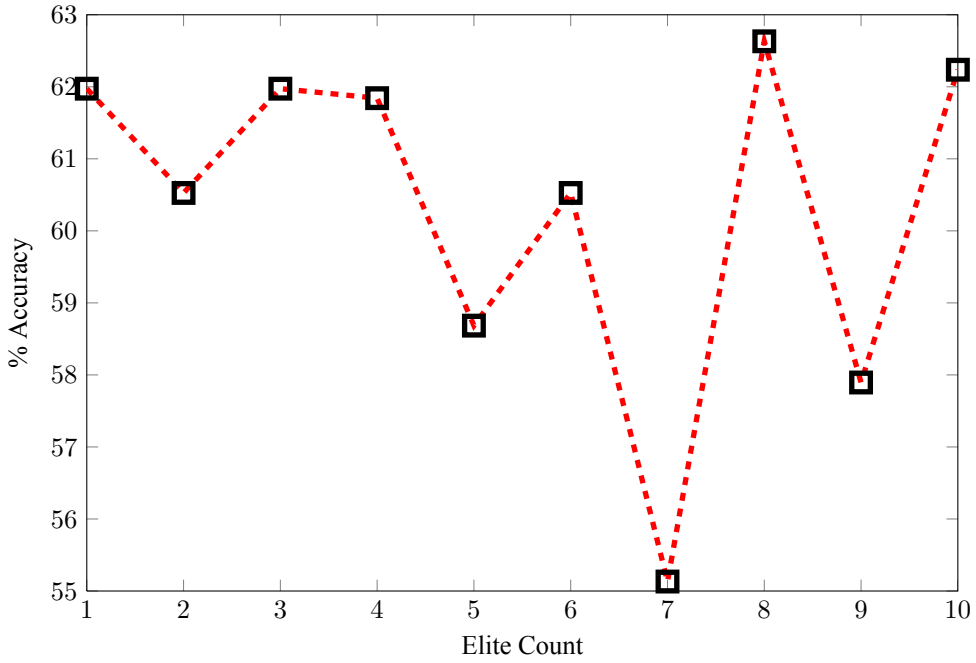
Η παράμετρος Elite Count είναι ο αριθμός των χρωμοσωμάτων που θα μεταφερθούν χωρίς καμία γενετική αλλαγή στην επόμενη γενιά. Τα χρωμοσώματα αυτά, παρουσιάζουν την καλύτερη τιμή στη συνάρτηση καταλληλότητας, από όλα τα άλλα χρωμοσώματα μιας συγκεκριμένης γενιάς. Μία μεγάλη τιμή στην παράμετρο αυτή μπορεί να προκαλέσει μια αναποτελεσματική αναζήτηση, διότι τα χρωμοσώματα που θα παράγονται θα είναι περισσότερα χωρίς καμία αλλαγή στη γενετική τους δομή. Γι' αυτό το λόγο έχουμε επιλέξει ένα διάστημα τιμών [1-10] για την παράμετρο αυτή που είναι μικρό, σε σχέση με το πλήθος των χρωμοσωμάτων που χρησιμοποιούμε σε κάθε σύνολο δεδομένων. Στο γράφημα (6.1) παρατηρούμε ότι η μέγιστη τιμή επιτυγχάνεται όταν το Elite Count είναι 8. Για το IrisData (6.2) η μέγιστη τιμή πετυχαίνεται για Elite Count 5, ενώ για το Sonar Data (6.3) είναι 8. Στα τρία αυτά σύνολα δεδομένων παρατηρούμε ότι όσο το Elite Count αυξάνεται τόσο η ταξινομική ακρίβεια μειώνεται.

Τέλος, για το σύνολο δεδομένων Music Data τα πειράματα έχουν πραγματοποιηθεί σε 10 διαφορετικές κλάσεις και τα αποτελέσματα παρουσιάζονται στο (6.4). Συγκεκριμένα οι μέγιστες τιμές για κάθε διαφορετική σύγκριση δίνονται στον πίνακα (6.8).

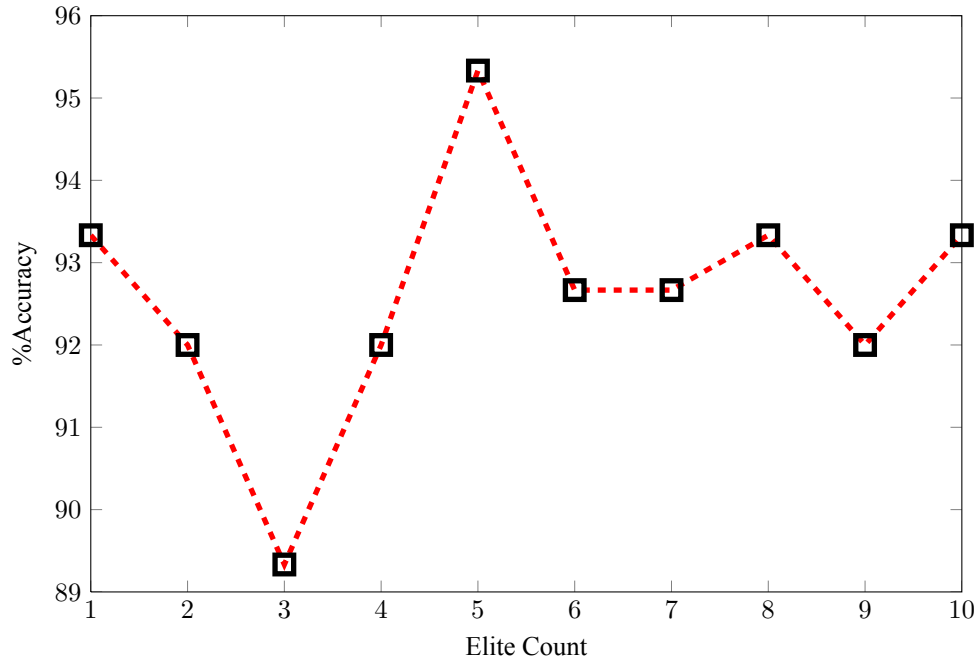
Πίνακας 6.8: Elite Count Music Data

Elite Count	Value
C1vsC2	4
C1vsC2vsC3	5
C1vsC2vsC3vsC4	3
C1vsC2vsC3vsC4vsC5	5
C1vsC2vsC3vsC4vsC5vsC6	1
C1vsC2vsC3vsC4vsC5vsC6vsC7	5
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	1
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	4
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	6

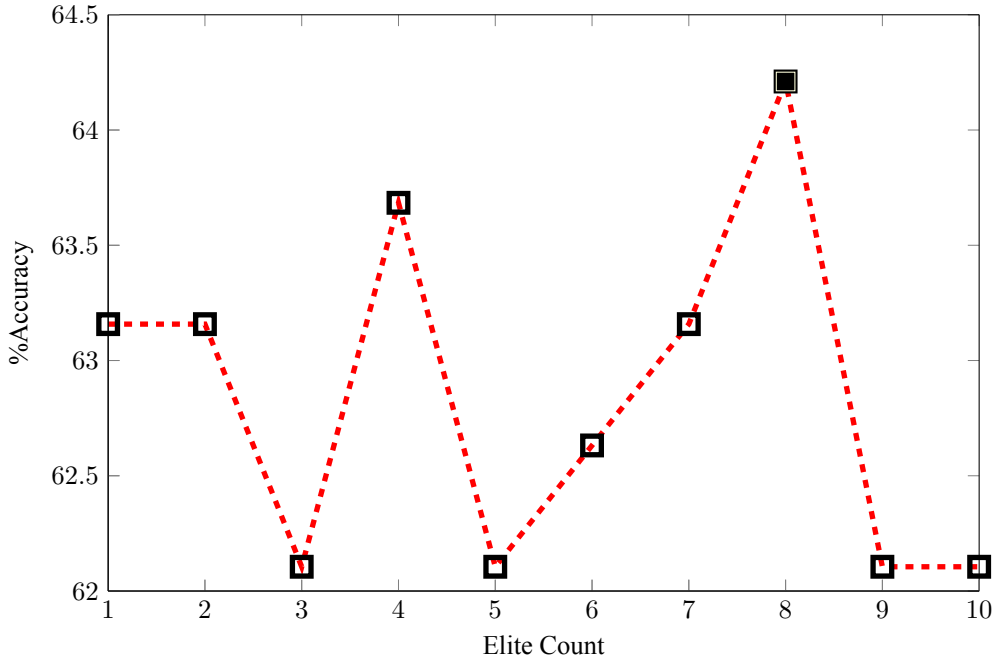
Ο πίνακας (6.8) επιβεβαιώνει ότι οι τιμές του Elite Count πρέπει να είναι όσο το δυνατόν μικρότερες. Είναι φανερό ότι στα περισσότερα πειράματα η μεγαλύτερη τιμή της απόδοσης πετυχαίνεται όταν η τιμή του Elite Count είναι μικρότερη του 5.



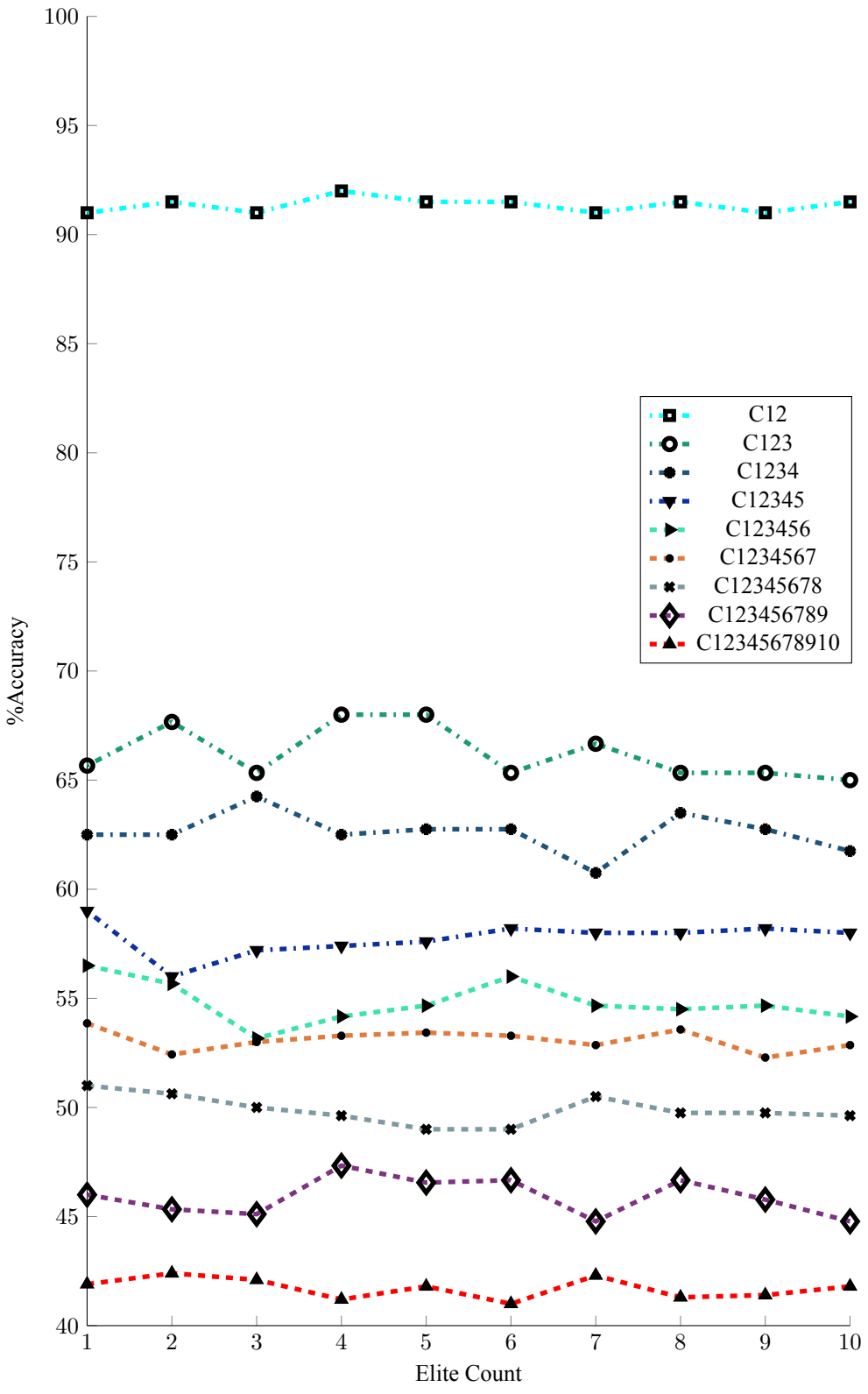
Σχήμα 6.1: Prima Indians Variable Elite Count



Σχήμα 6.2: Iris Data Variable Elite Count



Σχήμα 6.3: Sonar Data Variable Elite Count



Σχήμα 6.4: Music Features Variable Elite Count

6.4 Μεταβλητή Emax

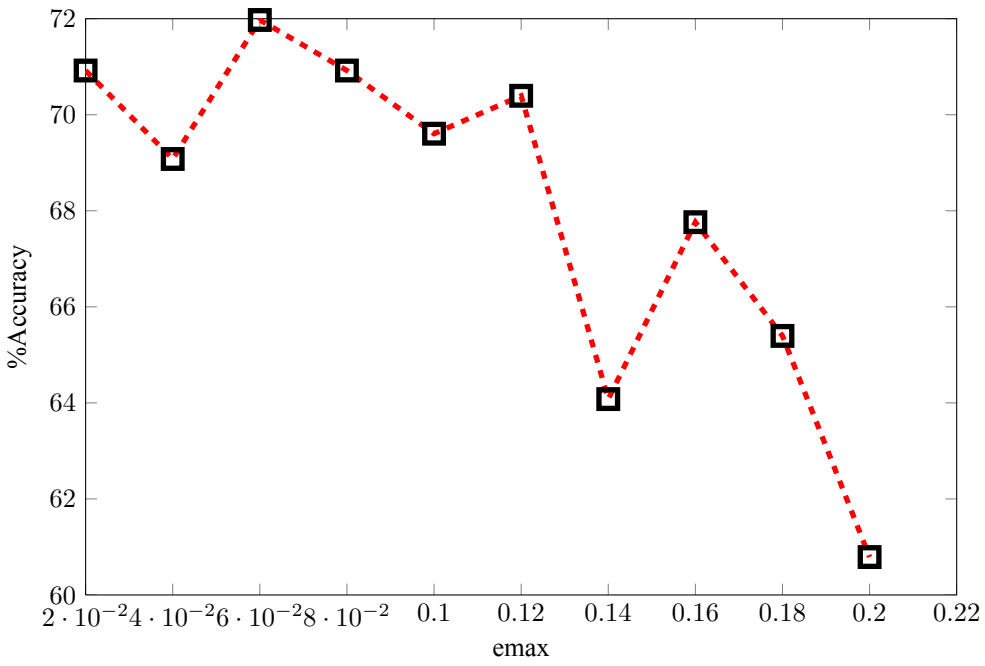
Η παράμετρος Emax είναι η απόσταση των ανιχνευτών (χρωμοσωμάτων) από τις τιμές προς ανίχνευση. Με άλλα λόγια είναι η απόσταση των λύσεων που παράγει ο γενετικός αλγόριθμος από τις πραγματικές λύσεις. Το διάστημα της παραμέτρου Emax είναι ανάλογο με την κατανομή του προς εξέταση συνόλου δεδομένων. Για παράδειγμα, στα σύνολα Iris Data, Sonar Data, Prima Indians παρατηρούμε ότι η παράμετρος για τιμές μεγαλύτερες του 0.2 φθίνει απότομα σε πολύ χαμηλές τιμές και για αυτό το λόγο το διάστημα σε αυτά τα σύνολα κινείται στο $[0.01 - 0.2]$. Από την άλλη πλευρά, στα σύνολα δεδομένων Music Data εξετάζουμε την παράμετρο στο διάστημα $[0.01 - 0.5]$ διότι η μεγαλύτερη τιμή επιτυγχάνεται για τιμές μεγαλύτερες του 0.2.

Γενικά, μπορούμε να πούμε ότι η τιμή που πετυχαίνει την μεγαλύτερη ακρίβεια ταξινόμησης εξαρτάται από την κατανομή ενός συνόλου δεδομένων. Μία μικρή τιμή για την παράμετρο Emax μπορεί να οδηγήσει σε μικρή ακρίβεια ταξινόμησης. Το γεγονός αυτό μπορεί να οφείλεται στο ότι ο γενετικός αλγόριθμος δεν μπορεί να προσεγγίσει αυτή την τιμή. Αυτό το παρατηρούμε στο σύνολο Sonar Data, εκεί για $Emax = 0.01$ λαμβάνουμε την μικρότερη ακρίβεια και για τιμή 0.16 και 0.17 την μεγαλύτερη (6.7). Στη συνέχεια για το σύνολο Prima Indians Diabetes η μεγαλύτερη τιμή λαμβάνεται για σχετικά μικρό $Emax$ για $Emax = 0.06$ (6.5) όπως και για το σύνολο Iris Data με $Emax = 0.04$ (6.6).

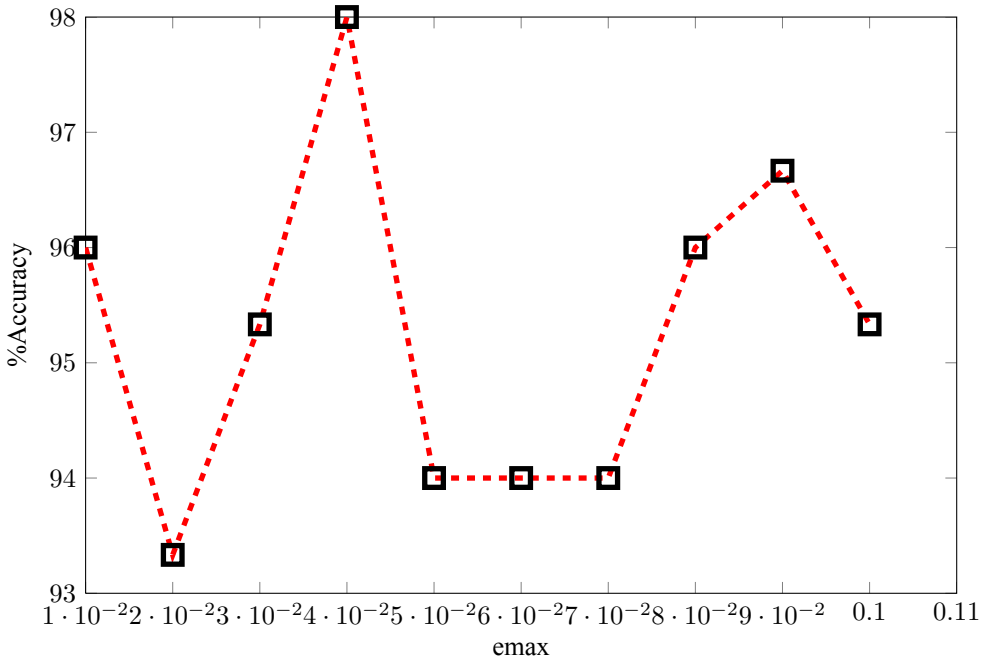
Τα αποτελέσματα για το σύνολο Music Data παρουσιάζονται στον πίνακα (6.9). Παρατηρούμε ότι σε αυτό το σύνολο το Emax λαμβάνει τιμές μεταξύ του 0.1 ως και 0.2. Τέλος, είναι εμφανές ότι μετά από την τιμή 0.2 η ταξινομική ακρίβεια φθίνει.

Πίνακας 6.9: Emax Music Data

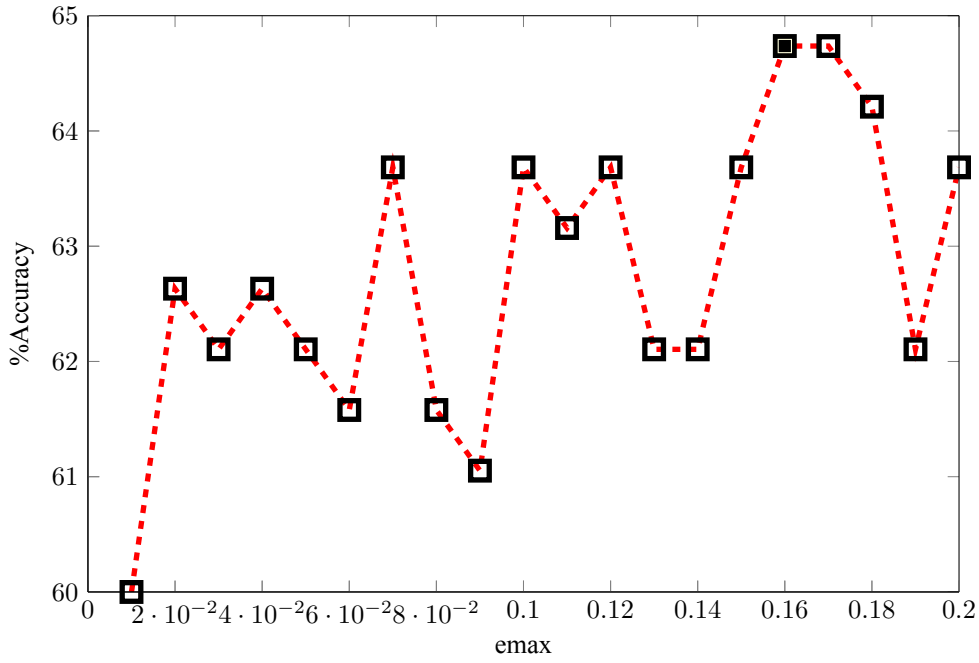
Emax	Value
C1vsC2	0.26
C1vsC2vsC3	0.23
C1vsC2vsC3vsC4	0.18
C1vsC2vsC3vsC4vsC5	0.1
C1vsC2vsC3vsC4vsC5vsC6	0.09
C1vsC2vsC3vsC4vsC5vsC6vsC7	0.07
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	0.21
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	0.1
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	0.1



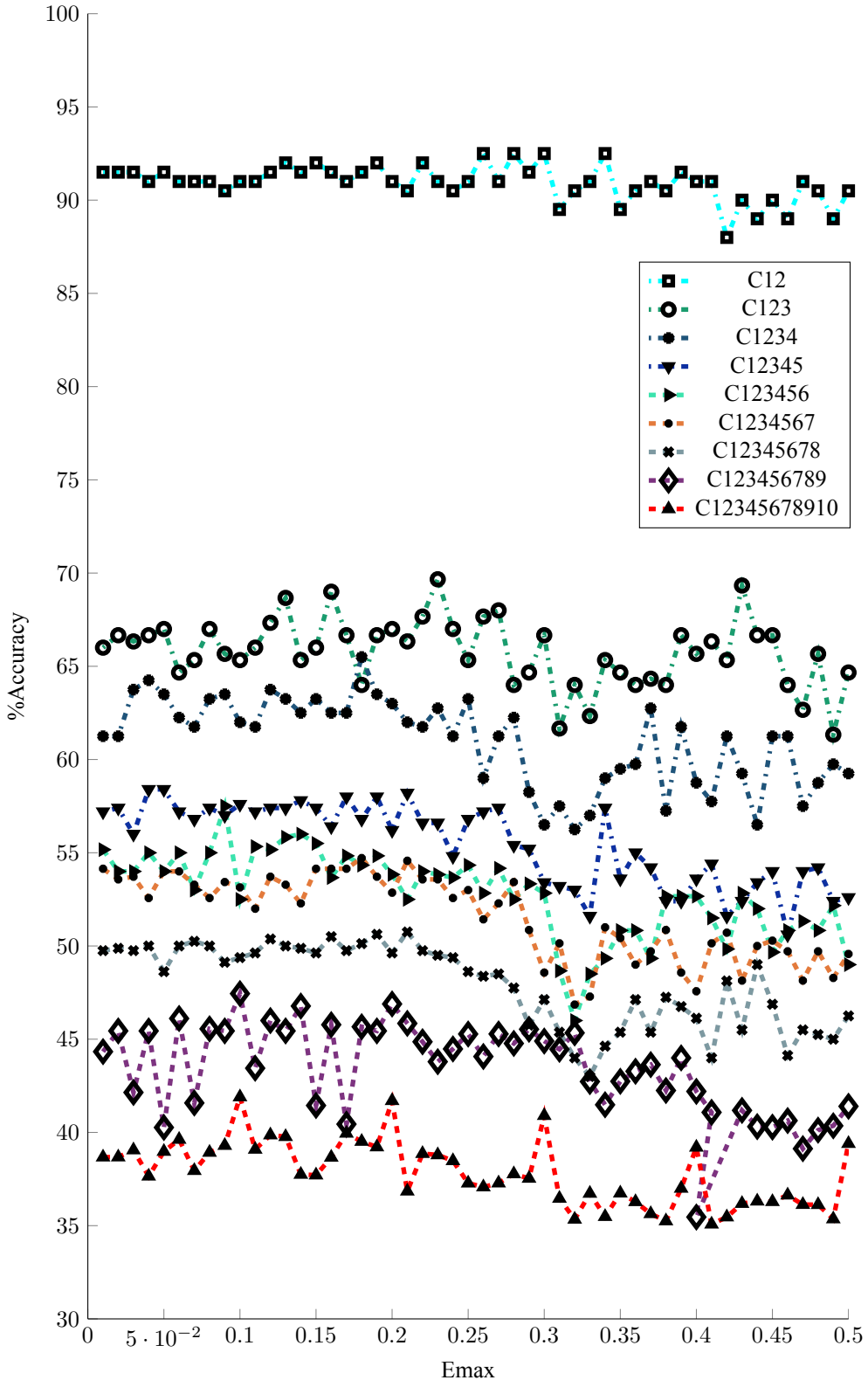
Σχήμα 6.5: PrimaIndians Variable Emax



Σχήμα 6.6: IrisData Variable Emax



Σχήμα 6.7: Sonar Data Variable Emax



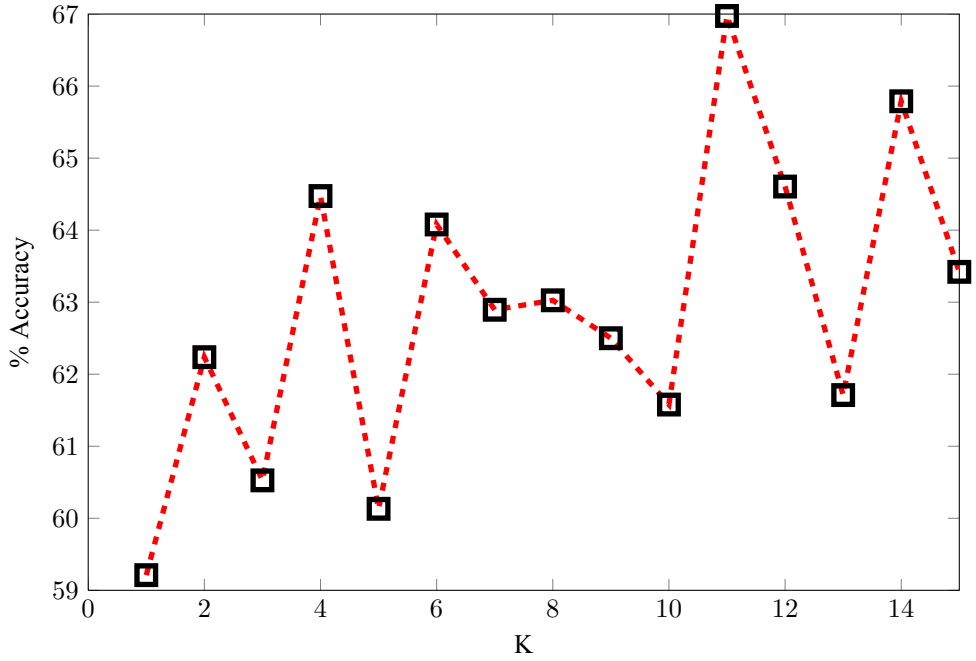
Σχήμα 6.8: Music Features Variable Emax

6.5 Μεταβλητή K

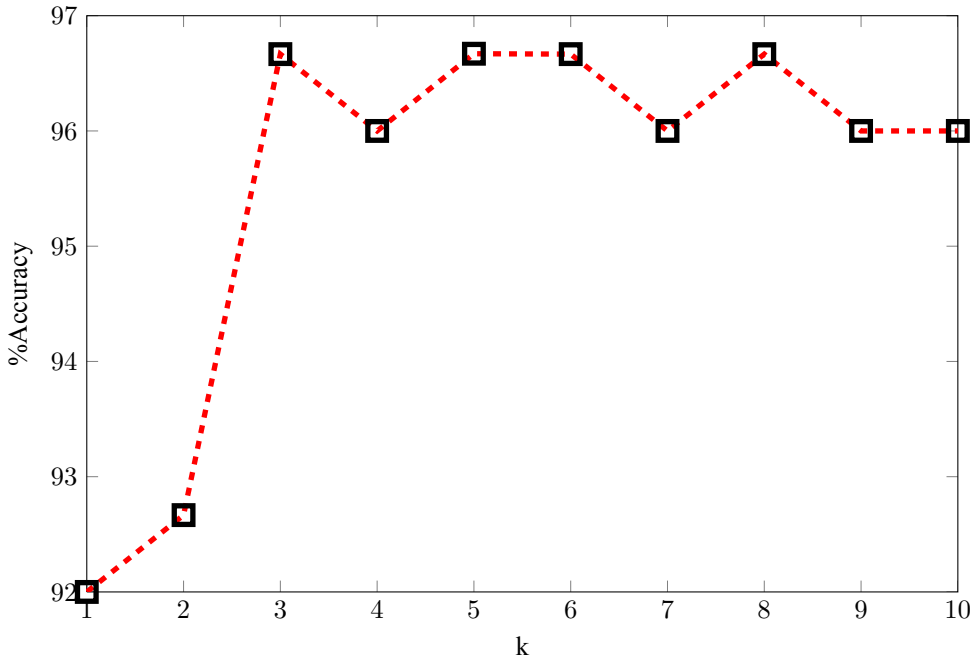
Η παράμετρος k είναι η παράμετρος που δέχεται ως είσοδο ο αλγόριθμος των k-πλησιέστερων γειτόνων. Με αυτόν τον τρόπο, ο αλγόριθμος ταξινομεί ανάλογα με την ψήφο k γειτόνων. Το διάστημα που κινείται το k είναι το $[1-10]$, δηλαδή η ψήφος από 1 ως 10 γειτόνων. Για το Prima Indians το καλύτερο k είναι στο 11 (6.9) το οποίο αποτελεί εξαίρεση. Παρατηρούμε στα γραφήματα ότι η ακρίβεια ταξινόμησης έχει την τάση να αυξάνει καθώς αυξάνει το k. Πιο συγκεκριμένα, για Iris Data (6.10) το k λαμβάνει τη μέγιστη τιμή του σε πολλές τιμές για $k = 3,\ ,k = 5,\ ,k = 6,\ k = 8$. Στο σύνολο δεδομένων Sonar Data (6.11) το k είναι ίσο με 2. Τέλος οι τιμές του k για το Music Data παρουσιάζονται στον πίνακα (6.10)

Πίνακας 6.10: K Music Data

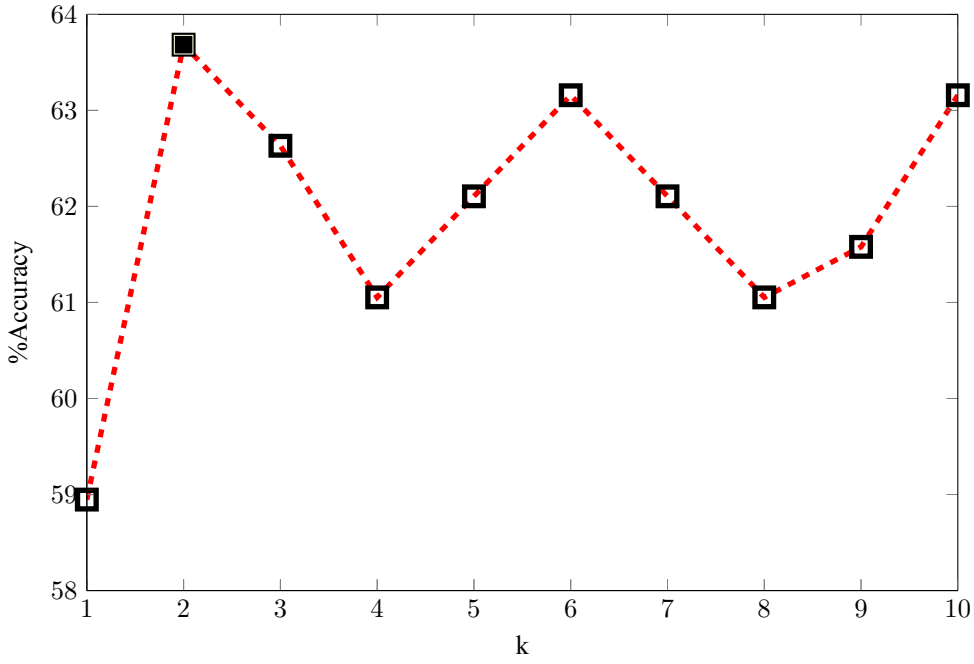
K	Value
C1vsC2	5
C1vsC2vsC3	9
C1vsC2vsC3vsC4	9
C1vsC2vsC3vsC4vsC5	3
C1vsC2vsC3vsC4vsC5vsC6	6
C1vsC2vsC3vsC4vsC5vsC6vsC7	5
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	6
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	4
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	3



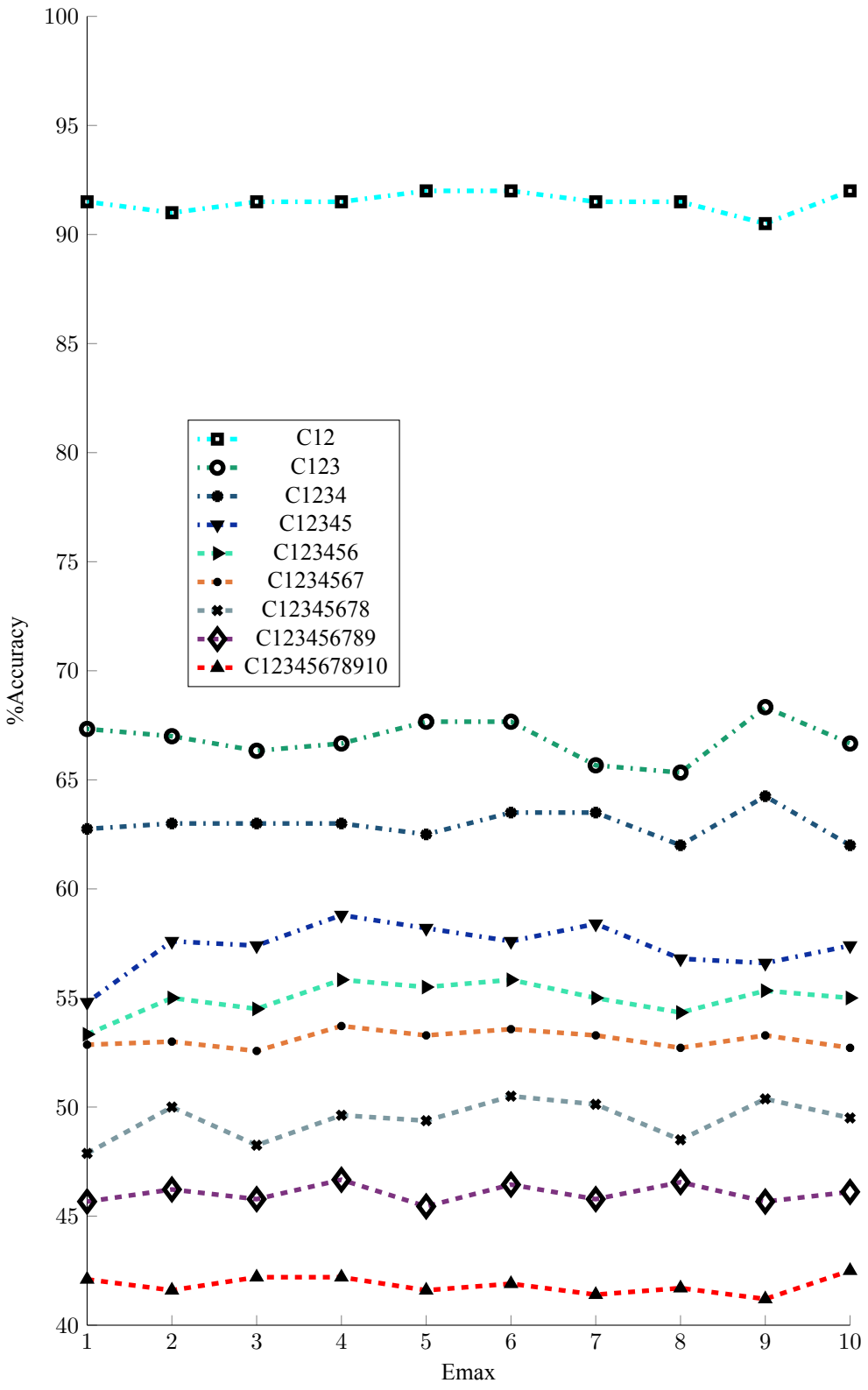
Σχήμα 6.9: PrimaIndians Variable K



Σχήμα 6.10: IrisData Variable K



Σχήμα 6.11: Sonar Data Variable K



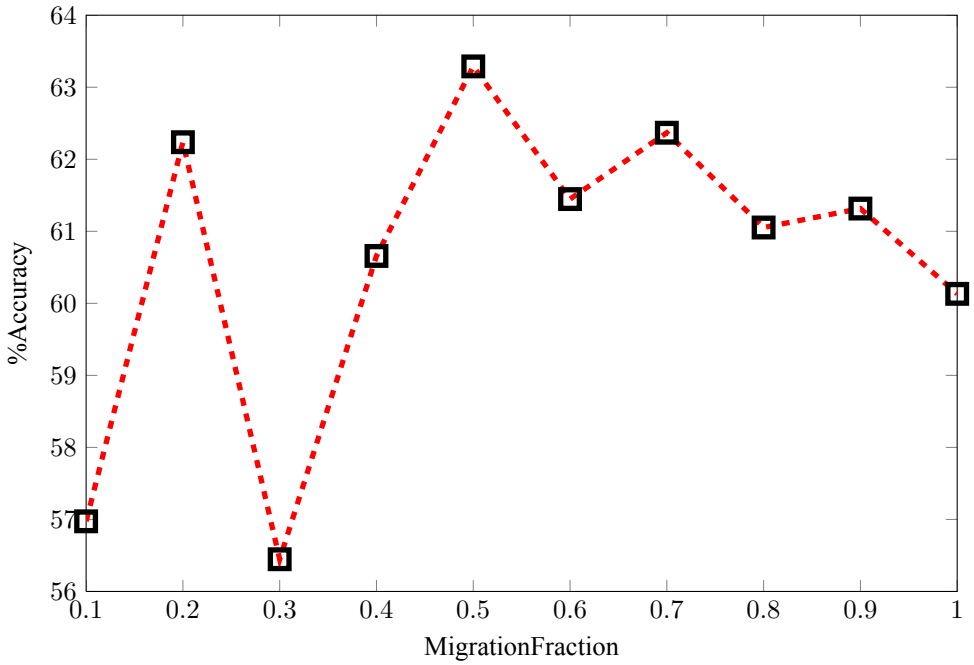
Σχήμα 6.12: Music Features Variable K

6.6 Migration Fraction

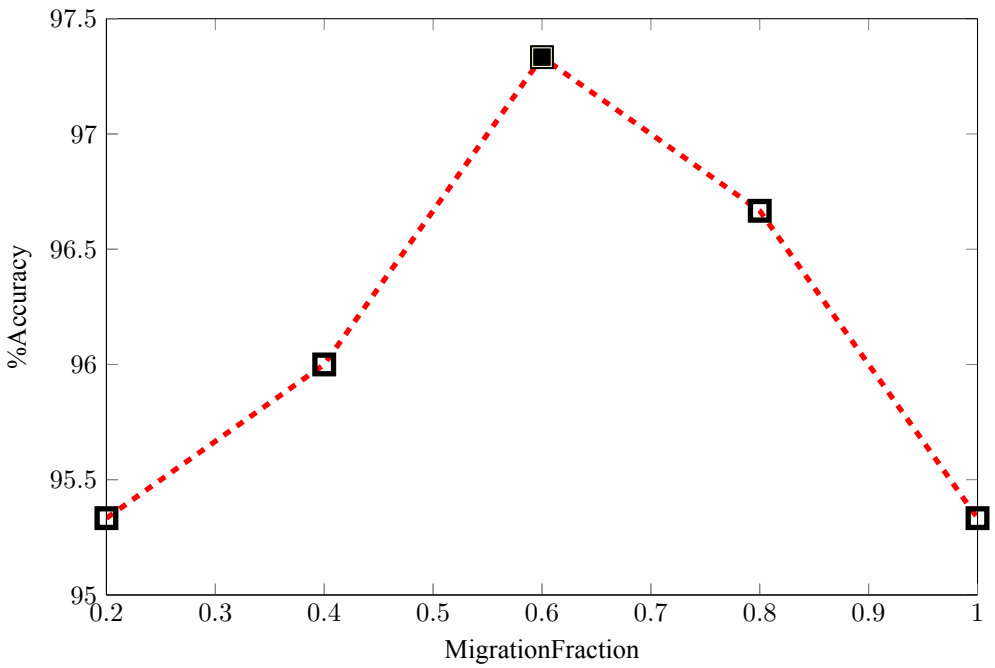
Η παράμετρος Migration Fraction είναι ένα κλάσμα που καθορίζει πόσα χρωμοσώματα (individuals) θα ανταλλάσσονται μεταξύ των υποπληθυσμών. Σε κάθε γενιά τα καλύτερα χρωμοσώματα ενός υποπληθυσμού αντικαθιστούν τα χειρότερα χρωμοσώματα. Το Migration Fraction λαμβάνει τιμές στο διάστημα [0-1]. Στο σύνολο Iris Data η καλύτερη τιμή για την παράμετρο είναι $MigrationFraction = 0.6$ (6.14), για το σύνολο Prima Indians $MigrationFraction = 0.5$ (6.13) ενώ για το Sonar Data $MigrationFraction = 1$ (6.15). Τέλος, οι τιμές του Migration Fraction για το σύνολο παρουσιάζονται στον πίνακα (6.15).

Πίνακας 6.11: Migration Fraction Music Data

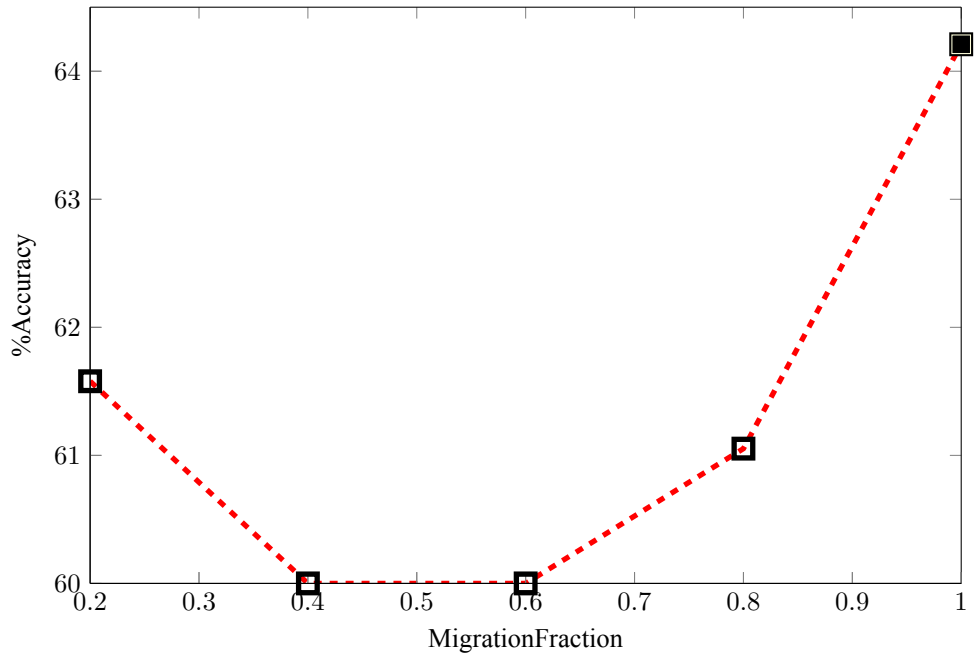
Migration Fraction	Value
C1vsC2	0.9
C1vsC2vsC3	0.6
C1vsC2vsC3vsC4	0.6
C1vsC2vsC3vsC4vsC5	0.7
C1vsC2vsC3vsC4vsC5vsC6	0.6
C1vsC2vsC3vsC4vsC5vsC6vsC7	0.2
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	0.1
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	0.9
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	0.7



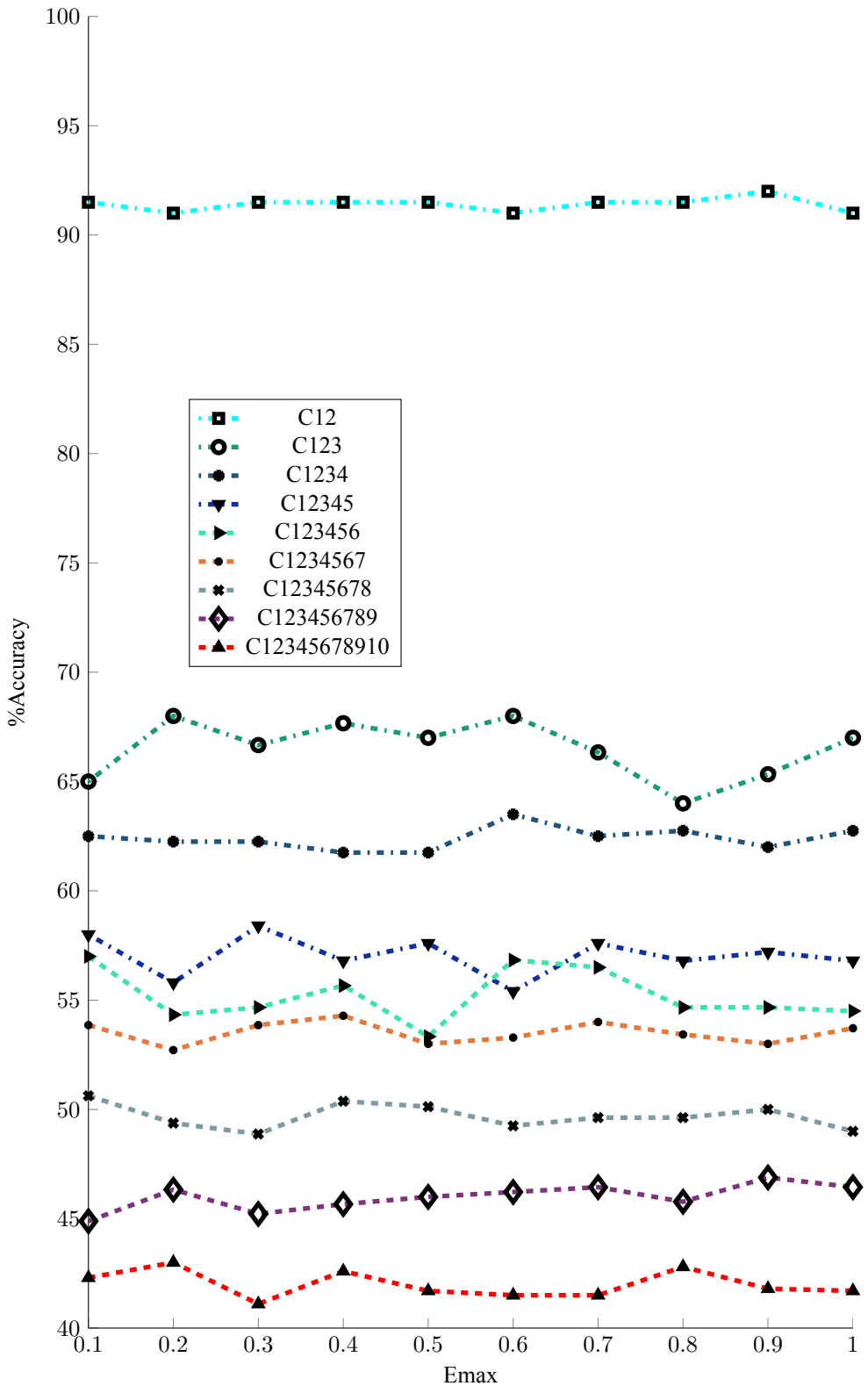
Σχήμα 6.13: PrimaIndians Variable Migration Function



Σχήμα 6.14: IrisData Variable Migration Function



Σχήμα 6.15: Sonar Data Variable Migration Function



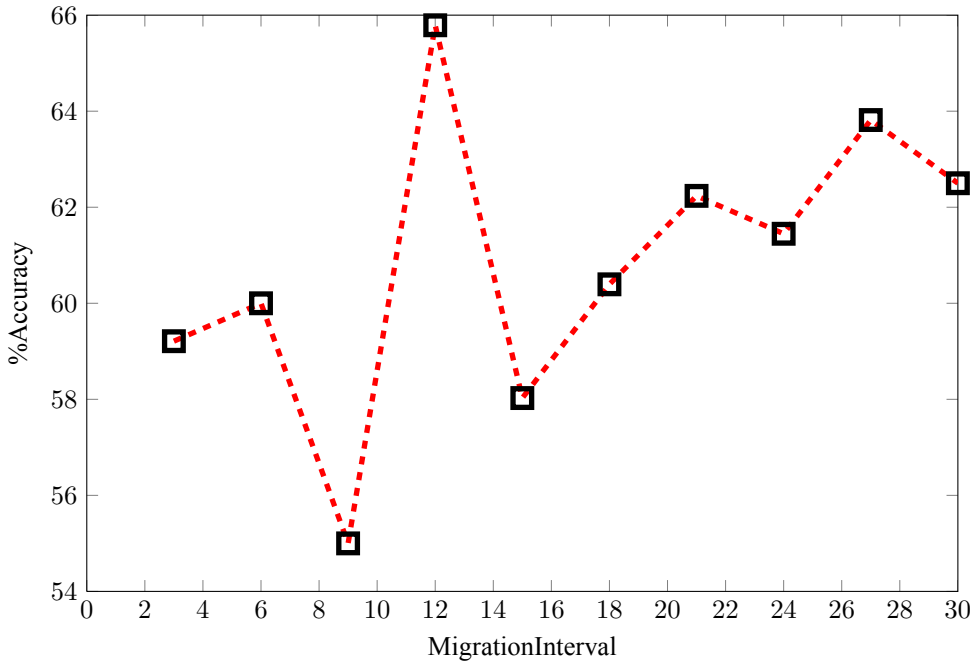
Σχήμα 6.16: Music Features Variable Migration Function

6.7 Μεταβλητή Migration Interval

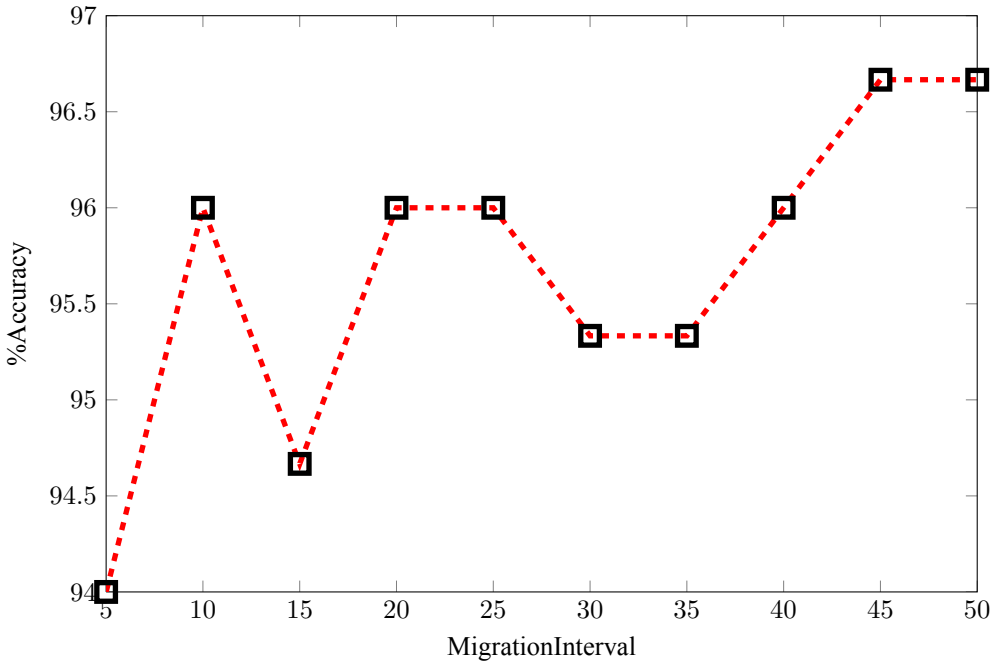
Η παράμετρος Migration Interval καθορίζει σε πόσες γενιές θα γίνει η διαδικασία migration. Το διάστημα της παραμέτρου αυτής εξαρτάται από το Population Size αλλά και το πλήθος των γενεών του γενετικού αλγορίθμου. Για το σύνολο Iris Data την μεγαλύτερη τιμή την έχει για $MigrationInterval = 45$ και 50 (6.18) , για το σύνολο Prima Indians $MigrationInterval = 12$ (6.17) και για το σύνολο Sonar Data $MigrationInterval = 20$ (6.19). Τέλος, για το σύνολο Music Data οι καλύτερες τιμές παρουσιάζονται παρακάτω (6.12)

Πίνακας 6.12: Migration Interval Music Data

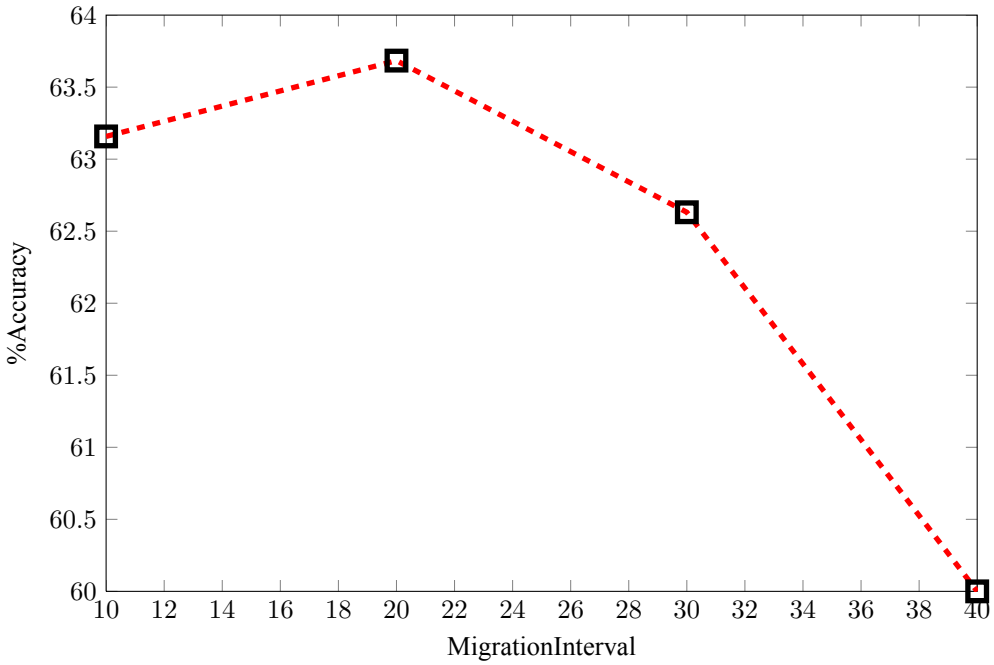
Migration Fraction	Value
C1vsC2	48
C1vsC2vsC3	72
C1vsC2vsC3vsC4	24
C1vsC2vsC3vsC4vsC5	15
C1vsC2vsC3vsC4vsC5vsC6	32
C1vsC2vsC3vsC4vsC5vsC6vsC7	56
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	24
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	80
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	15



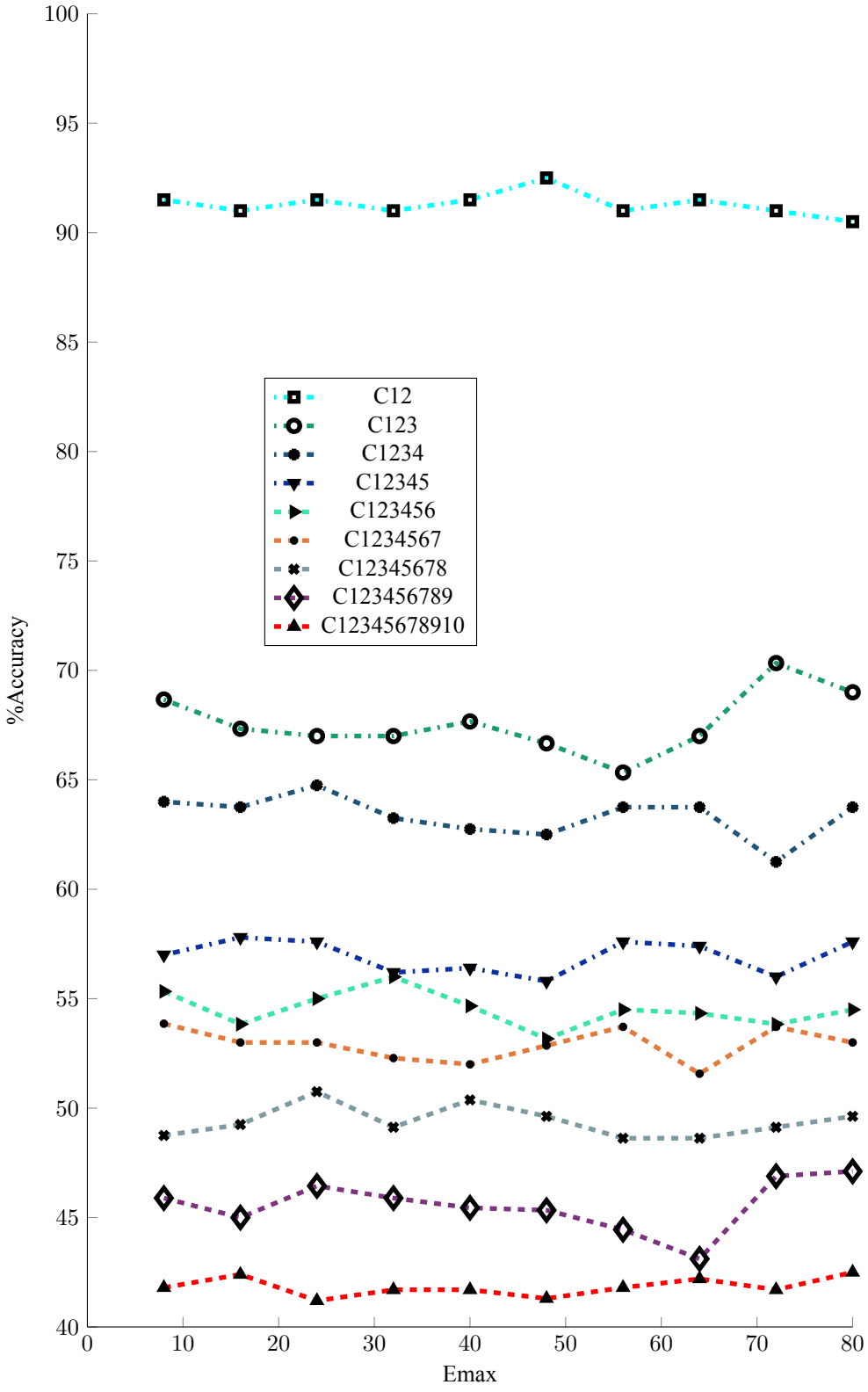
Σχήμα 6.17: PrimaIndians Variable Migration Interval



Σχήμα 6.18: IrisData Variable Migration Interval



Σχήμα 6.19: Sonar Data Variable Migration Interval



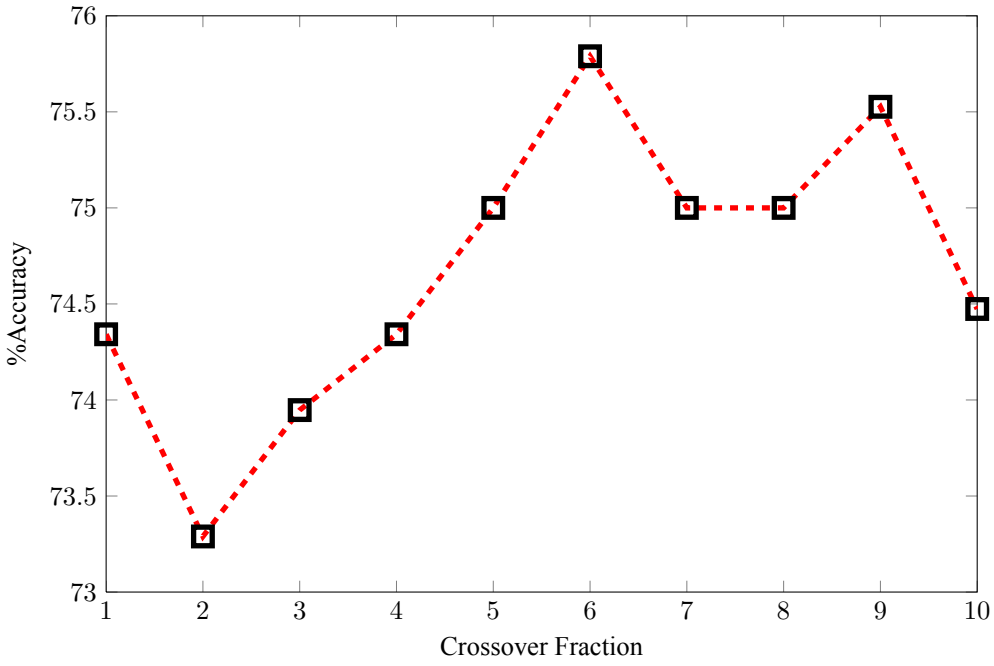
Σχήμα 6.20: Music Features Variable Migration Interval

6.8 Crossover Fraction

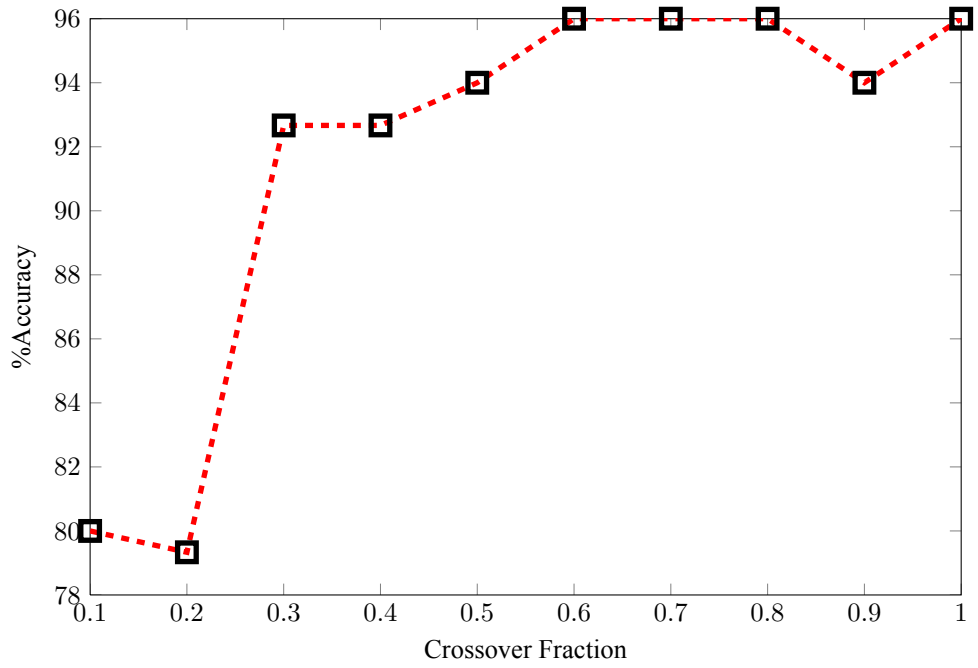
Η παράμετρος Crossover Fraction είναι το ποσοστό των χρωμοσωμάτων (individuals) που παράγονται από την διαδικασία της διασταύρωσης. Αυτή η παράμετρος καθορίζει και το ποσοστό των χρωμοσωμάτων που θα μεταλλαχθούν. Στην περίπτωση που το $CrossoverFraction = 1$ τότε όλα τα χρωμοσώματα παράγονται από τη διαδικασία της διασταύρωσης. Από την άλλη, όταν $CrossoverFraction = 0$ τότε όλα τα χρωμοσώματα παράγονται από τη διαδικασία της μετάλλαξης. Παρατηρούμε από τα γραφήματα (6.24), (6.21), (6.23), (6.22), ότι σε ορισμένα σύνολα είναι πιο σημαντική η διαδικασία της διασταύρωσης και σε ορισμένα η διαδικασία της μετάλλαξης. Πιο συγκεκριμένα, στο σύνολο Iris Data η $CrossoverFraction = 1$, στο σύνολο Sonar Data $CrossoverFraction = 0$, στο σύνολο Prima Indians 0.6. Τέλος τα αποτελέσματα για το Music Data παρουσιάζονται στον πίνακα (6.13).

Πίνακας 6.13: Crossover Fraction Music Data

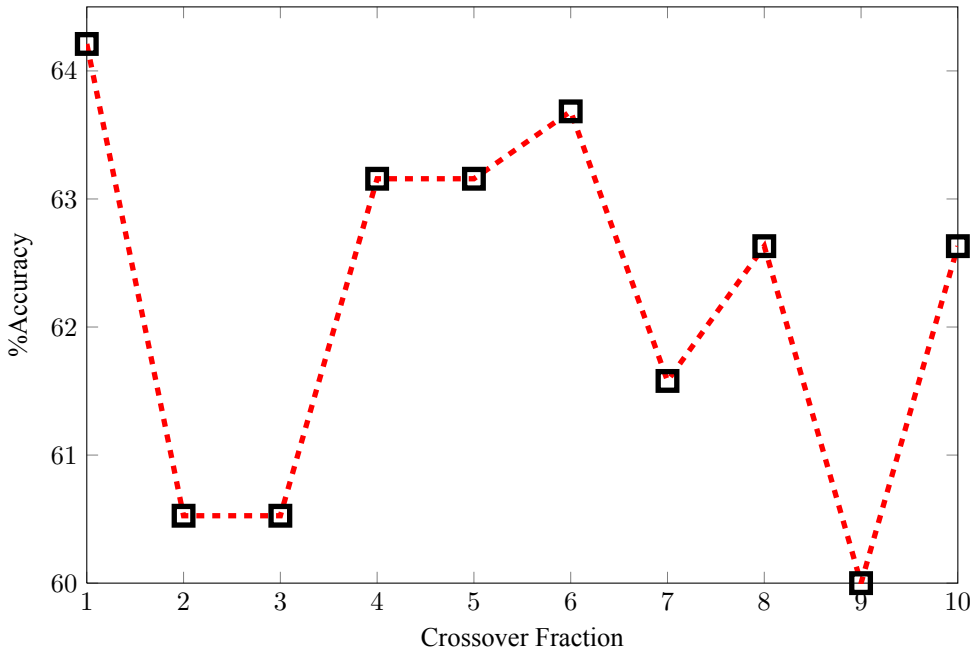
Crossover Fraction	Value
C1vsC2	0.8
C1vsC2vsC3	0.3
C1vsC2vsC3vsC4	0.8
C1vsC2vsC3vsC4vsC5	0.9
C1vsC2vsC3vsC4vsC5vsC6	0.9
C1vsC2vsC3vsC4vsC5vsC6vsC7	0.9
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	0.4
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	0.7
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	1



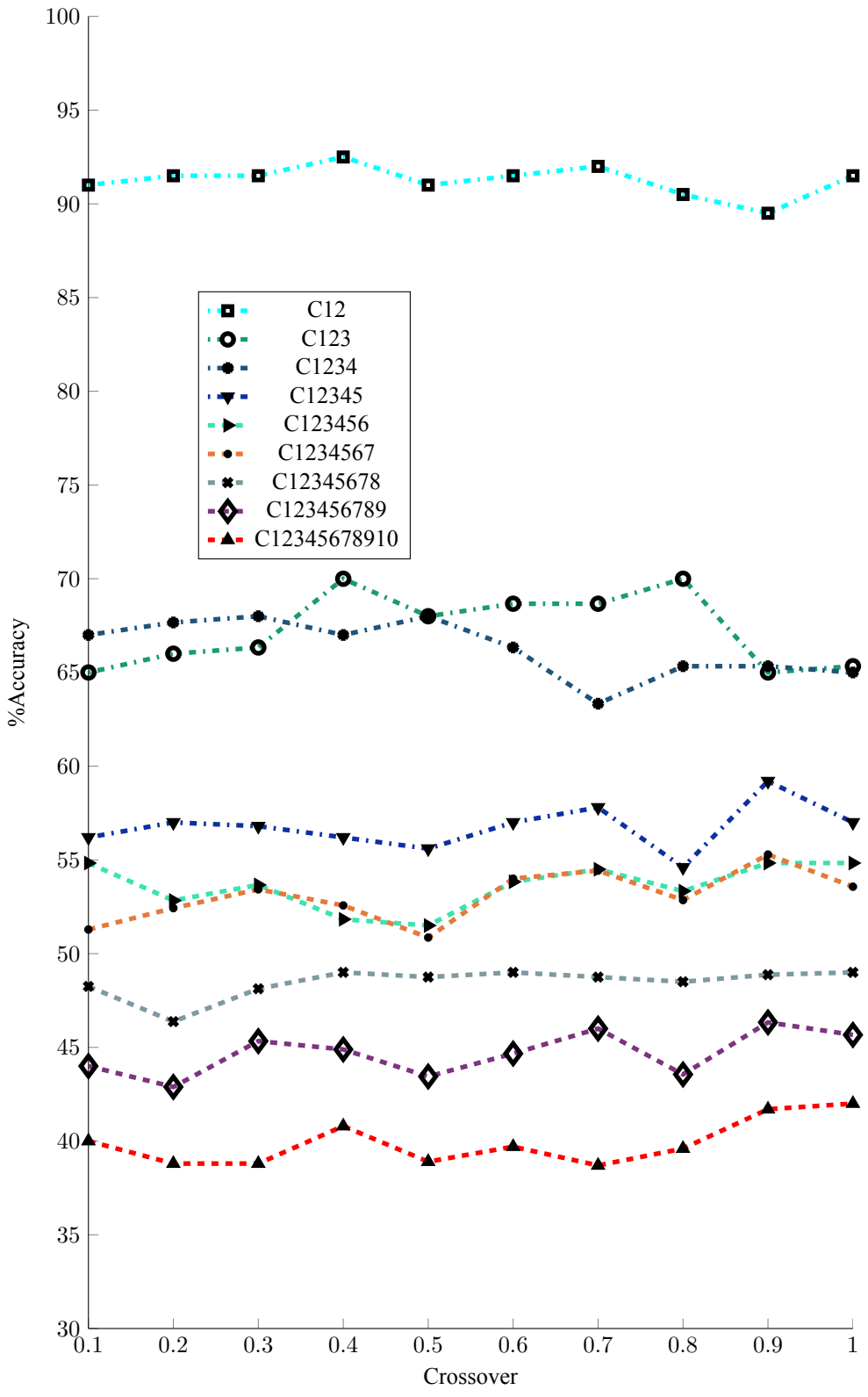
Σχήμα 6.21: Prima Indians Variable Crossover Fraction



Σχήμα 6.22: IrisData Variable Crossover Fraction



Σχήμα 6.23: Sonar Data Variable Crossover Fraction



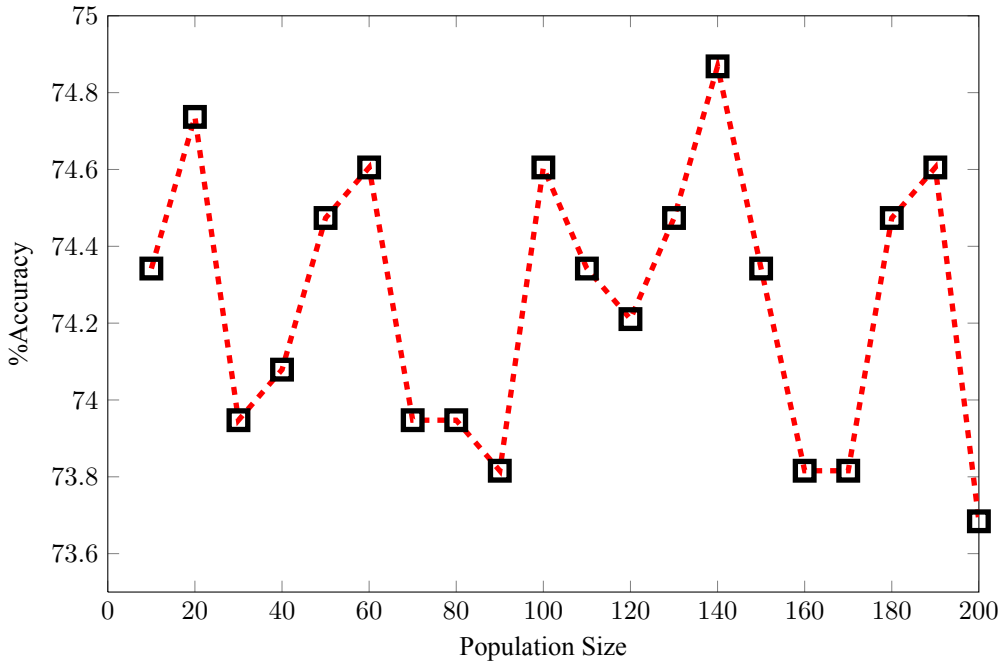
Σχήμα 6.24: Music Features Variable Crossover Fraction

6.9 Population Size

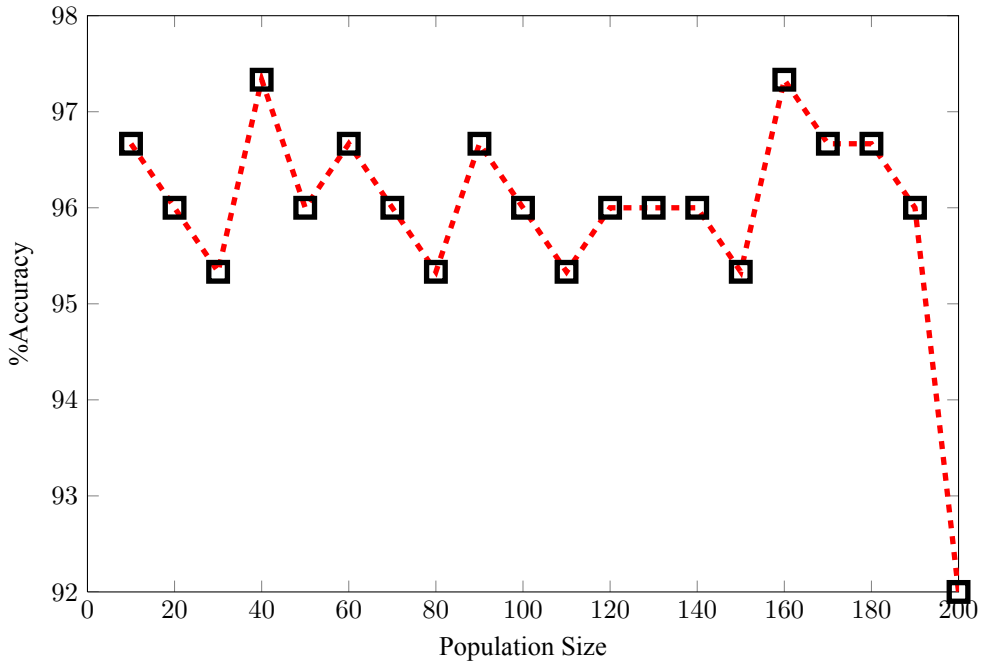
Το Population Size καθορίζει πόσα χρωμοσώματα θα υπάρχουν σε κάθε γενιά εκτέλεσης του γενετικού αλγορίθμου. Με μία μεγάλη τιμή στο Population Size ο αλγόριθμος αναζητά λύσεις σε μεγαλύτερο χώρο αναζήτησης αλλά από την άλλη πλευρά αυξάνεται ο χρόνος εκτέλεσης του. Η παράμετρος αυτή είναι από τις σημαντικότερες του αλγορίθμου αφού έχει σχέση με το μέγεθος του χώρου αναζήτησης των λύσεων. Θα χρησιμοποιήσουμε τιμές για το Population Size στο διάστημα [20-200]. Είναι φανερό σε όλα τα σύνολα δεδομένων ότι όσο το Population Size αυξάνει τόσο αυξάνεται και η ακρίβεια ταξινόμησης. Παρατηρούμε ότι στο σύνολο Iris Data $PopulationSize = 160$ και $PopulationSize = 40$ παρουσιάζει την μεγαλύτερη ακρίβεια, στο σύνολο Prima Indians $PopulationSize = 140$, ενώ στο σύνολο Sonar $PopulationSize = 90$. Για το σύνολο δεδομένων Music Data τα αποτελέσματα παρουσιάζονται στον πιο κάτω πίνακα (6.14)

Πίνακας 6.14: Population Size Music Data

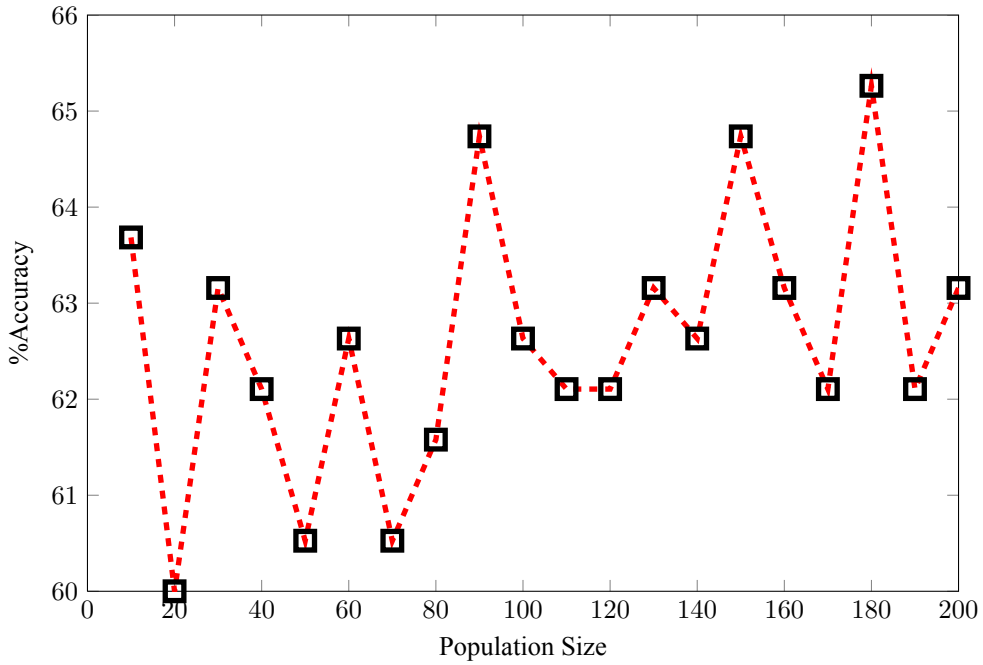
Population Size	Value
C1vsC2	90
C1vsC2vsC3	80
C1vsC2vsC3vsC4	40
C1vsC2vsC3vsC4vsC5	80
C1vsC2vsC3vsC4vsC5vsC6	50
C1vsC2vsC3vsC4vsC5vsC6vsC7	50
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	130
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	80
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	80



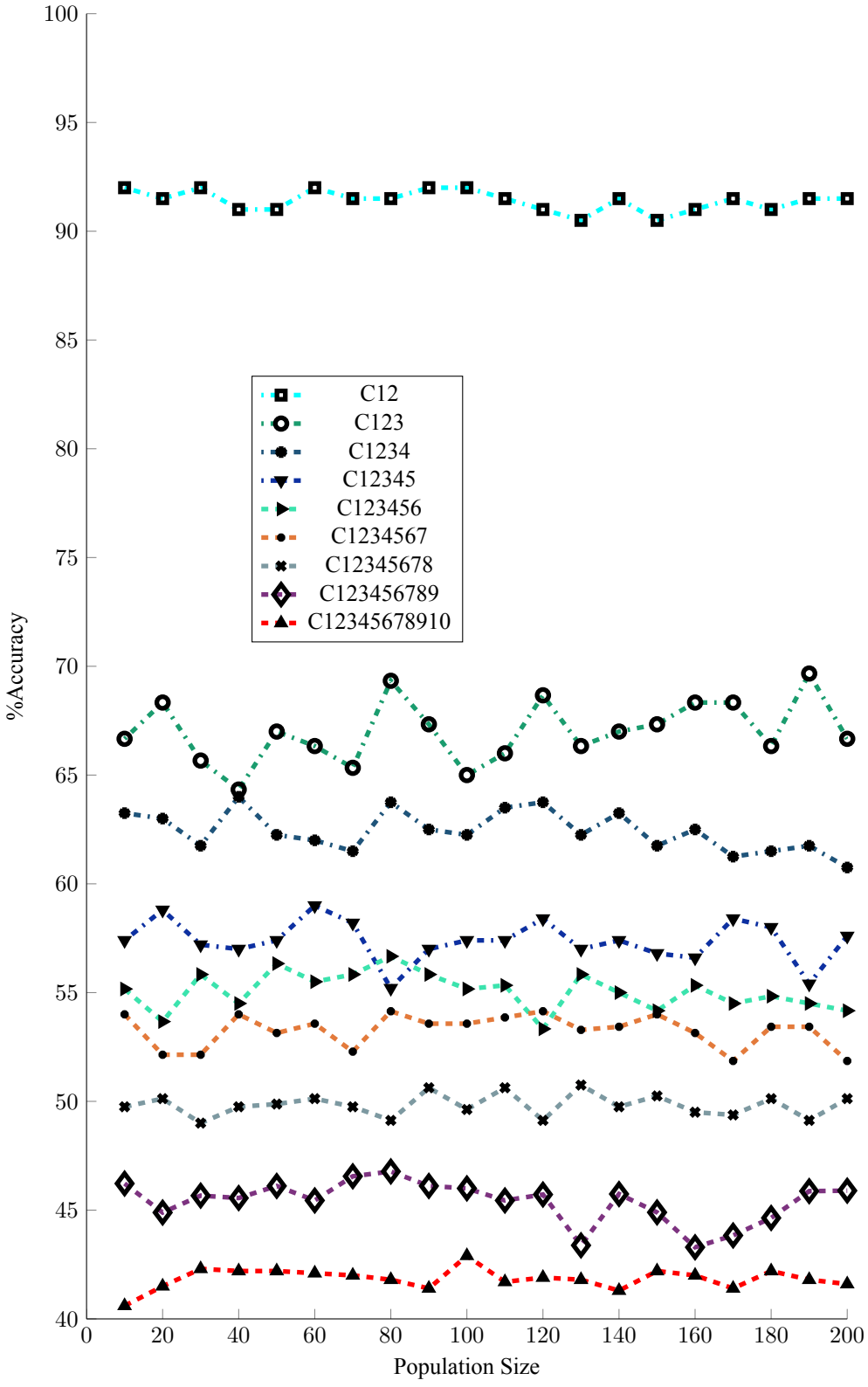
Σχήμα 6.25: Prima Indians Variable Population Size



Σχήμα 6.26: Iris Data Variable Population Size



Σχήμα 6.27: Sonar Data Variable Population Size



Σχήμα 6.28: Music Features Variable Population Size

6.10 Τελικά Αποτελέσματα

Έπειτα από την διεξαγωγή των παραπάνω πειραμάτων για την εύρεση των καλύτερων παραμέτρων των συνόλων που εξετάζουμε, διερευνούμε την συμπεριφορά του αλγορίθμου με διάφορες συναρτήσεις του Selection Function και του Crossover Function. Τα πειράματα αυτά έχουν πραγματοποιηθεί με τα καλύτερα αποτελέσματα των παραμέτρων Emax, Population Size, Elite Count, K, Migration Fraction, Migration Interval, Crossover Fraction.

Selection Function

Για την Selection Function έχουν χρησιμοποιηθεί τέσσερις διαφορετικές συναρτήσεις.

- **Roulette wheel** Με αυτή την διαδικασία δημιουργούμε μία επιφάνεια ρουλέτας σε σχήμα πίτας όπου κάθε μέλος απεικονίζεται στη ρουλέτα ανάλογα με το ποσοστό της αθροιστικής του πιθανότητας. Για την επιλογή των μελών του νέου πληθυσμού εκτελούμε n περιστροφές της ρουλέτας. Ανάλογα με το ποσοστό που έχει κάθε μέλος πάνω στη ρουλέτα τόσο πιθανό είναι να επιλεγεί. Προφανώς, με αυτή τη μέθοδο επιλογής είναι δυνατόν κάποια μέλη του πληθυσμού να επιλεχθούν περισσότερες από μία φορές, με αυτά που είχαν την καλύτερη απόδοση στην προηγούμενη γενιά να έχουν τις περισσότερες πιθανότητες γι' αυτό.

- **Stochastic Uniform**
  Ο μηχανισμός αυτός δημιουργεί μία γραμμή στην οποία κάθε γονέας έχει το δικό του μέρος (διάστημα) ανάλογα με το ποσοστό της αθροιστικής του πιθανότητας. Ο αλγόριθμος αυτός κινείται πάνω στην γραμμή με βήματα ίσου μεγέθους και παίρνει τυχαία ένα γονέα πάνω στον οποίο βρέθηκε στο i-στο βήμα.

- **Remainder**
  Η διαδικασία αυτή λαμβάνει το ακέραιο μέρος ενός χρωμοσώματος και έπειτα χρησιμοποιεί τη Roulette wheel για την επιλογή απογόνων.

- **Tournament**
  Η μέθοδος αυτή πραγματοποιεί ένα είδος αγώνα ανάμεσα σε n χρωμοσώματα και έπειτα επιλέγει το καλύτερο για να δημιουργήσει απογόνους.

Crossover Function

Για την Crossover Function έχουν χρησιμοποιηθεί πέντε διαφορετικές συναρτήσεις.

- **Single Point**:
  Η διαδικασία Single Point αποτελεί την πιο διαδεδομένη διαδικασία διασταύρωσης. Μετά την επιλογή μελών του πληθυσμού για διασταύρωση , σχηματίζουμε ζευγάρια από μέλη και για κάθε ζευγάρι επιλέγεται τυχαία ένας ακέραιος αριθμός pos στο διάστημα [1, m-1], όπου m είναι το μήκος σε δυαδικά ψηφία του χρωμοσώματος κάθε μέλους. Ο αριθμός pos προσδιορίζει το σημείο διασταύρωσης. Τα επιλεγμένα ζευγάρια διασταυρώνονται και την θέση τους στον πληθυσμό την παίρνουν οι απόγονοί τους.

- **Scattered:**
  Η διαδικασία Scattered δημιουργεί τυχαία ένα δυαδικό πίνακα μεγέθους ίσου με το μέγεθος των χρωμοσωμάτων και επιλέγει για διασταύρωση τα γονίδια από το πρώτο γονέα για τιμή 1 και τα γονίδια του δεύτερου γονέα για τιμή 0.

- **Two Point:**
  Η διαδικασία αυτή επιλέγει δύο τυχαίους ακεραίους m και n στο διάστημα $[1, l]$ όπου l είναι το μέγεθος των χρωμοσωμάτων. Στη συνέχεια η συνάρτηση επιλέγει m γονίδια από τον πρώτο γονέα, $m + 1$ ως n από τον δεύτερο, και n ως l από τον πρώτο ξανά για να δημιουργήσει το παιδί.

- **Intermediate:**
  Η διασταύρωση με αυτή την τεχνική δημιουργεί παιδιά σε συνάρτηση με το μέσο βάρος των γονέων.

- **Arithmetic:**
  Η διασταύρωση με αυτή την τεχνική δημιουργεί παιδιά σε συνάρτηση με το μέσο αριθμητικό βάρος των γονέων.

- **Heuristic:**
  Η διασταύρωση με αυτή την τεχνική δημιουργεί παιδιά που έχουν περισσότερη γενετική πληροφορία από τον γονέα που παρουσιάζει την καλύτερη τιμή στην συνάρτηση καταλληλότητας.

6.11 Selection Function

Για την επιλογή της κατάλληλης συνάρτησης Selection Function πραγματοποιούμε 10 πειράματα 10 cross-validation με τα καλύτερα αποτελέσματα των παραμέτρων και στη συνέχεια υπολογίζουμε τον μέσο όρο των επαναλήψεων. Με αυτόν το τρόπο, στο σύνολο Prima Indians η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $RouletteWheel = 74.40$ όπως παρουσιάζεται στον πίνακα (6.15). Για το σύνολο Iris Data επιλέγουμε τη συνάρτηση $RouletteWheel = 96.33$ (6.16), ενώ για το σύνολο Sonar $Remainder = 63.52$ (6.17). Τέλος, για τα σύνολα Music Data έχουμε την εξής κατάσταση:

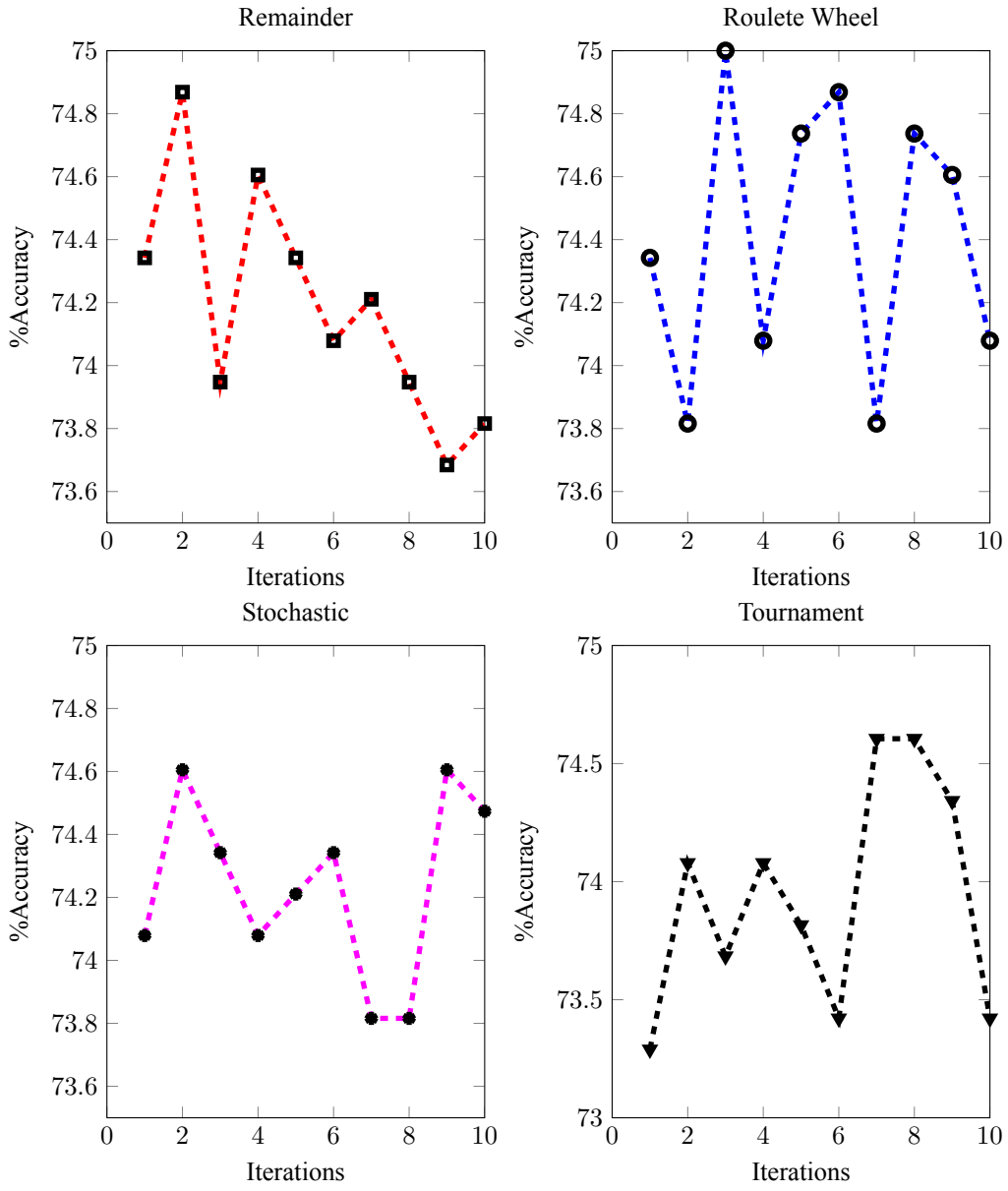
- **C1vsC2** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Stochastic = 91.65$ πίνακας (6.18).
- **C1vsC2vsC3** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Stochastic = 67.7$ πίνακας (6.19).
- **C1vsC2vsC3vsC4** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Stochastic = 62.77$ πίνακας (6.20).
- **C1vsC2vsC3vsC4vsC5** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Remainder = 57.48$ πίνακας (6.21).
- **C1vsC2vsC3vsC4vsC5vsC6** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Tournament = 55.15$ πίνακας (6.22).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Tournament = 53.55$ καθώς και με τη συνάρτηση $Stochastic = 53.55$ πίνακας (6.23).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Roulette = 50.03$ πίνακας (6.24).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9**

  Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Remainder = 46.6$ πίνακας (6.25).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10**

  Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Roulette = 41.99$ πίνακας (6.26).

Πίνακας 6.15: Prima Indians Selection Function

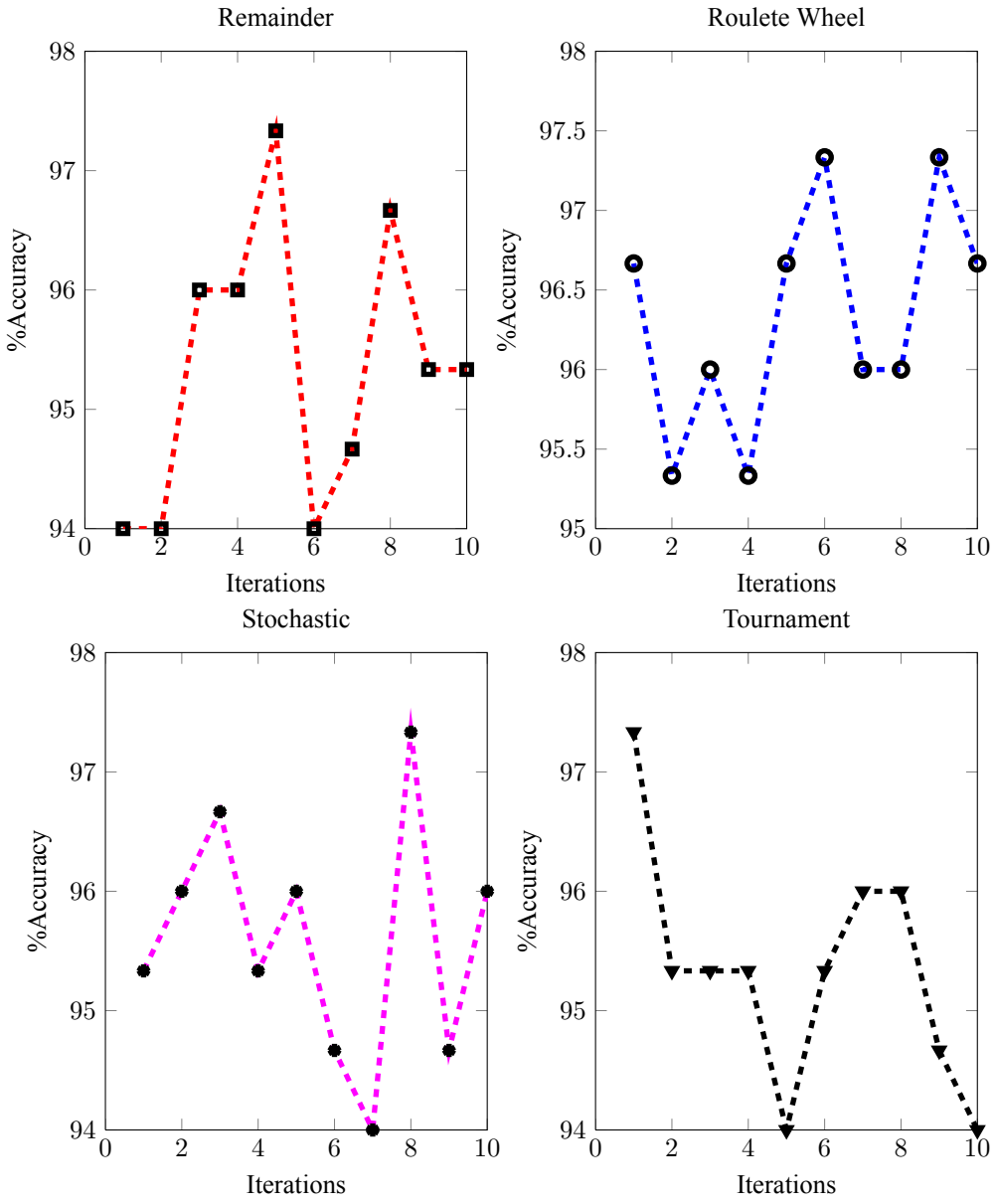
74.18421	Remainder
74.40789	Roulette
74.23684	Stochastic
73.93421	Tournament



Σχήμα 6.29: Prima Indians Selection Function

Πίνακας 6.16: Iris Data Selection Function

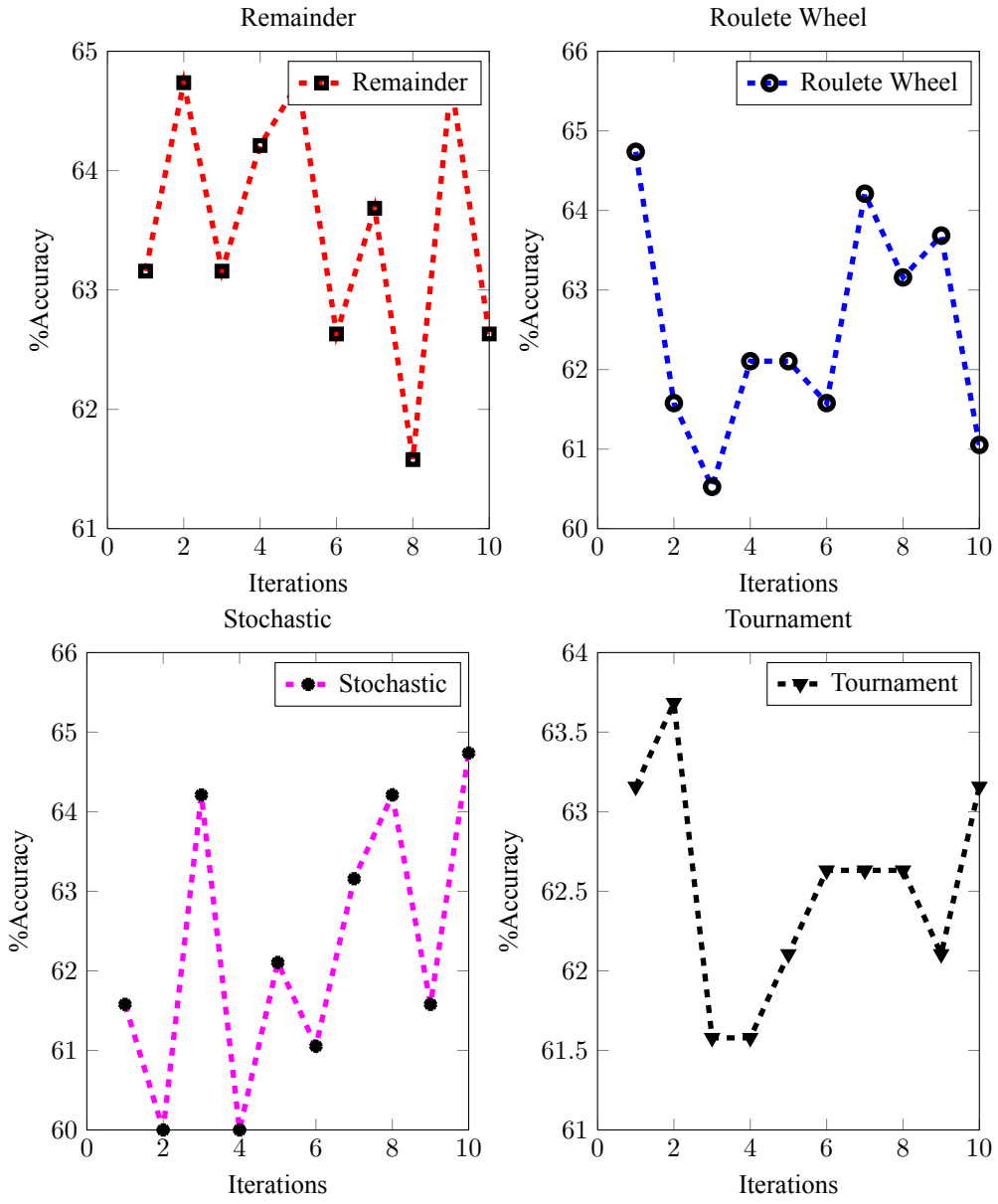
95.33333	Remainder
96.33333	Roulette
95.6	Stochastic
95.33333	Tournament



Σχήμα 6.30: Iris Data Selection Function

Πίνακας 6.17: Sonar Data Selection Function

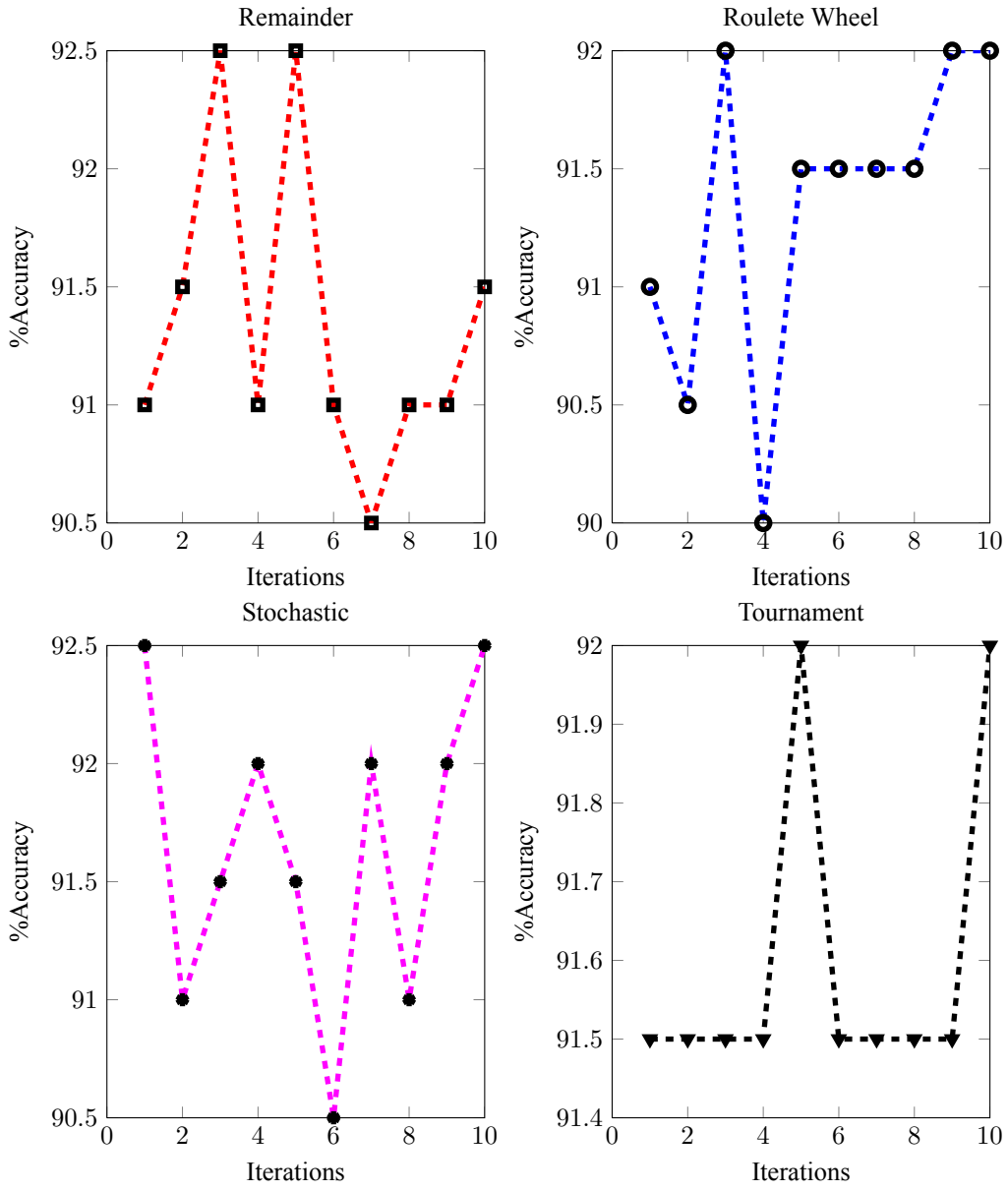
63.52632	Remainder
62.47368	Roulette
62.26316	Stochastic
62.52632	Tournament



Σχήμα 6.31: Sonar Data Selection Function

Πίνακας 6.18: C1vsC2 Selection Function

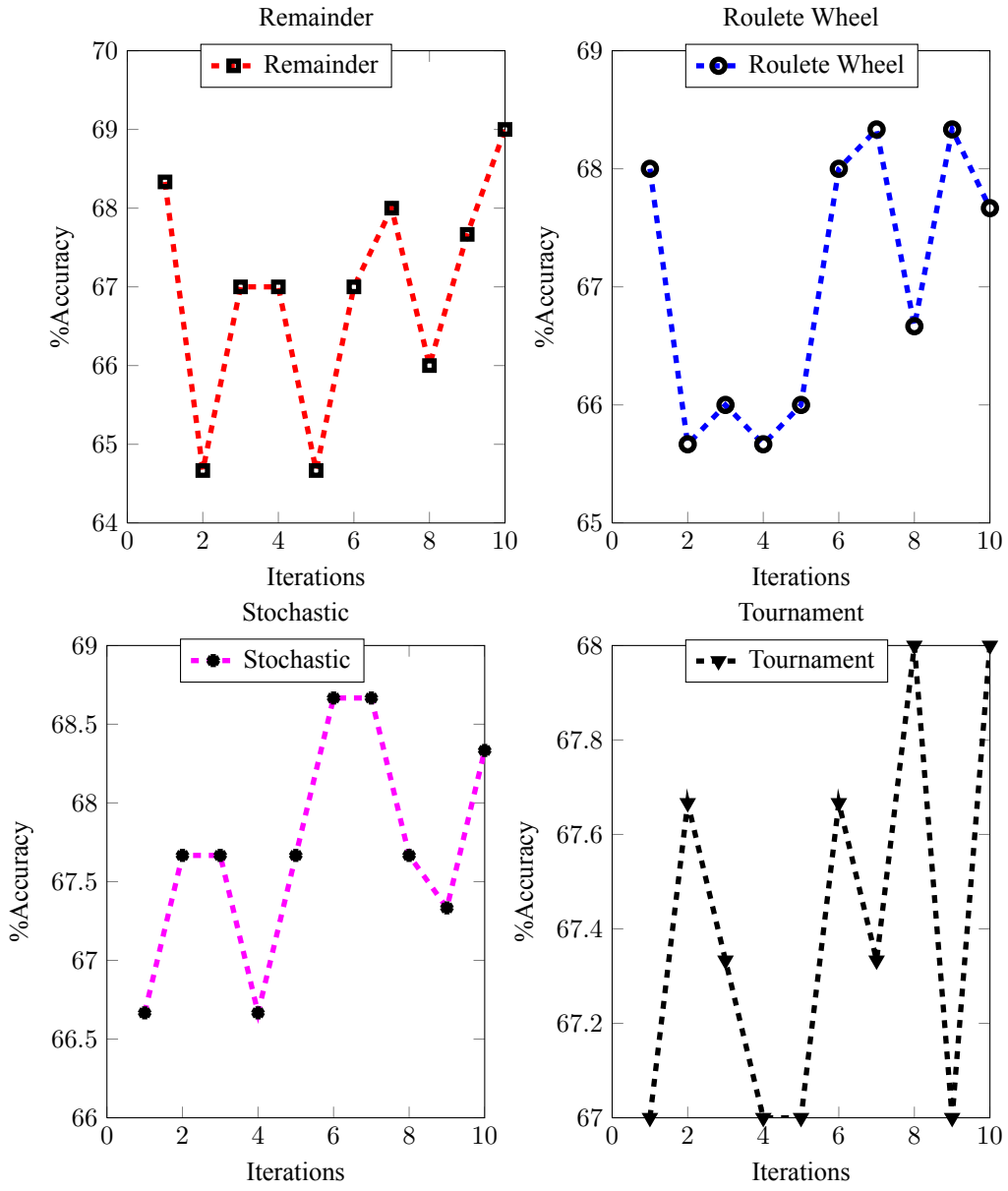
91.35	Remainder
91.35	Roulette
91.65	Stochastic
91.6	Tournament



Σχήμα 6.32: C1vsC2 Selection Function

Πίνακας 6.19: C1vsC2vsC3 Selection Function

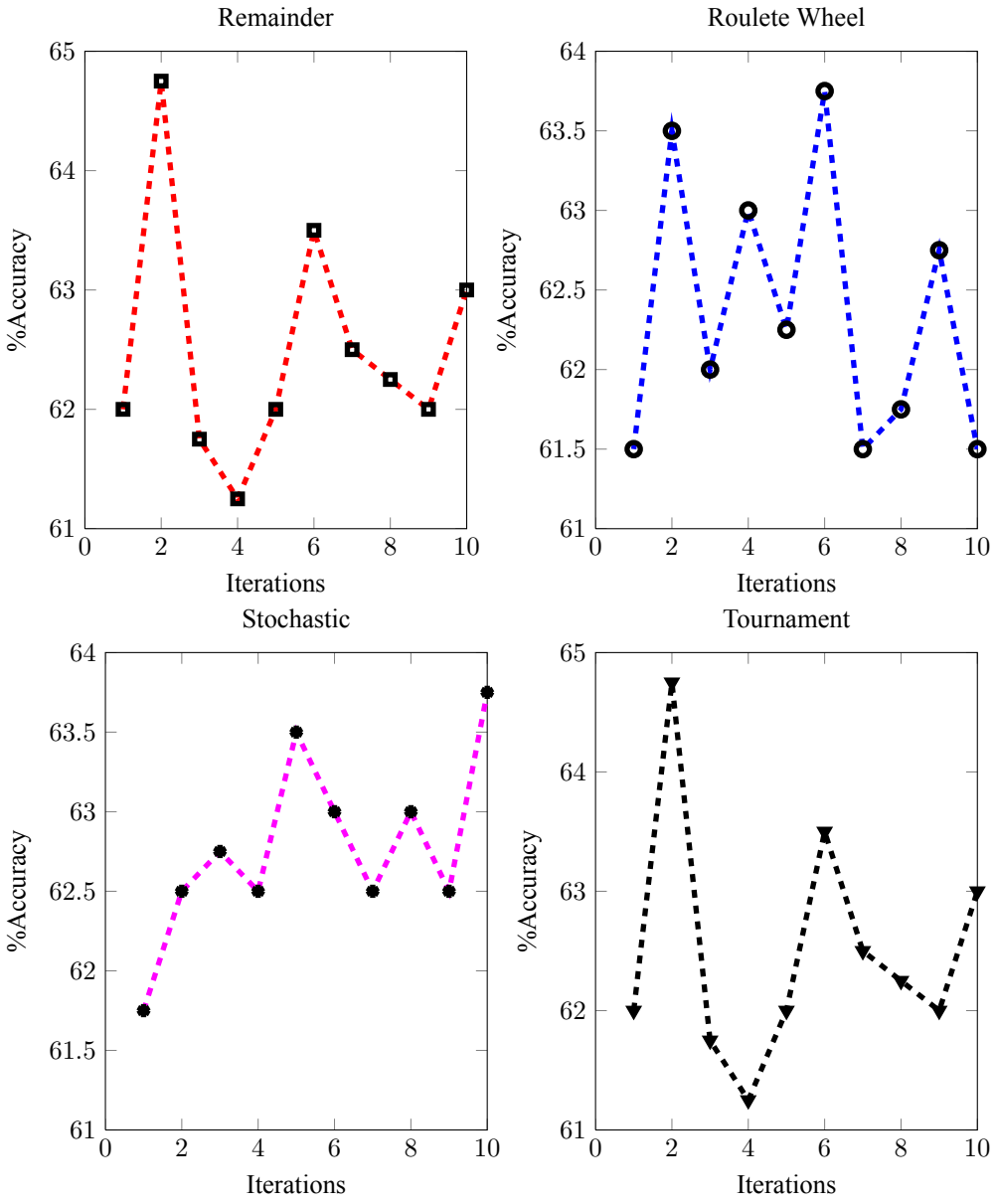
66.93333	Remainder
67.03333	Roulette
67.7	Stochastic
67.4	Tournament



Σχήμα 6.33: C1vsC2vsC3 Selection Function

Πίνακας 6.20: C1vsC2vsC3vsC4 Selection Function

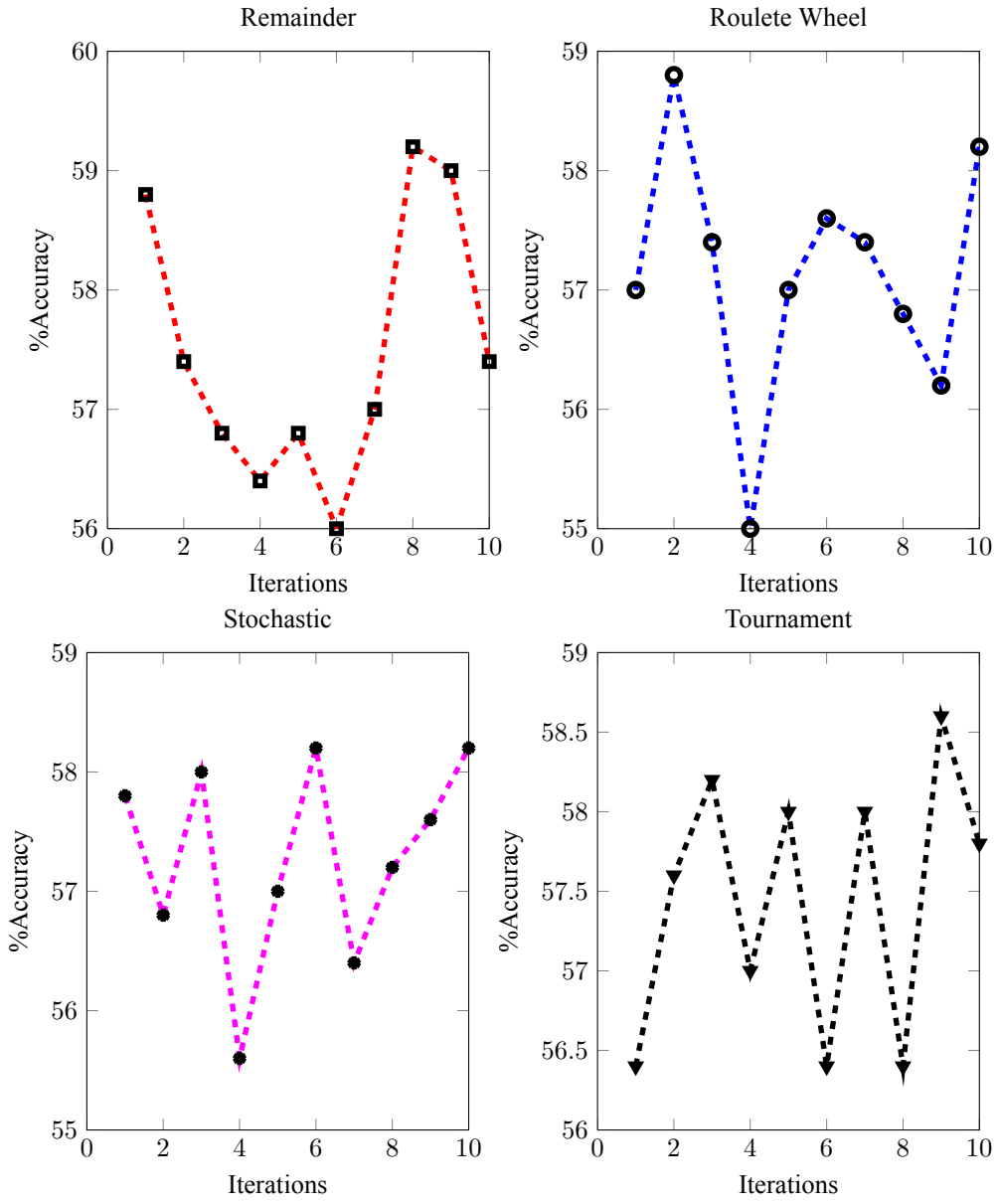
62.5	Remainder
62.35	Roulette
62.775	Stochastic
62.5	Tournament



Σχήμα 6.34: C1vsC2vsC3vsC4 Selection Function

Πίνακας 6.21: C1vsC2vsC3vsC4vsC5 Selection Function

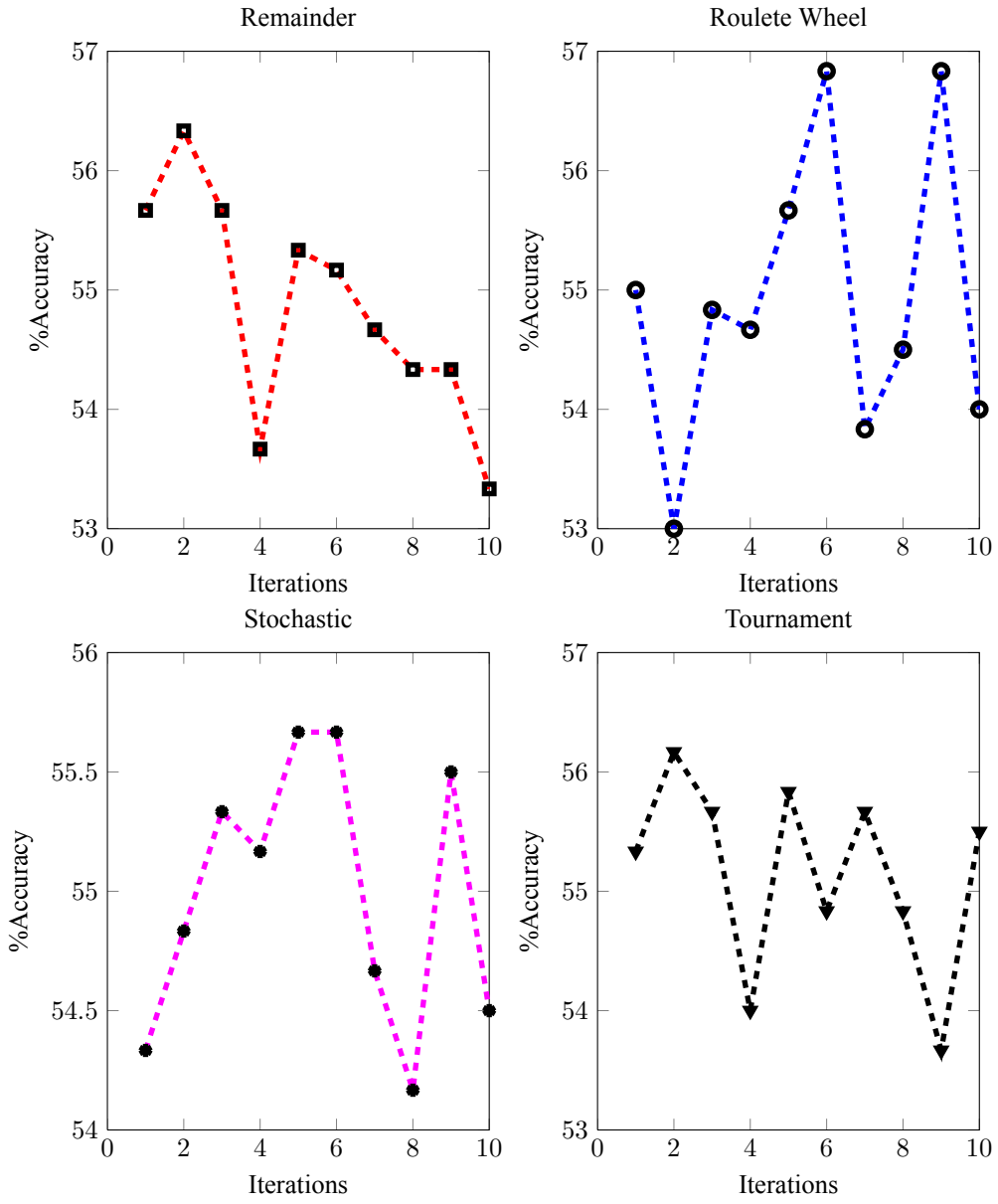
57.48	Remainder
57.14	Roulette
57.28	Stochastic
57.44	Tournament



Σχήμα 6.35: C1vsC2vsC3vsC4vsC5 Selection Function

Πίνακας 6.22: C1vsC2vsC3vsC4vsC5vsC6 Selection Function

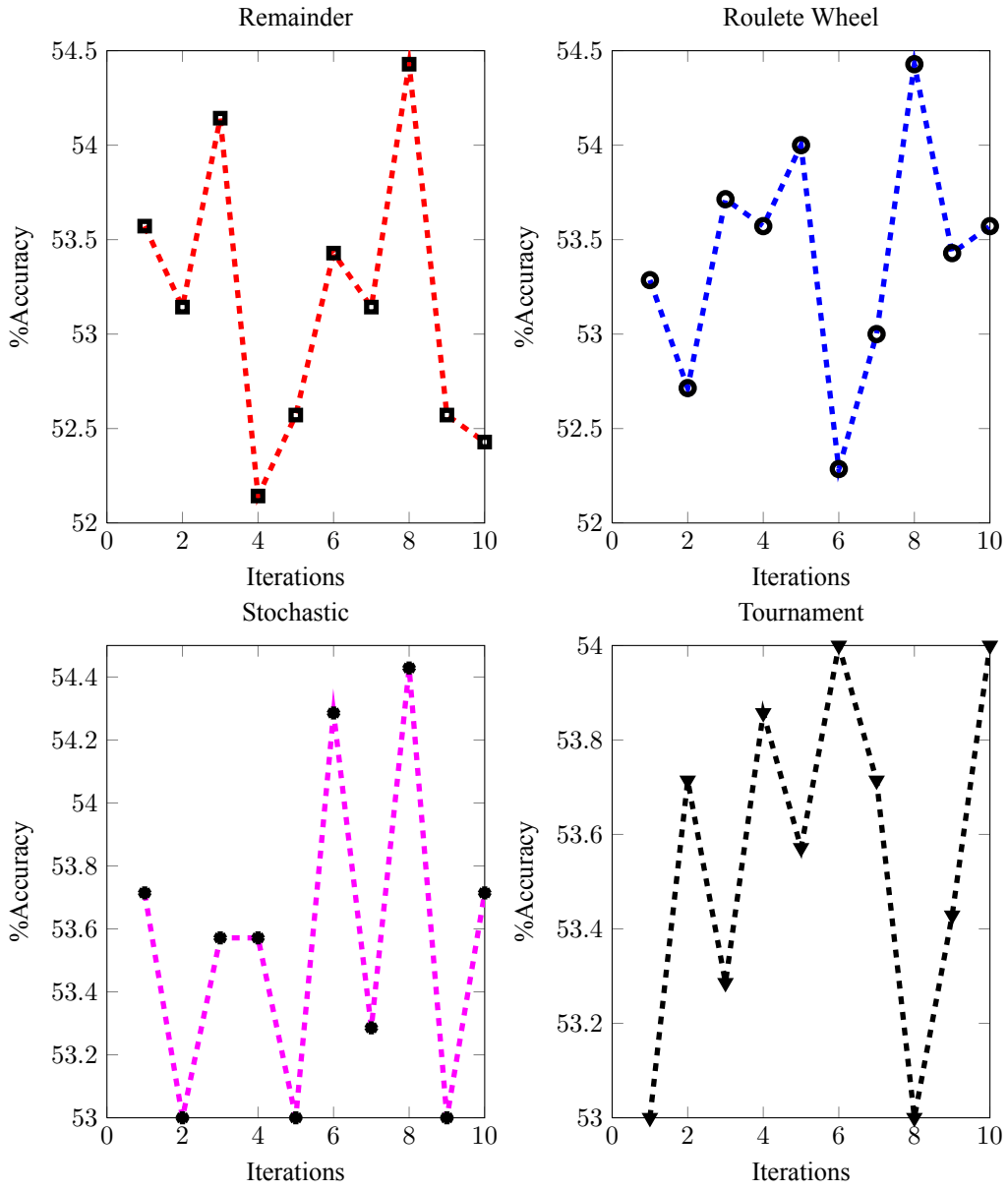
54.85	Remainder
54.91667	Roulette
54.98333	Stochastic
55.15	Tournament



Σχήμα 6.36: C1vsC2vsC3vsC4vsC5vsC6 Selection Function

Πίνακας 6.23: C1vsC2vsC3vsC4vsC5vsC6vsC7 Selection Function

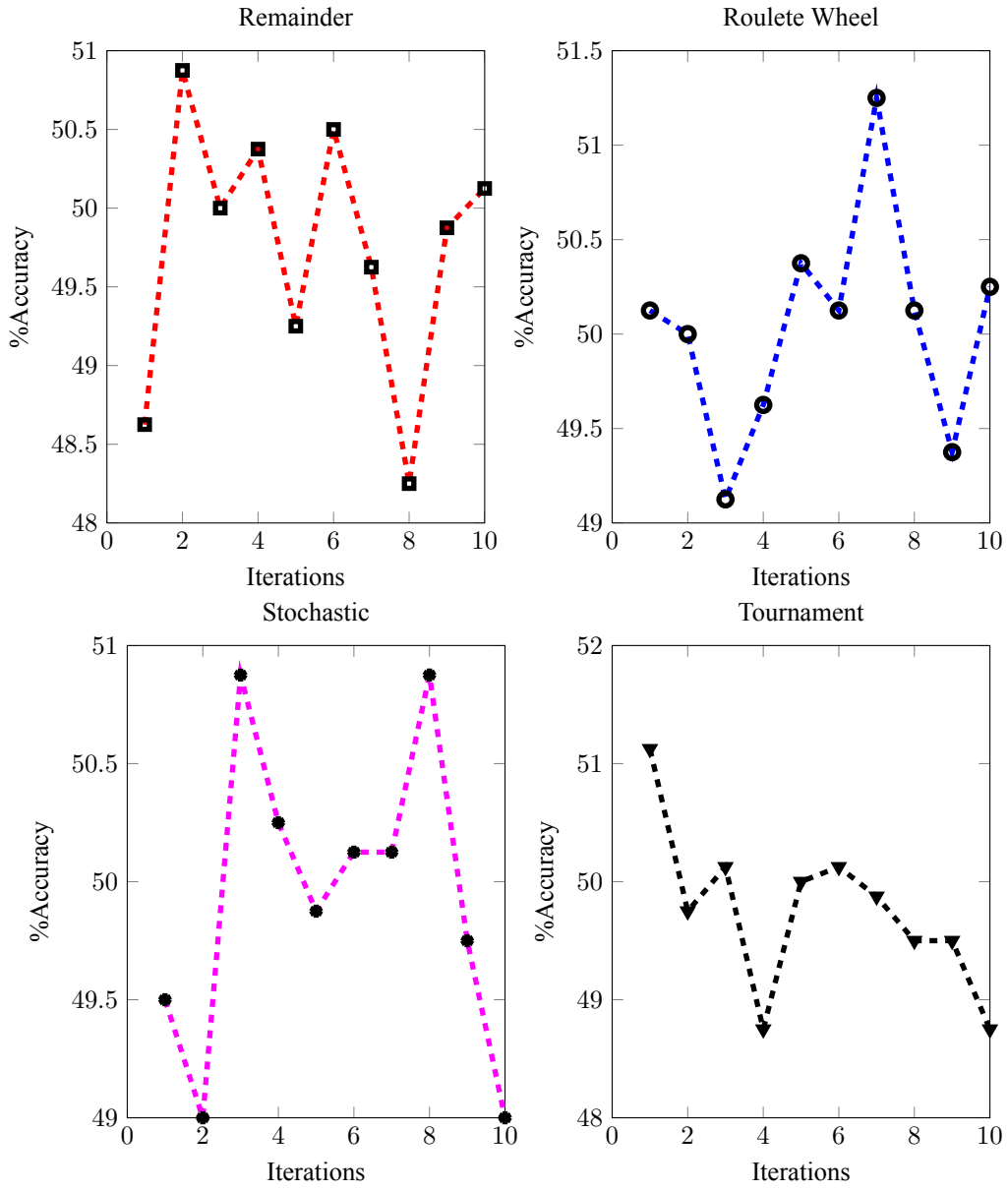
53.15714	Remainder
53.4	Roulette
53.55714	Stochastic
53.55714	Tournament



Σχήμα 6.37: C1vsC2vsC3vsC4vsC5vsC6vsC7 Selection Function

Πίνακας 6.24: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8 Selection Function

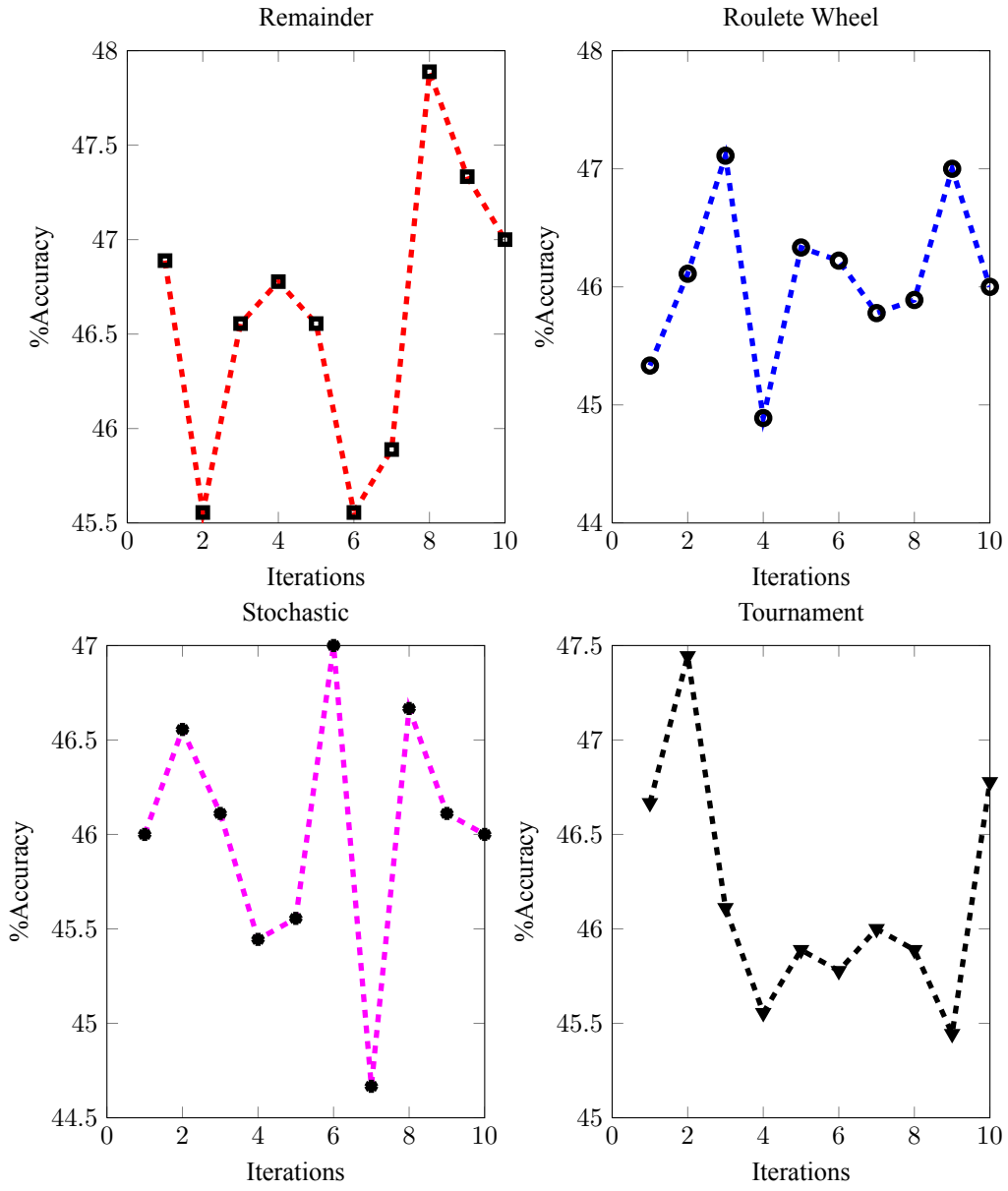
49.75	Remainder
50.0375	Roulette
49.9375	Stochastic
49.75	Tournament



Σχήμα 6.38: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8 Selection Function

Πίνακας 6.25: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9 Selection Function

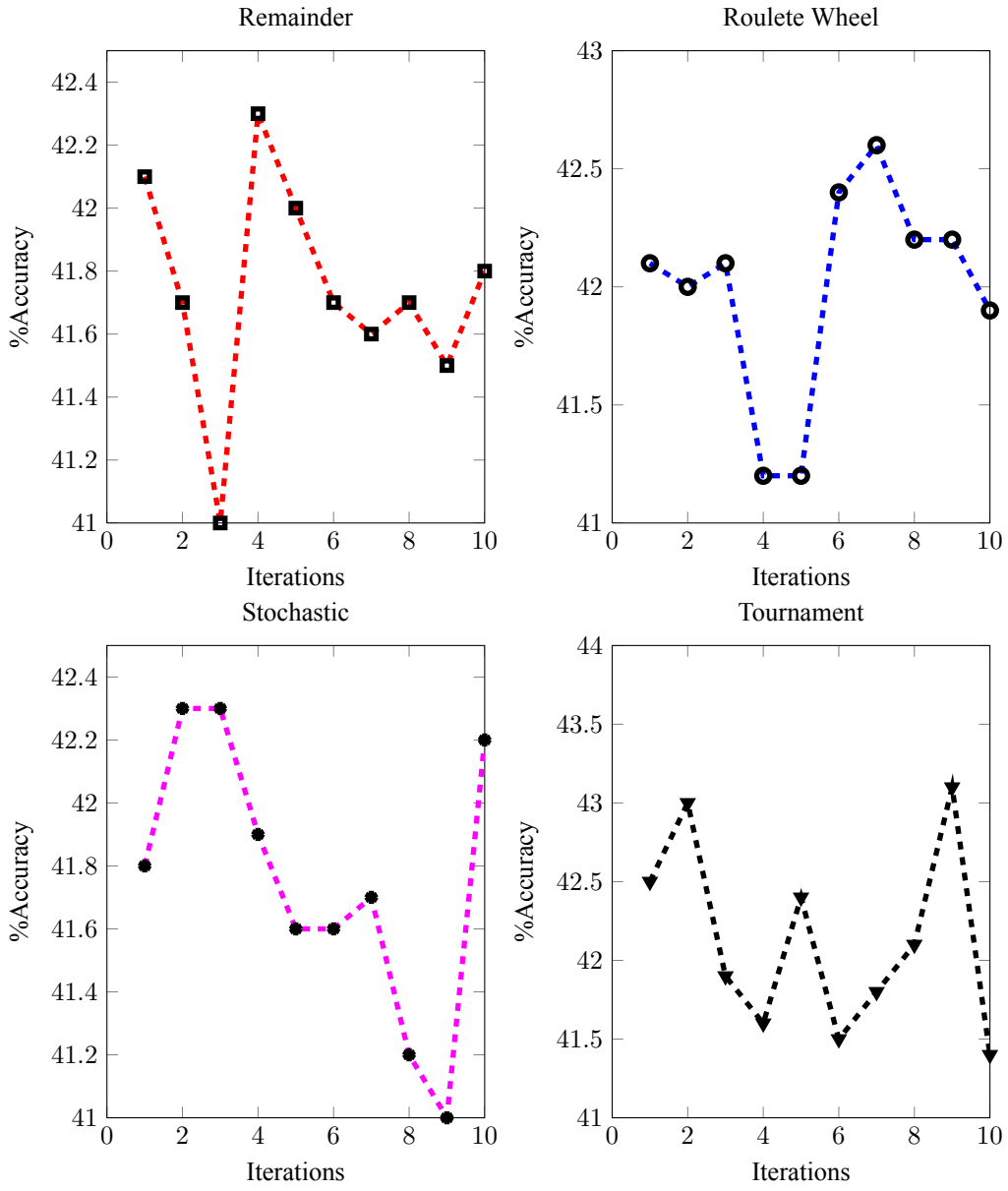
46.6	Remainder
46.06667	Roulette
46.01111	Stochastic
46.15556	Tournament



Σχήμα 6.39: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9 Selection Function

Πίνακας 6.26: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10 Selection Function

41.74	Remainder
41.99	Roulette
41.76	Default
41.83	Tournament



Σχήμα 6.40: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10 Selection Function

6.12 Crossover Function

Για την επιλογή της κατάλληλης συνάρτησης Crossover Function πραγματοποιούμε 10 πειράματα 10 cross-validation. Τα πειράματα διεξάγονται με τα καλύτερα αποτελέσματα των παραμέτρων και με την καλύτερη συνάρτηση Selection Function που παρουσιάστηκε στα προηγούμενα πειράματα. Με αυτόν το τρόπο, στο σύνολο Prima Indians η καλύτερη απόδοση πραγματοποιείται με την Crossover συνάρτηση $Scattered = 74.40$ όπως παρουσιάζεται στον πίνακα (6.27). Για το σύνολο Iris Data επιλέγουμε τη συνάρτηση $TwoPoint = 95.6$ (6.28), ενώ για το σύνολο Sonar $Scattered = 63.52$ (6.29). Τέλος για τα σύνολα Music Data έχουμε την εξής κατάσταση:

- **C1vsC2** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 91.65$ πίνακας (6.30).
- **C1vsC2vsC3** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 67.7$ πίνακας (6.31).
- **C1vsC2vsC3vsC4** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 62.77$ πίνακας (6.32).
- **C1vsC2vsC3vsC4vsC5** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 57.48$ πίνακας (6.33).
- **C1vsC2vsC3vsC4vsC5vsC6** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 55.15$ πίνακας (6.34).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 53.55$ πίνακας (6.35).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8** Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 50.03$ πίνακας (6.36).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9**

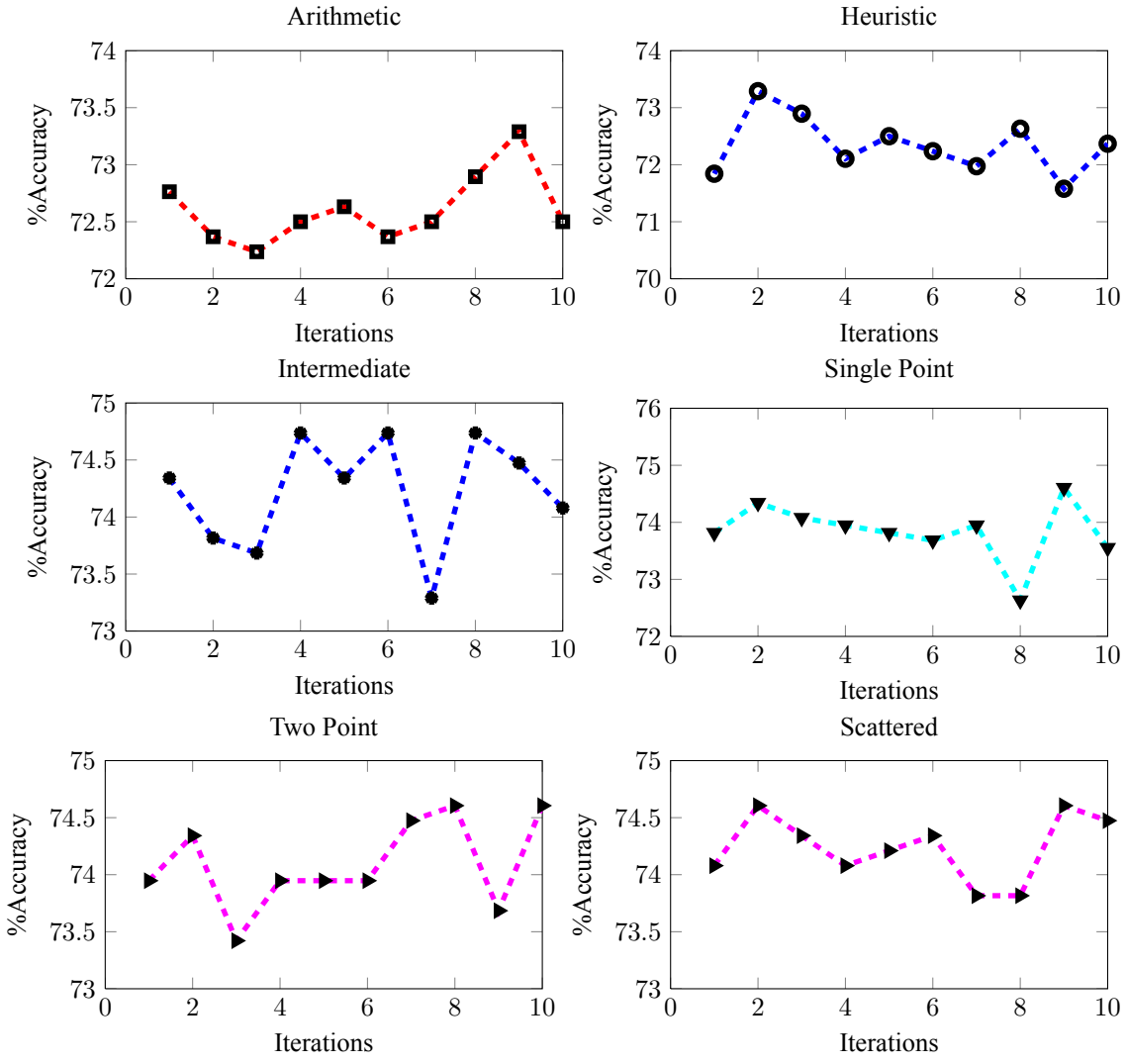
  Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 46.6$ πίνακας (6.37).
- **C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10**

  Η καλύτερη απόδοση πραγματοποιείται με την συνάρτηση $Scattered = 41.99$ πίνακας (6.26).

Παρατηρούμε ότι η καλύτερη Crossover συνάρτηση σε όλα τα σύνολα δεδομένων είναι η Scattered. Αυτό ήταν αναμενόμενο, αφού είναι η καλύτερη τεχνική για την διαδικασία Crossover.

Πίνακας 6.27: Prima Indians Crossover Function

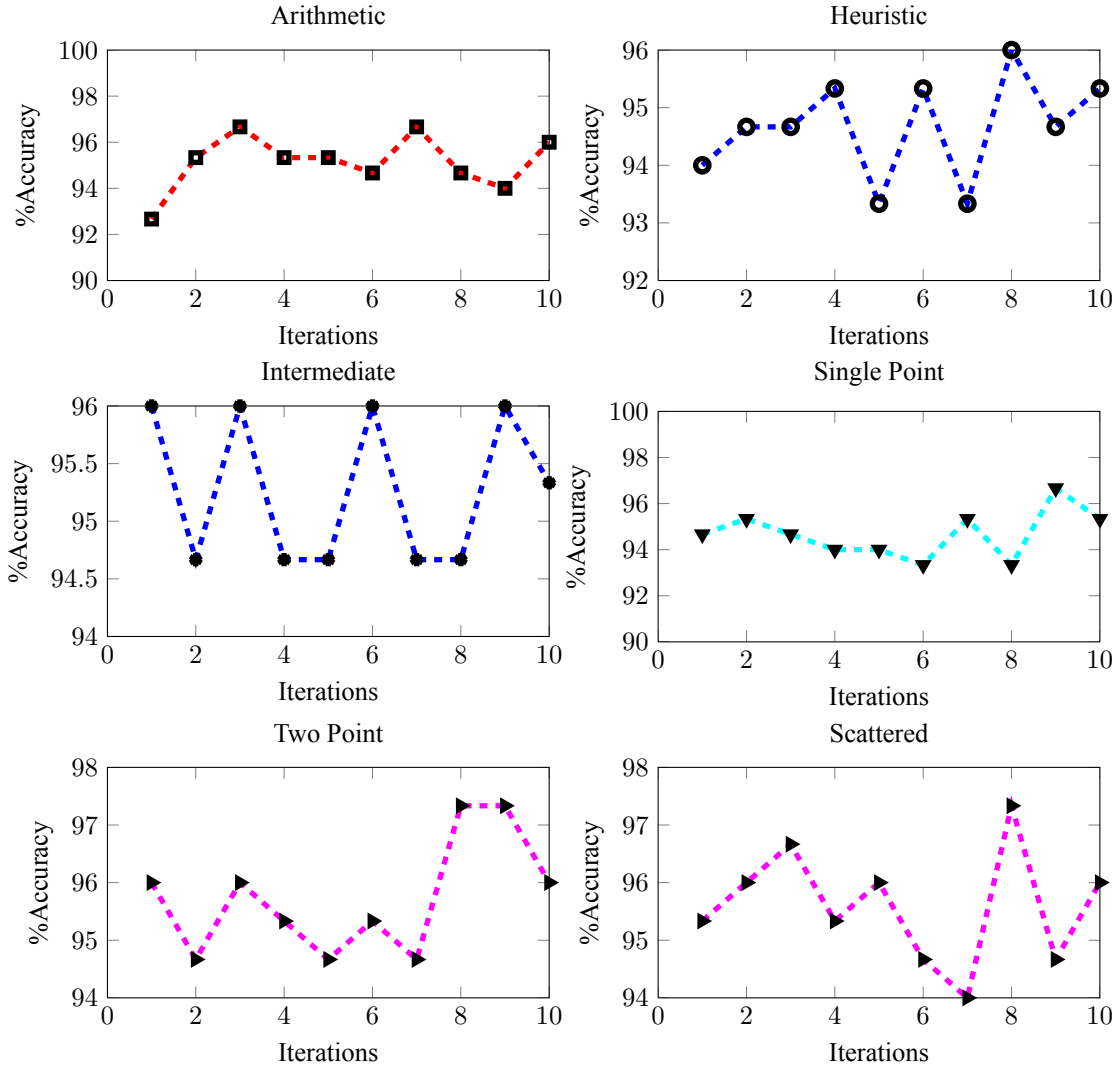
72.60526	Arithmetic
72.34211	Heuristic
74.22368	Intermediate
73.84211	Single Point
74.09211	Two Point
74.43684	Scattered



Σχήμα 6.41: Prima Indians Crossover Function

Πίνακας 6.28: Iris Data Crossover Function

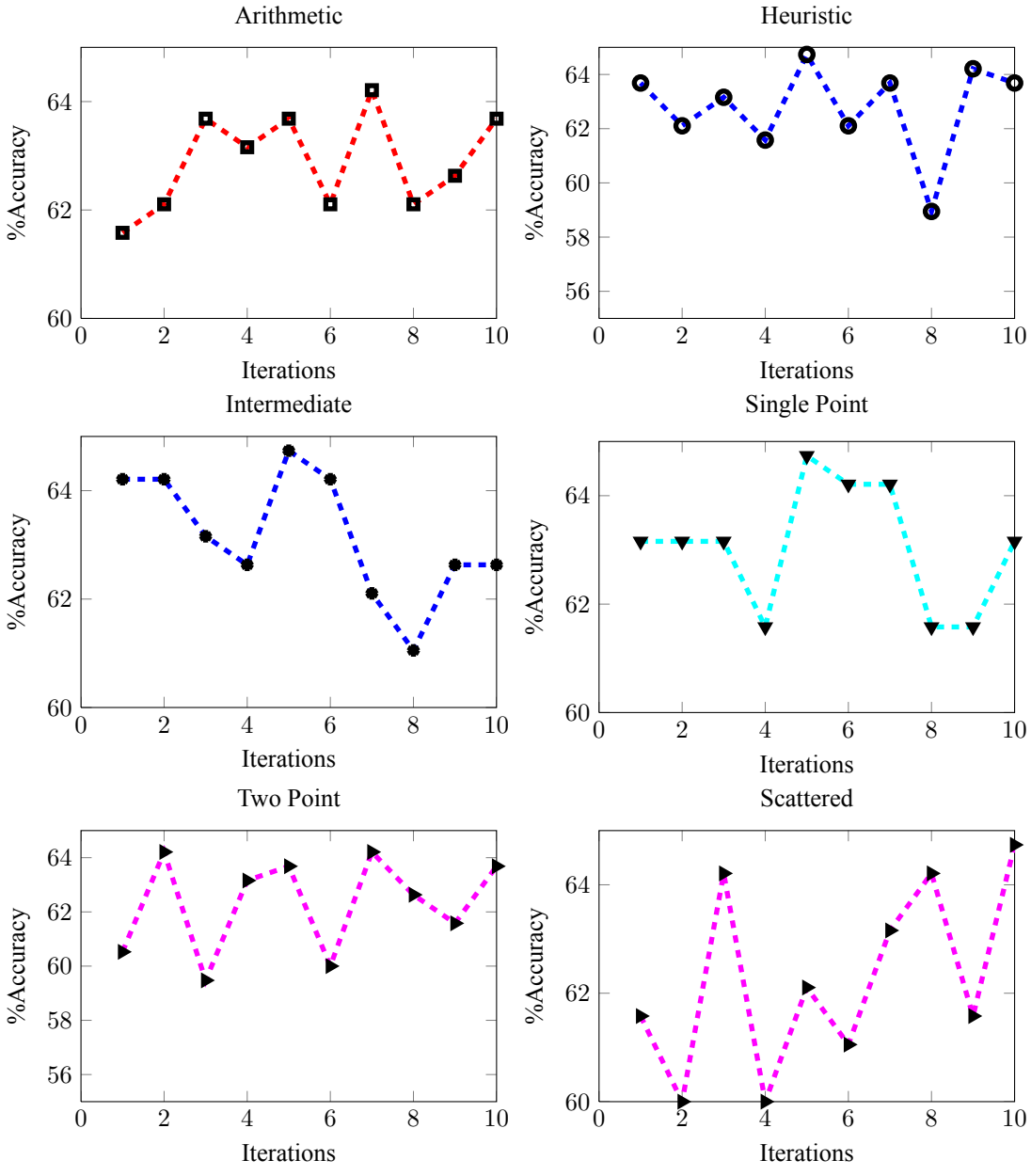
95.13333	Arithmetic
94.66667	Heuristic
95.26667	Intermediate
94.66667	Single Point
95.73333	Two Point
95.6	Scattered



Σχήμα 6.42: Iris Data Crossover Function

Πίνακας 6.29: Sonar Data Crossover Function

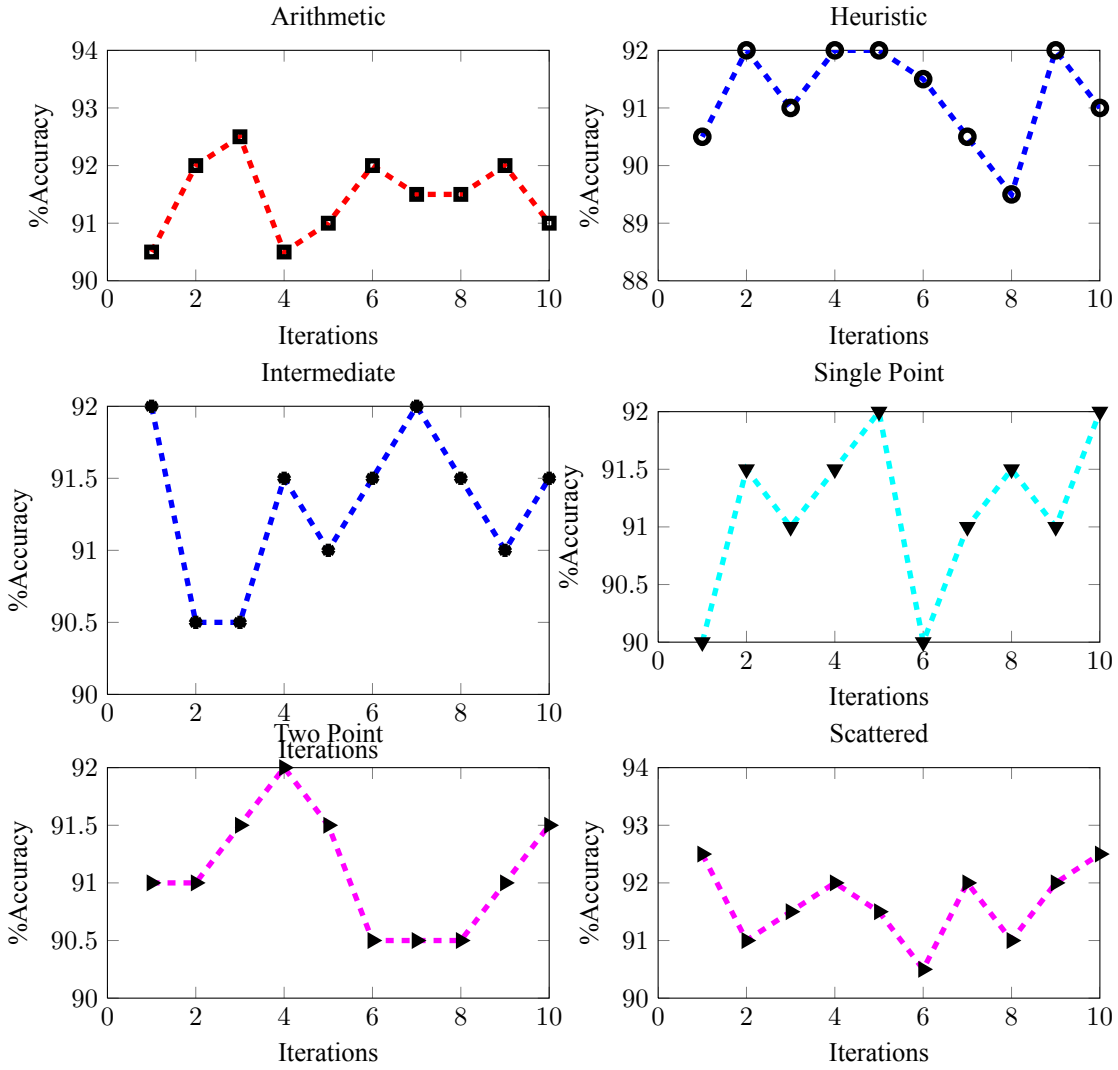
62.89473684	Arithmetic
62.78947368	Heuristic
63.15789474	Intermediate
63.05263158	Single Point
62.31578947	Two Point
63.52315789	Scattered



Σχήμα 6.43: Sonar Data Crossover Function

Πίνακας 6.30: C1vsC2 Crossover Function

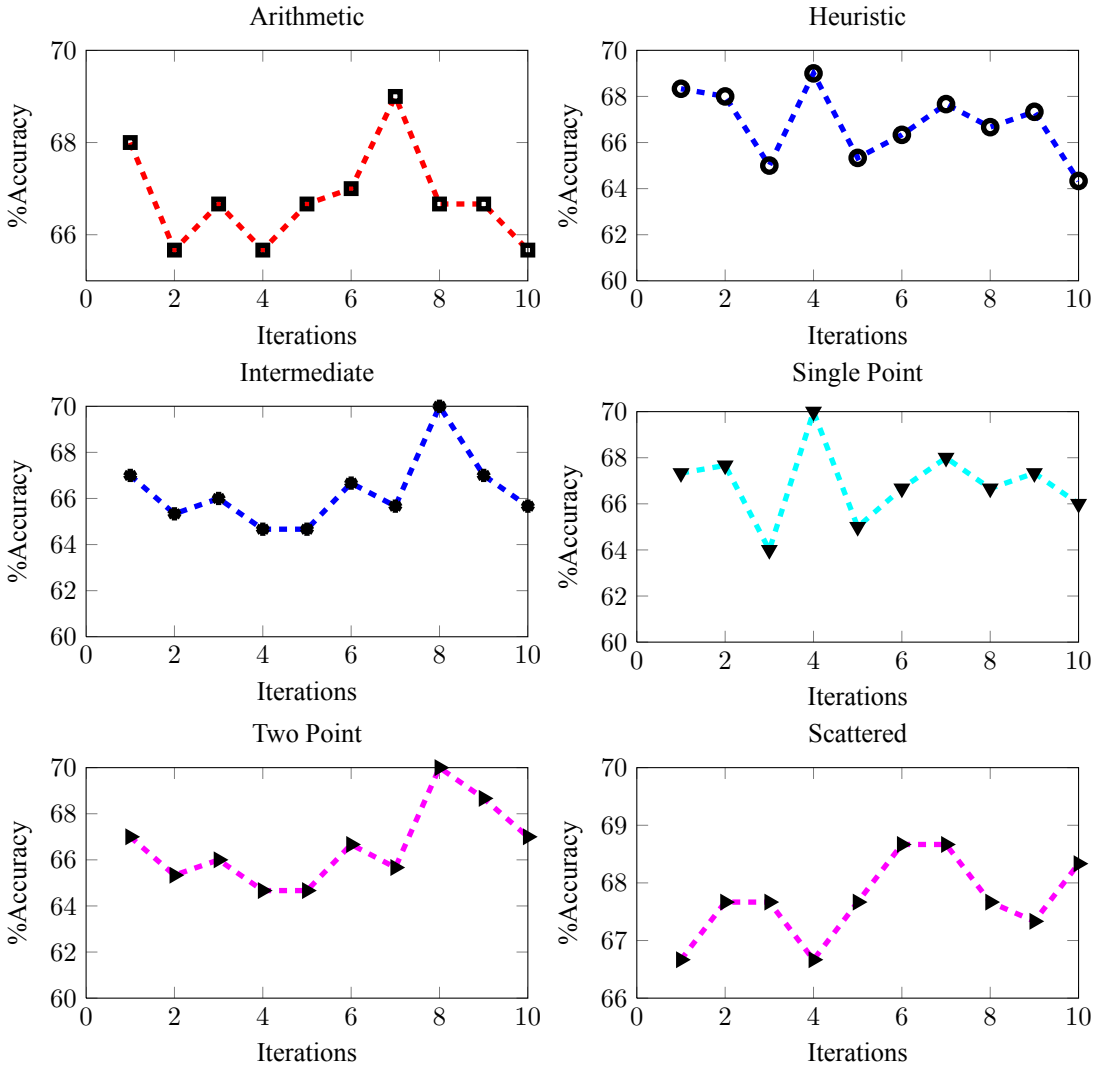
91.45	Arithmetic
91.2	Heuristic
91.3	Intermediate
91.15	Single Point
91.1	Two Point
91.65	Scattered



Σχήμα 6.44: C1vsC2 Crossover Function

Πίνακας 6.31: C1vsC2vsC3 Crossover Function

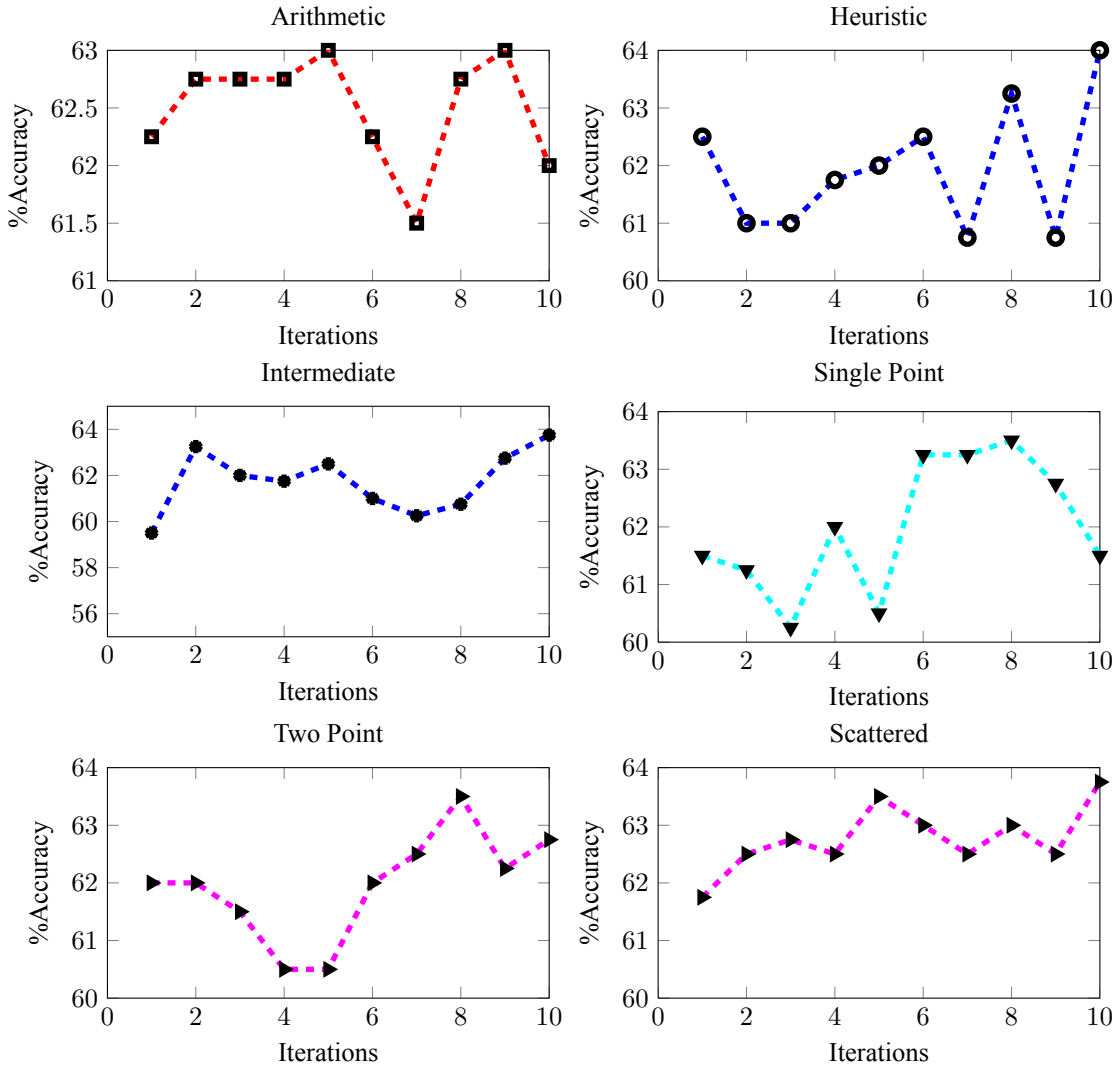
66.7666667	Arithmetic
66.8	Heuristic
66.2666667	Intermediate
66.8666667	Single Point
66.5666667	Two Point
67.7	Scattered



Σχήμα 6.45: C1vsC2vsC3 Crossover Function

Πίνακας 6.32: C1vsC2vsC3vsC4 Crossover Function

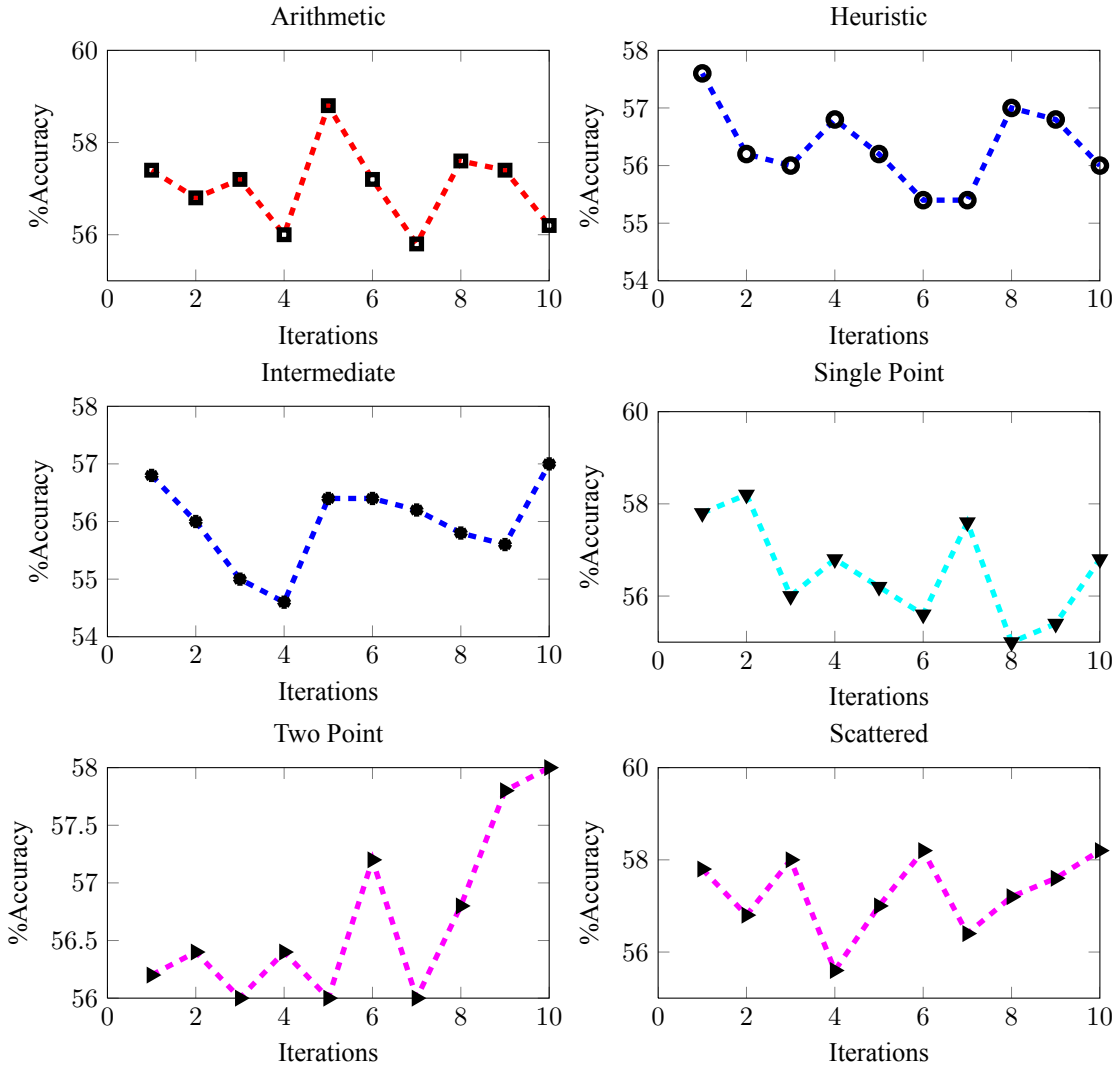
62.5	Arithmetic
61.95	Heuristic
61.75	Intermediate
61.975	Single Point
61.95	Two Point
62.775	Scattered



Σχήμα 6.46: C1vsC2vsC3vsC4 Crossover Function

Πίνακας 6.33: C1vsC2vsC3vsC4vsC5 Crossover Function

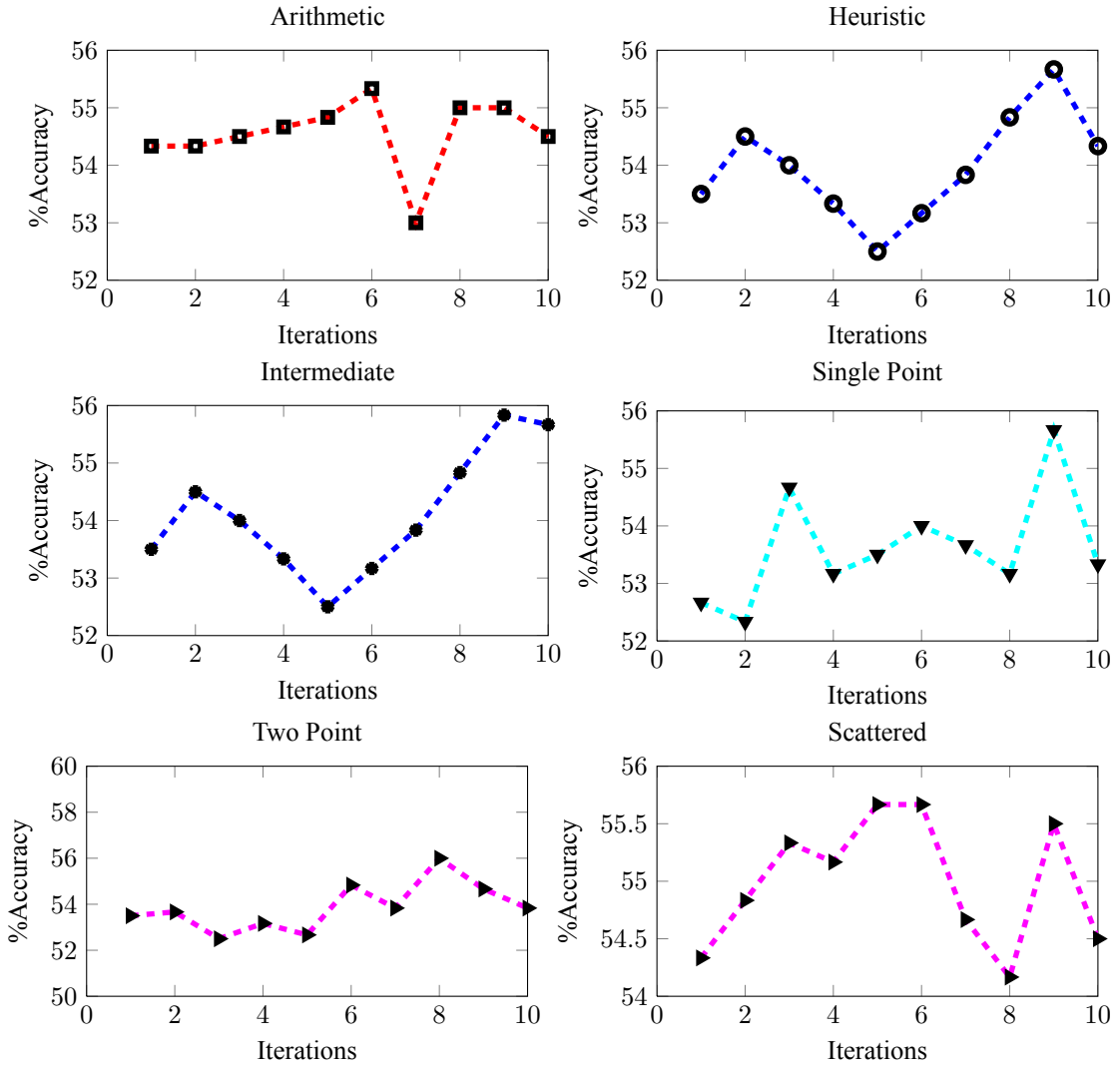
57.04	Arithmetic
56.34	Heuristic
55.98	Intermediate
56.54	Single Point
56.68	Two Point
57.28	Scattered



Σχήμα 6.47: C1vsC2vsC3vsC4vsC5 Crossover Function

Πίνακας 6.34: C1vsC2vsC3vsC4vsC5vsC6 Crossover Function

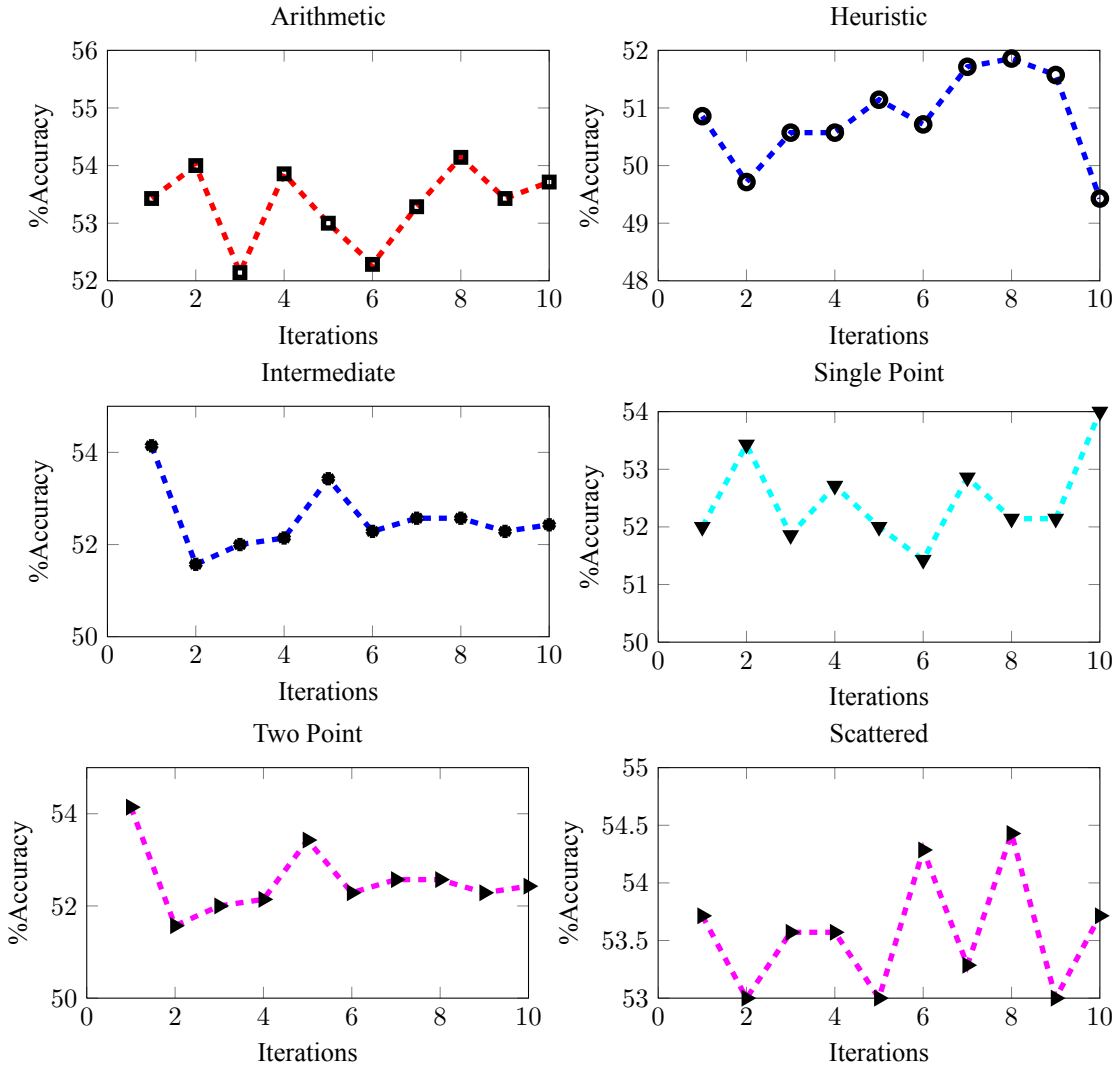
54.55	Arithmetic
53.96667	Heuristic
54.11667	Intermediate
53.61667	Single Point
53.86667	Two Point
55.15333	Scattered



Σχήμα 6.48: C1vsC2vsC3vsC4vsC5vsC6 Crossover Function

Πίνακας 6.35: C1vsC2vsC3vsC4vsC5vsC6vsC7 Crossover Function

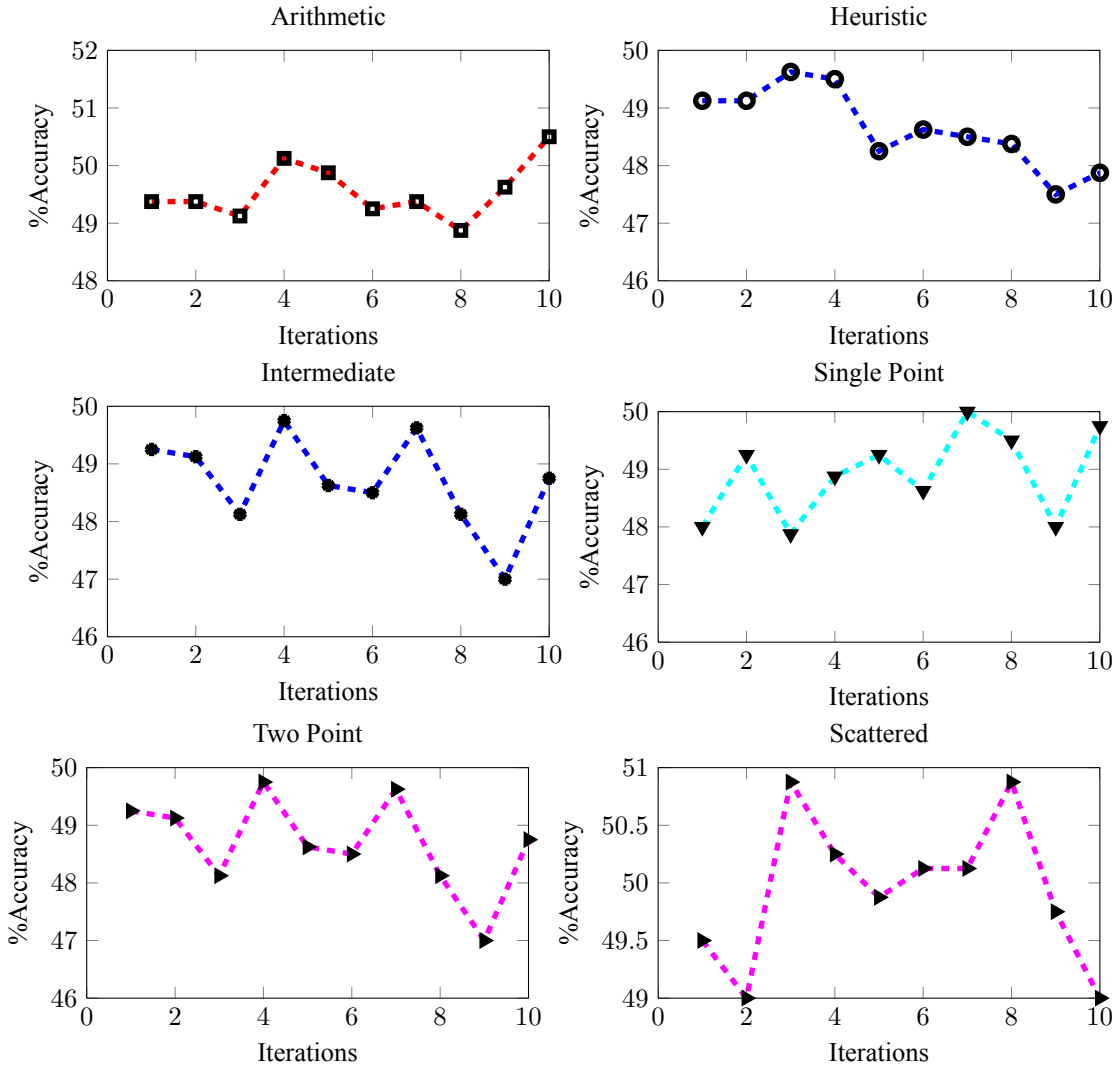
53.32857	Arithmetic
50.81429	Heuristic
52.54286	Intermediate
52.45714	Single Point
52.54286	Two Point
53.55714	Scattered



Σχήμα 6.49: C1vsC2vsC3vsC4vsC5vsC6vsC7 Crossover Function

Πίνακας 6.36: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8 Crossover Function

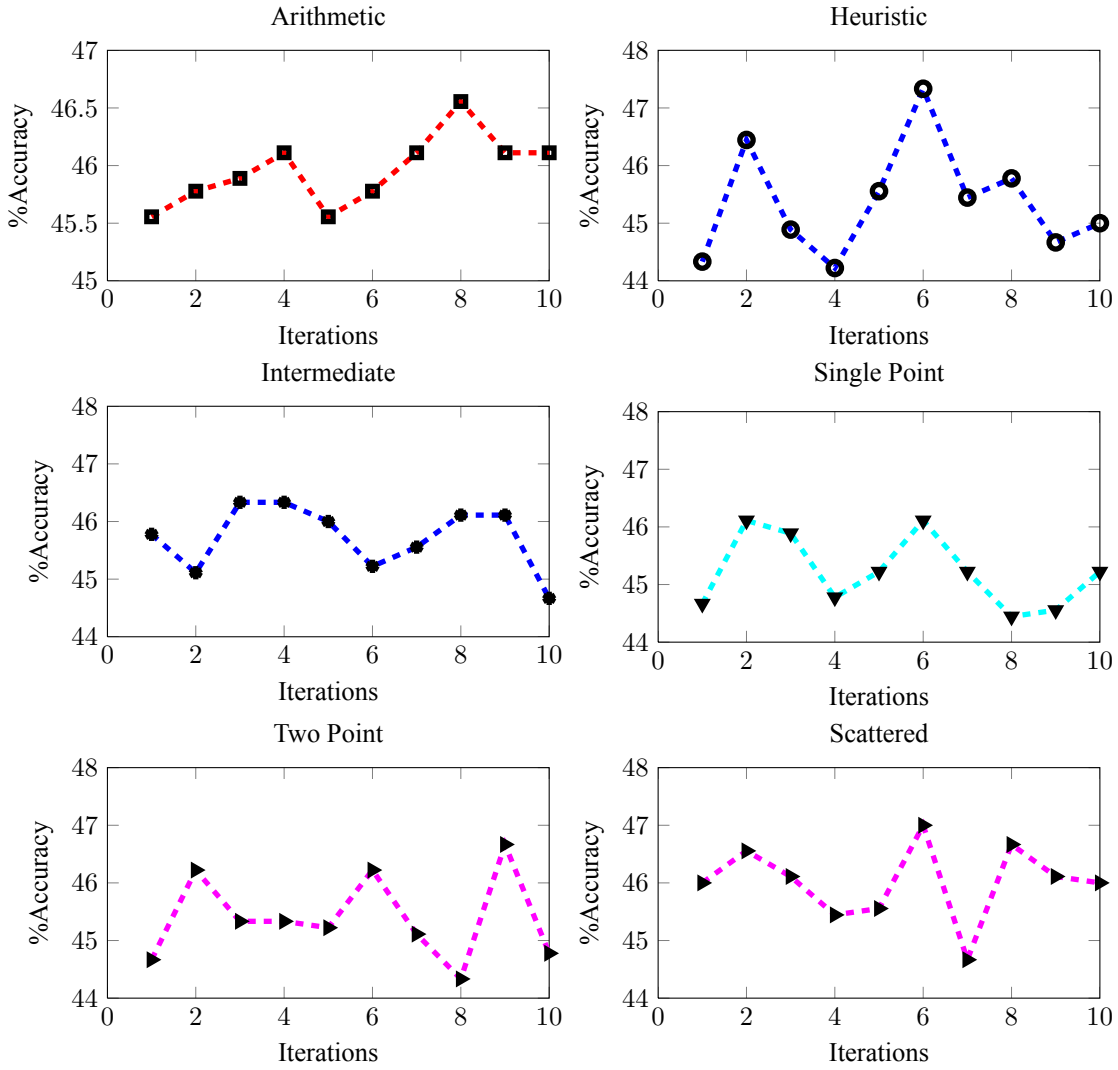
49.55	Arithmetic
48.65	Heuristic
48.6875	Intermediate
48.9125	Single Point
48.6875	Two Point
50.0375	Scattered



Σχήμα 6.50: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8 Crossover Function

Πίνακας 6.37: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9 Crossover Function

45.95556	Arithmetic
45.36667	Heuristic
45.72222	Intermediate
45.22222	Single Point
45.38889	Two Point
46.69111	Scattered



Σχήμα 6.51: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9 Crossover Function

6.13 Τελικά Αποτελέσματα

Τα τελικά αποτελέσματα για κάθε σύνολο δεδομένων καθώς και οι παράμετροι που χρησιμοποιούνται για την επίτευξη αυτών των τιμών παρατίθενται στους πιο κάτω πίνακες.

Πίνακας 6.38: Iris Data Τελικά Αποτελέσματα

Variables	Values
Population Size	160
Emax	0.04
Migration Fraction	0.6
Migration Interval	50
K	5
Crossover Fraction	1
Elite Count	5
Initial Population	Random
Detectors	80%
Creation Function	Linear Feasible
Selection Function	Roulette Wheel
Crossover Function	Scattered
Mutation Function	Adapt Feasible
Test Method	10 Cross Validation
Final Accuracy	96.4

Πίνακας 6.39: Prima Indians Data Τελικά Αποτελέσματα

Variables	Values
Population Size	140
Emax	0.06
Migration Fraction	0.5
Migration Interval	12
K	11
Crossover Fraction	0.6
Elite Count	8
Initial Population	$(mean - 0.05, mean + 0.05)$
Detectors	100%
Creation Function	Custom (Mean Function)
Selection Function	Roulette Wheel
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	74.4

Πίνακας 6.40: Sonar Data Τελικά Αποτελέσματα

Variables	Values
Population Size	180
Emax	0.16
Migration Fraction	1
Migration Interval	20
K	2
Crossover Fraction	0
Elite Count	8
Initial Population	$(mean - 0.05, mean + 0.05)$
Detectors	100%
Creation Function	Custom (Mean Function)
Selection Function	Remainder Wheel
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	63.52

Πίνακας 6.41: C1vsC2 Τελικά Αποτελέσματα

Variables	Values
Population Size	90
Emax	0.26
Migration Fraction	0.9
Migration Interval	48
K	5
Crossover Fraction	0.8
Elite Count	4
Initial Population	$(mean - 0.001, mean + 0.001)$
Detectors	80%
Creation Function	Custom (Mean Function)
Selection Function	Stochastic
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	92

Πίνακας 6.42: C1vsC2vsC3

Variables	Values
Population Size	80
Emax	0.23
Migration Fraction	0.6
Migration Interval	72
K	9
Crossover Fraction	0.3
Elite Count	5
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	80%
Creation Function	Custom
Selection Function	Stochastic
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	67.7

Πίνακας 6.43: C1vsC2vsC3vsC4

Variables	Values
Population Size	40
Emax	0.18
Migration Fraction	0.6
Migration Interval	24
K	9
Crossover Fraction	0.8
Elite Count	3
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	80%
Creation Function	Custom
Selection Function	Stochastic
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	62.77

Πίνακας 6.44: C1vsC2vsC3vsC4vsC5

Variables	Values
Population Size	80
Emax	0.1
Migration Fraction	0.7
Migration Interval	15
K	3
Crossover Fraction	0.9
Elite Count	5
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	80%
Creation Function	Custom
Selection Function	Remainder
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	57.48

Πίνακας 6.45: C1vsC2vsC3vsC4vsC5vsC6

Variables	Values
Population Size	50
Emax	0.1
Migration Fraction	0.6
Migration Interval	32
K	6
Crossover Fraction	0.9
Elite Count	1
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	90%
Creation Function	Custom
Selection Function	Tournament
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	55.15

Πίνακας 6.46: C1vsC2vsC3vsC4vsC5vsC6vsC7

Variables	Values
Population Size	50
Emax	0.07
Migration Fraction	0.2
Migration Interval	56
K	5
Crossover Fraction	0.9
Elite Count	5
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	90%
Creation Function	Custom
Selection Function	Tournament
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	53.55

Πίνακας 6.47: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8

Variables	Values
Population Size	130
Emax	0.21
Migration Fraction	0.1
Migration Interval	24
K	6
Crossover Fraction	0.4
Elite Count	1
Initial Population	$(mean - 0.08, mean + 0.08)$
Detectors	100%
Creation Function	Custom
Selection Function	Roulette
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	50.03

Πίνακας 6.48: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9

Variables	Values
Population Size	130
Emax	0.1
Migration Fraction	0.9
Migration Interval	80
K	4
Crossover Fraction	0.7
Elite Count	4
Initial Population	$(mean - 0.05, mean + 0.05)$
Detectors	90%
Creation Function	Custom
Selection Function	Roulette
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	46.6

Πίνακας 6.49: C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10

Variables	Values
Population Size	80
Emax	0.1
Migration Fraction	0.7
Migration Interval	15
K	3
Crossover Fraction	1
Elite Count	6
Initial Population	$(mean - 0.05, mean + 0.05)$
Detectors	80%
Creation Function	Custom
Selection Function	Roulette
Crossover Function	Scattered
Mutation Function	Default
Test Method	10 Cross Validation
Final Accuracy	41.99

6.14 Συγκριτικά Αποτελέσματα

Στην παράγραφο αυτή παρουσιάζουμε τα τελικά αποτελέσματα του Genetic Airs σε σύγκριση με τον Airs2. Η σύγκριση έχει πραγματοποιηθεί σε σχέση με την ακρίβεια ταξινόμησης που παρουσιάζουν οι δύο αλγόριθμοι στα σύνολα δεδομένων που εξετάζουμε. Έχει υλοποιηθεί η διαδικασία 10-cross-validation και στου δύο αλγορίθμους ενώ η τελική ακρίβεια ταξινόμησης για κάθε σύνολο έχει προκύψει ως ο μέσος όρος των 10 επαναλήψεων. Την διεξαγωγή των πειραμάτων για τον αλγόριθμο Airs2 έχει πραγματοποιηθεί στο πρόγραμμα WEKA ενώ για τον αλγόριθμο Genetic Airs στο Matlab.

Οι κύριες διαφοροποιήσεις των δύο αλγορίθμων που σχετίζονται με την ακρίβεια ταξινόμησης είναι το ποσοστό του Data Reduction καθώς και ο αριθμός των Seeds του αλγορίθμου Airs2. Το Data Reduction είναι το ποσοστό του συνόλου εκπαίδευσης (Training Set) που εξαιρείται από την διαδικασία της εκπαίδευσης. Για παράδειγμα στον Genetic Airs όταν οι Detectors είναι το 80 % του Training Set τότε το ποσοστό Data Reduction είναι το 20 % επί του συνόλου Training Set. Το Data Reduction χρησιμοποιείται για την μείωση των δεδομένων προς εκπαίδευση άρα και την μείωση του χρόνου εκτέλεσης. Με αυτό τον τρόπο επιτυγχάνουμε τη δημιουργία ενός συνόλου ανίχνευσης, στο οποίο ένα αντικείμενο του, αναγνωρίζει πλήθος διαφορετικών αντικειμένων. Ο αλγόριθμος Airs2 πραγματοποιεί αυτόματα ένα Data Reduction στο σύνολο δεδομένων που εκπαιδεύει και τις περισσότερες φορές κυμαίνεται στο διάστημα $[0 - 10\%]$. Στην πρώτη στήλη του πίνακα 6.50 παρουσιάζονται τα αποτελέσματα του Airs με Data Reduction $[0 - 10\%]$ και για αυτό το λόγο τα αποτελέσματα είναι από 2% ως και 15% καλύτερα από τα αποτελέσματα του Genetic Airs. Τα αποτελέσματα του Genetic Airs από την άλλη έχουν γίνει με ένα Data Reduction από 10% ως και 20%. Για αυτό τον λόγο πραγματοποιούμε ένα Data Reduction στο σύνολο Airs2 της τάξης του $[10 - 20\%]$ στη στήλη 3 του πίνακα 6.50 έτσι ώστε τα συγκριτικά αποτελέσματα να είναι πιο αξιόπιστα.

Η παράμετρος Seed του αλγορίθμου Airs είναι η τροφοδότηση του αρχικού συνόλου αντισωμάτων μνήμης με στοιχεία από το σύνολο εκπαίδευσης. Το Seed λαμβάνει ακέραιες τιμές δηλαδή το πλήθος των στοιχείων από το σύνολο εκπαίδευσης που θα αρχικοποιήσουν τον αλγόριθμο. Στην ουσία η παράμετρος αυτή είναι ένα σημείο αναφοράς για την εκπαίδευση του συνόλου των ανιχνευτών και έχει ως στόχο την αρχικοποίηση του αλγορίθμου ως προς ένα σύνολο. Όταν η τιμή της παραμέτρου Seed είναι μεγαλύτερη από 0, έστω m τότε υπάρχουν m ανιχνευτές οι οποίοι ταυτίζονται με κάποια m στοιχεία του συνόλου εκπαίδευσης (αντιγόνα). Με αυτόν τον τρόπο ο αλγόριθμος Airs αποκτά ένα προβάδισμα γιατί εκτός ότι εκπαιδεύει m λιγότερους ανιχνευτές (αντισώματα), δημιουργείται μία κατανομή που έχει ως βάση m "κατάλληλα" δεδομένα.

Από την άλλη πλευρά, στον αλγόριθμο Genetic Airs τον ρόλο του Seed τον έχει το Initial Population. Το Initial Population είναι ο αρχικός πληθυσμός του Γενετικού Αλγορίθμου. Ο πληθυσμός αυτός είναι στοιχεία (χρωμοσώματα) τα οποία είναι ϵ-κοντά στην μέση τιμή των δεδομένων του συνόλου εκπαίδευσης. Με τον τρόπο αυτό εκπαιδεύουμε τους ανιχνευτές έχοντας ως σημείο αναφοράς τον μέσο όρο όλων των δεδομένων και όχι κάποιων συγκεκριμένων.

6.14.1 Iris Data

Για το πρώτο σύνολο δεδομένων Iris Data παρατηρούμε ότι ο αλγόριθμος Airs2 χωρίς το Data Reduction παρουσιάζει καλύτερη ταξινομική ακρίβεια 96.67 αντί του 96.4, δηλαδή στην ουσία αναγνωρίζει ένα στοιχείο περισσότερο από τον Genetic Airs. Από την άλλη πλευρά ο Airs2 με Data Reduction της τάξης 10-20% παρουσιάζει μικρότερη ακρίβεια ταξινόμησης 95 αντί 96.4. Συμπερασματικά παρατηρούμε ότι η ακρίβεια ταξινόμησης με τους δύο αλγορίθμους είναι σχεδόν ίδια και δεν παρουσιάζει μεγάλη διαφοροποίηση.

6.14.2 Prima Indians

Στο σύνολο Prima Indians παρατηρούμε ότι ο Genetic Airs πετυχαίνει την καλύτερη ακρίβεια 74.4% και από τον Airs2 και από τον Airs2 με Data Reduction. Πιο συγκεκριμένα, ο Genetic Airs ταξινομεί σωστά 565 από τα 700 δεδομένα που περιέχει το σύνολο Prima Indians. Γενικά το σύνολο Prima Indians είναι ένα δύσκολο σύνολο για ταξινόμηση και η καλύτερη ακρίβεια ταξινόμησης είναι η 75.5% που παρουσιάζει ο αλγόριθμος k-NN με k=22 και μετρική Manhatan. Το γεγονός αυτό κατατάσσει τον Genetic Airs μέσα στη καλύτερη 10-άδα των αλγορίθμων με καλύτερη ακρίβεια ταξινόμησης σε αυτό το σύνολο.

6.14.3 Sonar

Στο σύνολο Sonar ο αλγόριθμος Genetic Airs δεν είναι αποτελεσματικός αφού παρουσιάζει ακρίβεια ταξινόμησης 63.52% ενώ ο Airs2 83.99 και ο Airs2 με Data Reduction 66.01. Σε αυτό το σύνολο ο Genetic Airs ταξινομεί σωστά 120 από τα 190 δεδομένα ενώ συγκριτικά ο Airs2 160 από τα 190. Η συμπεριφορά αυτή του Genetic Airs μπορεί να οφείλεται στο γεγονός ότι το συγκεκριμένο σύνολο είναι καλύτερα να αντιμετωπιστεί τοπικά και όχι με σφαιρικό τρόπο.

6.14.4 Music Data

Στο σύνολο Music Data παρατηρούμε μία σταθερή συμπεριφορά των αλγορίθμων ως προς τις κατηγορίες $C_i vs C_j$ που έχουμε ορίσει. Στον πίνακα 6.50 βλέπουμε ότι ο Airs2 πετυχαίνει καλύτερη ακρίβεια σε κάθε κατηγορία $C_i vs C_j$ από τον Genetic Airs της τάξης του 2% ως 5%. Από την άλλη πλευρά, ο Airs για να πετύχει αυτή την ακρίβεια πραγματοποιεί Data Reduction της τάξης του $[0-4\%]$. Χρησιμοποιώντας όλο το σύνολο εκπαίδευσης (Training Set) είναι λογικό η ακρίβεια ταξινόμησης να είναι μεγαλύτερη από ότι του Genetic Airs που χρησιμοποιεί το 80% ή το 90% του συνόλου. Για αυτό το λόγο τα αποτελέσματα του Airs2 με Data Reduction $[10-20\%]$ παρουσιάζουν πολυ μικρότερη ακρίβεια από τον Genetic Airs. Χαρακτηριστικότερη διαφορά της ακρίβειας ταξινόμησης βρίσκεται στην κατηγορία σύγκρισης C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8 της τάξης του 12.4% δηλαδή 99 στοιχεία περισσότερα από τον Airs2 με Data Reduction $[10-20\%]$

Πίνακας 6.50: Συγκριτικά Αποτελέσματα

DataSet	Airs2	Genetic Airs Data Reduction 10-20%	Airs2 Data Reduction 10-20%
Iris Plant	96.67	96.4	95
Prima Indians Diabetes	74.08	74.4	70.872
Sonar	83.99	63.52	66.01
C1vsC2	94	92	87
C1vsC2vsC3	72.33	67.7	61.6
C1vsC2vsC3vsC4	66	62.775	52.5
C1vsC2vsC3vsC4vsC5	61	57.48	50.8
C1vsC2vsC3vsC4vsC5vsC6	57.66	55.15	45.16
C1vsC2vsC3vsC4vsC5vsC6vsC7	57.85	53.55	44.85
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8	51.62	50.03	37.62
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9	48.66	46.6	37.11
C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10	44.3	41.99	34.5

Κεφάλαιο 7

Συμπεράσματα

Στο κεφάλαιο αυτό παραθέτουμε τα συμπεράσματα για την συμπεριφορά και τα αποτελέσματα του υβριδικού αλγορίθμου Genetic Airs. Ο αλγόριθμος αυτός κατατάσσεται στον τομέα της μηχανικής μάθησης και πιο συγκεκριμένα στο πρόβλημα της ταξινόμησης. Ο αλγόριθμος συνδυάζει τεχνικές της θεωρίας των γενετικών αλγορίθμων και των τεχνητών ανοσοποιητικών συστημάτων. Εμπνευσμένοι από τον αλγόριθμο AIRS που ανήκει στην κατηγορία των τεχνητών ανοσοποιητικών συστημάτων έγινε μία προσπάθεια διερεύνησης των μηχανισμών του με τη βοήθεια των γενετικών αλγορίθμων.

Σκοπός του αλγορίθμου Genetic Airs είναι η εύρεση ενός αποδοτικού αλγορίθμου ταξινόμησης που θα είναι ανταγωνιστικός με άλλους αλγόριθμους ταξινόμησης. Είναι γεγονός ότι έχουν γίνει λιγοστές προσπάθειες για την εφαρμογή των γενετικών αλγορίθμων σε προβλήματα ταξινόμησης αυτό οφείλεται κυρίως στην φύση των γενετικών μηχανισμών για σφαιρική αναζήτηση.

Πλεονεκτήματα

- Χρησιμοποιεί λιγότερα στοιχεία του Training Set από τον αλγόριθμο Airs. Ο αλγόριθμος Genetic Airs χρησιμοποιεί [10-20%] του συνόλου εκπαίδευσης για την παραγωγή των ανιχνευτών ενώ από την άλλη μεριά ο Airs χρησιμοποιεί ως και το 100% του συνόλου εκπαίδευσης.

- Είναι αποτελεσματικός και παρουσιάζει καλά αποτελέσματα ανεξαρτήτως του μεγέθους του συνόλου δεδομένων. Παρατηρούμε ότι στο μεγαλύτερο σύνολο που εξετάζουμε αυτό του Music Data (C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10) ο αλγόριθμος παρουσιάζει καλά αποτελέσματα και υπερβαίνει την ακρίβεια ταξινόμησης του Airs.

- Δεν χρησιμοποιεί δεδομένα του συνόλου εκπαίδευσης για την δημιουργία των ανιχνευτών. Όπως είδαμε προηγουμένως ο αλγόριθμος Airs χρησιμοποιεί δεδομένα του συνόλου εκπαίδευσης για αρχικοποίηση. Αντιθέτως, ο Genetic Airs χρησιμοποιεί μια διαφορετική διαδικασία και δεν χρειάζεται δεδομένα του συνόλου, αλλά μια γενική πληροφορία, αυτή της μέσης τιμής των τιμών των δεδομένων για την δημιουργία του αρχικού πληθυσμού.

- Βασίζεται στη θεωρία των γενετικών αλγορίθμων που είναι κοινώς αποδεκτή.

Μειονεκτήματα

- Παρουσιάζει μεγαλύτερο χρόνο εκτέλεσης από τον Airs. Αυτό οφείλεται στο γεγονός ότι οι γενετικοί αλγόριθμοι είναι χρονοβόρα διαδικασία. Μία ακόμη αιτία είναι διότι η υλοποίηση του Airs Genetic έχει πραγματοποιηθεί στο Matlab που είναι μια γλώσσα αργή εκ φύσεως, σε σχέση με το Weka που έχει υλοποιηθεί σε Java.

- Η ακρίβεια ταξινόμησης σε σχέση με τον Airs χωρίς Data Reduction είναι μικρότερη.
- Η σφαιρική προσέγγιση του αλγορίθμου Genetic Airs είναι σε ορισμένα σύνολα μη αποτελεσματική. Παράδειγμα αποτελεί το σύνολο Sonar Data στο οποίο δεν παρουσιάζει καλά αποτελέσματα.

Ένα τελικό συμπέρασμα που προκύπτει από την σύγκριση αυτών των αλγορίθμων είναι το γεγονός, ότι ο Genetic Airs ανταγωνίζεται επαρκώς τον Airs παρουσιάζοντας θετικά αποτελέσματα εκεί που ο Airs δεν τα καταφέρνει. Με αυτό τον τρόπο μπορούμε να παρατηρήσουμε ότι ένας σφαιρικός τρόπος προσέγγισης των προβλημάτων ταξινόμησης μπορεί να φέρει θετικά αποτελέσματα καθώς και ότι οι Γενετικοί Αλγόριθμοι μπορούν να χρησιμοποιηθούν για τέτοιου είδους προβλήματα.

Στην παρούσα εργασία χρησιμοποιήθηκε η γλώσσα Matlab, η οποία μπορεί να υστερεί σε υπολογιστικούς χρόνους σε σχέση με άλλες γλώσσες αλλά έχει την ευχέρεια της ευκολότερης και γρηγορότερης υλοποίησης εξαιτίας των συναρτήσεων που παρέχει για πράξεις πινάκων καθώς και εργαλεία που διαθέτει για βελτιστοποίηση.

Τέλος, ο αλγόριθμος Airs Genetic αποτελεί μία πολύ αποτελεσματική μέθοδο ταξινόμησης, καθώς έχει μεγάλα ποσοστά επιτυχίας και συνεπώς πραγματοποιεί σωστές ταξινομήσεις. Ένα μειονέκτημά του είναι ο χρόνος εκτέλεσης, ο οποίος μπορεί να είναι πολύ μεγάλος σε μεγάλα και απαιτητικά σύνολα δεδομένων. Παρόλα αυτά, αυτό αντισταθμίζεται με τα πολύ καλά ποσοστά επιτυχίας του.

7.1 Μελλοντικές Επεκτάσεις

Ο αλγόριθμος που υλοποιήθηκε μπορεί να αποτελέσει τη βάση για αρκετές μελλοντικές επεκτάσεις. Πιο συγκεκριμένα, παρακάτω παρουσιάζονται ορισμένες επεκτάσεις και διαφοροποιήσεις που μπορούν να γίνουν με σκοπό τη διερεύνηση της συμπεριφοράς του αλγορίθμου. Αρχικά, θα ήταν επιθυμητή η διεξαγωγή πειραμάτων σε περισσότερα σύνολα δεδομένων έτσι ώστε να διερευνηθεί η συμπεριφορά του αλγορίθμου στα σύνολα αυτά. Στη συνέχεια θα μπορούσε να διερευνηθεί η προσθήκη ενός πίνακα βαρών όπως του αλγορίθμου GA-WKNN. Ο πίνακας αυτός, θα περιέχει βάρη ανάλογα με την βαρύτητα του χαρακτηριστικού στη διαδικασία της ταξινόμησης. Ένα επιπλέον χαρακτηριστικό που θα μπορούσε να προστεθεί στον αλγόριθμο Genetic Airs είναι μία συνάρτηση για την μεταφορά των δεδομένων μας σε μεγαλύτερες διαστάσεις. Με αυτόν τον τρόπο θα δημιουργηθούν διαφορετικές συσχετίσεις ανάμεσα στα δεδομένα μας και θα είναι δυνατόν να ταξινομηθούν καλύτερα. Ακολούθως ένα ακόμη σημείο έρευνας θα μπορούσε να είναι η προσθήκη μίας διαδικασίας μείωσης των χαρακτηριστικών των συνόλων δεδομένων. Τέλος, η διερεύνηση μείωσης του υπολογιστικού χρόνου του Genetic Airs θα ήταν καθοριστικής σημασίας, αφού ο υπολογιστικός χρόνος του αλγορίθμου αποτελεί ένα από τα βασικά του μειονεκτήματα.

Βιβλιογραφία

[1] Sanghamitra Bandyopadhyay and Sankar Kumar Pal. *Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 3540496068.

[2] R J Jr Bayardo, R Agrawal, and D Gunopulos. Constraint-based rule mining in large, dense databases, 1999. ISSN 10636382. URL http://www.springerlink.com/index/L17511T651567G82.pdf.

[3] Christopher M Bishop. Neural networks: a pattern recognition perspective. *Neural Networks*, (1973): 1–23, 1996. URL http://eprints.aston.ac.uk/1130/.

[4] C L Blake and C J Merz. UCI Repository of machine learning databases, 1998. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

[5] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. In *Machine Learning*, pages 95–122, 1996.

[6] Leandro Nunes De Castro and Fernando J Von Zuben. An Evolutionary Immune Network for Data Clustering. *Symposium A Quarterly Journal In Modern Foreign Literatures*, pages 84–89, 2000. doi: 10.1109/SBRN.2000.889718. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=889718.

[7] L N De Castro and F J Von Zuben. aiNet: An Artificial Immune Network for Data Analysis. In H A Abbass, R A Sarker, and C S Newton, editors, *Data Mining A Heuristic Approach*, chapter XII, pages 231–259. Idea Group Publishing, 2001. URL http://books.google.com/books?hl=en&lr=&id=vmV2B_bzyD8C&oi=fnd&pg=PA231&dq=aiNet:+An+Artificial+Immune+Network+for+Data+Analysis&ots=EAaJlAKhiY&sig=x3v9iPZUcpRWZ95AbN39p7UoJ8M.

[8] L.N de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. [[Springer Science+Business Media|Springer]], 2002. ISBN 1852335947, 9781852335946. URL http://en.wikipedia.org/wiki/Artificial_immune_system.

[9] L.N de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springe, Berlin, 2002.

[10] L.N. de Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *Evolutionary Computation, IEEE Transactions on*, 6(3):239–251. doi: 10.1109/TEVC.2002.1011539. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1011539.

[11] Thomas G Dietterich. Machine-Learning Research: Four Current Directions. *AI Magazine*, 18(4): 97–136, 1997. ISSN 07384602. doi: 10.1609/aimag.v18i4.1324. URL citeseer.ist.psu.edu/dietterich97machine.html.

[12] Agoston E Eiben and J E Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003. ISBN 3540401849.

[13] Laurene Fausett, editor. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994. ISBN 0-13-334186-0.

[14] D B Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press, 1995. ISBN 0780310381. doi: 10.1002/wics.005. URL http://books.google.com/books?hl=en&lr=&id=1SQuadczM9oC&oi=fnd&pg=PR9&dq=Evolutionary+computation&ots=w2sxZRRGf8&sig=hSITJmhe_wAWJ5u_22-FqKLMO1E.

[15] L J Fogel, A J Owens, and M J Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966.

[16] S. Forrest, L. Allen, and A. Perelson. Self-Nonself Discrimination in a Computer. *Proc. IEEE Symposium on Research Security and Privacy*, pages 202–212, 1994.

[17] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

[18] P Helman and S Forrest. An efficient algorithm for generating random antibody strings. Technical report, 1994.

[19] Paulien Hogeweg. The Roots of Bioinformatics in Theoretical Biology. *PLoS Comput Biol*, 7 (3):e1002021, 2011. doi: 10.1371/journal.pcbi.1002021. URL http://dx.doi.org/10.1371%2Fjournal.pcbi.1002021.

[20] John H Holland. Adapation in Natural and Artificial Systems. *SIAM Review*, 18(3):287–299, 1975. ISSN 00361445. doi: 10.1137/1018105. URL http://link.aip.org/link/SIREAD/v18/i3/p529/s1&Agg=doi.

[21] Divya Iyer, Arti Mohanpurkar, Sneha Janardhan, Dhanashree Rathod, and Amruta Sardeshmukh. Credit Card Fraud Detection Using Hidden Markov Model, 2011. ISSN 15455971. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4358713.

[22] Kelly James and Davis Lawrence. A Hybrid Genetic Algorithm for Classification. In *IJCAI*, 1991.

[23] N K Jerne. Towards a Network Theory of the Immune System. *Annals of Immunology*, 125(C):373–389, 1973.

[24] D Judd, P K McKinley, and A K Jain. Large-scale parallel data clustering, 1998. ISSN 10514651. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=709614.

[25] T Kaukoranta, P Franti, and O Nevalainen. A new iterative algorithm for VQ codebook generation, 1998.

[26] T Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, 1995. ISBN 3540679219. URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/3540679219.

[27] K F Lee, H W Hon, M Y Hwang, S Mahajan, and R Reddy. The SPHINX speech recognition system, 1989. ISSN 15206149. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=266459.

[28] K F Lee, H W Hon, and R Reddy. An overview of the SPHINX speech recognition system, 1990. ISSN 00963518. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=45616.

[29] W S McCulloch and W Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943. ISSN 00928240. doi: 10.1007/BF02478259. URL http://www.springerlink.com/index/61446605110620KG.pdf.

[30] T Mitchell. *Machine Learning*, volume 4. June 1997. ISBN 0070428077. doi: 10.1146/annurev.cs.04.060190.002221.

[31] Sherif A Mohamed and Moustafa M Fahmy. Image compression using VQ-BTC. *IEEE Transactions on Communications*, 43(7):2177–2182, 1995.

[32] J K Percus, O E Percus, and A S Perelson. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. *Proceedings of the National Academy of Sciences of the United States of America*, 90(5):1691–5, March 1993. ISSN 0027-8424. URL /pmc/articles/PMC45945/?report=abstract.

[33] I Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, TU Berlin, 1971.

[34] I Rechenberg. *Evolutionsstrategien: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Frommann Verlag, 1973.

[35] H P Schwefel. Natural evolution and collective optimum seeking. In A Sydow, editor, *Computational Systems AnalysisTopics and Trends*, pages 5–14. Elsevier, 1992.

[36] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994. ISSN 08997667. doi: 10.1162/neco.1994.6.2.215. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1994.6.2.215.

[37] C J Veenman, M J T Reinders, and E Backer. A Maximum Variance Cluster Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell*, 24:1273–1280, 2002.

[38] Alex Waibel. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation*, 1(1):39–46, 1989. ISSN 08997667. doi: 10.1162/neco.1989.1.1.39. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.1.39.

[39] Andrew Watkins and Lois C Boggess. A New Classifier Based on Resource Limited Artificial Immune Systems. *Proceedings of the 2002 Congress on Evolutionary Computation CEC02 Cat No02TH8600*, 2:1546–1551, 2002. URL http://kar.kent.ac.uk/13793/.

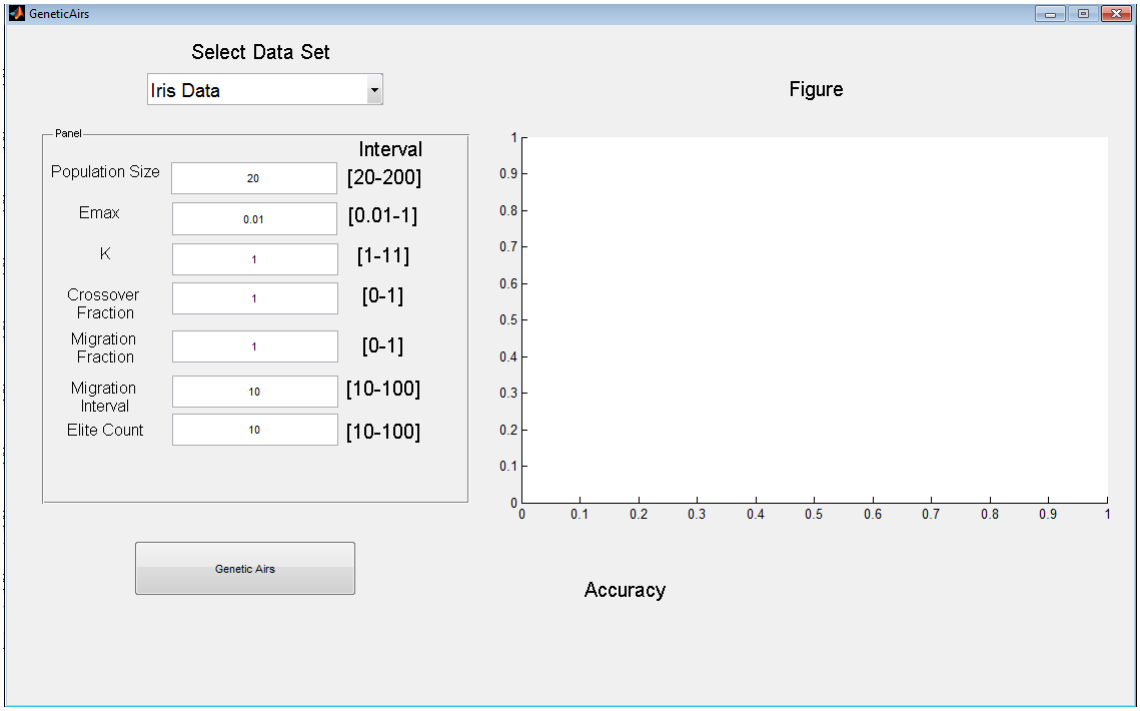
[40] Andrew Watkins, Jon Timmis, and Lois C Boggess. Artificial Immune Recognition System (AIRS) : An Immune Inspired Supervised Machine Learning Algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, 2004. URL http://eprints.whiterose.ac.uk/54717/.

[41] Andrews Watkins. *AIRS: A resource limited artificial immune system*. PhD thesis, Mississippi State University, 2001. URL `http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract`.

Παράρτημα Α΄

Αλγόριθμος Matlab

Α΄.1 GUI



Σχήμα Α΄.1: GUI Αλγορίθμου

Α΄.2 Κοινές συναρτήσεις για κάθε σύνολο δεδομένων

Α΄.2.1 Αρχικοποίηση Συνόλου

```
function [NM] = get_normalized_matrix(M)
% This function normalizes matrix M in the [0,1] interval
Min = min(M);
```

```
Max = max(M);
D = Max - Min;
D = 1./D;
Mm = [];
R = [];
[r,c] = size(M);
for k = 1:1:r
    R = [R;D];
    Mm = [Mm;Min];
end;
NM = M - Mm;
NM = NM.*R;
```

Α΄.2.2 Αρχικός Πληθυσμός

```
function [Range1]=InitialRange4(Data,nvars,PopulationSize)
[Rows,Colls]=size(Data);
C=mean(Data);

Range1=[];
for i=1:length(C);
   a1=C(i)+0.05;
   b1=C(i)-0.05;
 R = a1 + (b1-a1).*rand(PopulationSize,1);
   Range1=[Range1 R];
end

Range1=repmat(Range1,1,nvars/Colls);


end
```

Α΄.2.3 Αντικειμενική Συνάρτηση

```
%2-10-12
function [F] = Fitness_Function_Vec(X,Data,emax,dim)
% ************************************************************************
%                          FUNCTION DESCRIPTION:
% ************************************************************************
% Objective Function of the Airs Genetic Algorithm.
% This Function computes the minimum Distance of the values of Data and
% the elements that the Genetic Algorithm generates. This function trys to
% minimize the distance between the Data and the Detectors to some value
% less than emax.
% % ************************************************************************
%                          INPUT VARIABLES DEFINITION:
% ************************************************************************
% Data: The values our Dataset
% X: The values that the Genetic Algorithm generate.
%
% ************************************************************************
%                          OUTPUT VARIABLES DEFINITION:
% ************************************************************************
% F: The sum of the minimum distance.
```

```
%% Set our auxiliary variables
[~,Colls]=size(X);              % Compute the Rows and Collumns
sep=Colls/dim;                     % Compute the Rows of the Reshaped X matrix
                                   % so it would Colls/dim x dim dimension
XReshaped=reshape(X,dim,sep);   % Reshape the X matrix
X2Colls=XReshaped';
%% Compute the distance between one element (one Row) of matrix Data to
%% each element of matrix X
Distance=(((pdist2(Data,X2Colls)/sqrt(dim))-emax));
%% Set to zero all the distances which are less than emax. This means that
%% if d(x,y)<=emax d(x,y)-emax<=0 so the Data is inside the radious of the Detector.
Distance(Distance<=0)=0;
%% Compute the minimum of each row
MinimumDistance=min(Distance,[],2);
%% Compute the sum of the minimum
F=sum(MinimumDistance);
end
```

Α΄.2.4 Βοηθητική Συνάρτηση

```
function [F] = Fitness_Evaluation_Vec(X,Data,emax,dim)
fun = @(x)Fitness_Function_Vec(x,Data,emax,dim);
L = size(X,2);
F = blkproc(X,[1 L],fun);
end
```

Α΄.3 Iris Data

Α΄.3.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_Iris_Data_Opt(Data,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
CrossoverFraction,~)

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12


%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                          %Dimensions of the Dataset

%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);


%% We call the auxiliary function ... which then computes the objective
```

```
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);

%% We set the options of the genetic algorithm


nvars=round(Rows*Colls*0.8);   %The number of Detectors (How many values the genetic algorithm will generat
% PopInit=InitialRange4(Data,nvars,PopulationSize);

if(mod(nvars,2)==1)
    nvars=nvars+1;
end

options = gaoptimset;

options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'CreationFcn', @gacreationlinearfeasible);
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'SelectionFcn', {  @selectiontournament,4 });
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
 lb=repmat(Minimum,1,nvars/dim);
 ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;

end
```

A΄.3.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy,Output]=Iris_Graph_iterations(PopulationSize,EliteCount,...
MigrationInterval,MigrationFraction,...
CrossoverFraction,k,emax)
%% Application of The Genetic_Airs_01 Algorith to Iris DataSet. With a
%% training set and a test set.
%% Loads and Normalize Data from Iris Dataset.
load fisheriris.mat                 %Load Iris dataset
Data=meas;                          %Set the Iris Dataset as Data
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Separate the Data into Species.
SetosaData=Data(1:50,1:4);
VersicolorData=Data(51:100,1:4);
VirginicaData=Data(101:150,1:4);
```

```
%% Length of the Test Set is 10% of the entire set.
LengthSetosaTestData=10;
OverallC=zeros(3,3);
for i=1:LengthSetosaTestData:length(SetosaData)-1
SetosaTrainingData=SetosaData;
VersicolorTrainingData=VersicolorData;
VirginicaTrainingData=VirginicaData;
SetosaTestData=SetosaTrainingData(i:i+LengthSetosaTestData-1,:);
SetosaTrainingData(i:i+LengthSetosaTestData-1,:)=[];
VersicolorTestData=VersicolorTrainingData(i:i+LengthSetosaTestData-1,:);
VersicolorTrainingData(i:i+LengthSetosaTestData-1,:)=[];
VirginicaTestData=VirginicaTrainingData(i:i+LengthSetosaTestData-1,:);
VirginicaTrainingData(i:i+LengthSetosaTestData-1,:)=[];
[~,D1,~,Output]=AirsGenetic_Iris_Data_Opt(SetosaTrainingData,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
CrossoverFraction);
[~,D2,~,Output]=AirsGenetic_Iris_Data_Opt(VersicolorTrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction);
[~,D3,~,Output]=AirsGenetic_Iris_Data_Opt(VirginicaTrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction);
Sample=[SetosaTestData;VersicolorTestData;VirginicaTestData];
Training=[D1;D2;D3];
[Rows, Colls]=size(D1);
Group=[ones(Rows,1);ones(Rows,1)*2;ones(Rows,1)*3];
G=[ones(LengthSetosaTestData,1);ones(LengthSetosaTestData,1)*2; ...
    ones(LengthSetosaTestData,1)*3];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/150)*100;
['Test set Accuracy ',num2str(Accuracy),' % ']
end
```

Α΄.3.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% Iris Data Test
clear;
clc;
PopulationSize=150 ;
emax=0.04;
MigrationInterval = 50;
MigrationFraction = 0.6;
EliteCount=5;
k=5;
CrossoverFraction = 1;
AccuracyAll=[];
Matrixs=[];
matlabpool(4)
parfor i=1:10
[Accuracy]=Iris_Graph_iterations(PopulationSize,EliteCount, ...
MigrationInterval,MigrationFraction,CrossoverFraction,k,emax);
```

```
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
a=sum(AccuracyAll)/10;
fprintf('Accuracy : %f\n', a);
matlabpool close
```

A΄.4 Prima Indians

A΄.4.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_PrimaIndiansData(Data,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,~)
%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
nvars=round(Rows*Colls*1);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation', InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
```

```
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.4.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy,OverallC]=PrimaIndiansDataTest10fold(PopulationSize,...
    EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,...
    k,emax);
load PrimaIndians.mat;
Class1=pima0x2Dindians0x2Ddiabetes;
a=find(Class1(:,9)==1);
ClassA=Class1(a,:);
ClassA(:,9)=[];
b=find(Class1(:,9)==0);
ClassB=Class1(b,:);
ClassB(:,9)=[];
[NM1]=get_normalized_matrix(ClassA);
[NM2]=get_normalized_matrix(ClassB);
ClassA=NM1(1:260,:);
ClassB=NM2;
Fold1=length(ClassA)*0.1;
Fold2=length(ClassB)*0.1;
OverallC=zeros(2,2);
for i=1:10
Fs1=Fold1*(i-1);
Fe1=Fold1*i;
Fs2=Fold2*(i-1);
Fe2=Fold2*i;
ClassATrainingData=ClassA;
ClassBTrainingData=ClassB;

ClassATestData=ClassA(Fs1+1:Fe1,:);
ClassATrainingData(Fs1+1:Fe1,:)=[];

ClassBTestData=ClassB(Fs2+1:Fe2,:);
ClassBTrainingData(Fs2+1:Fe2,:)=[];

[F1,D1,nvars,Output]=AirsGenetic_PrimaIndiansData(ClassATrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F2,D2,nvars,Output]=AirsGenetic_PrimaIndiansData(ClassBTrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);

[Rows1,Colls]=size(D1);
[Rows2,Colls]=size(D2);
[Rows3,Colls]=size(ClassATestData);
```

```
[Rows4, Colls]=size(ClassBTestData);

Sample=[ClassATestData;ClassBTestData];
Training=[D1;D2];

Group=[ones(Rows1,1);ones(Rows2,1)*2];
G=[ones(Rows3,1);ones(Rows4,1)*2];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
OverallC=OverallC+C;

end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/760)*100;
['Test set Accuracy ',num2str(Accuracy),' % ']

end
```

Α΄.4.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% Fixed 28-Oct-12 96.20 me 96.26 xwris reset me reset paei sto 97.0
clear;
clc;
PopulationSize=140;
emax=0.06; %Fix
MigrationInterval =12; %Fix
MigrationFraction = 0.5; %Fix.
k=11;
EliteCount=8; %Fix
CrossoverFraction = 1;
AccuracyAll=[];
Matrixs=[];
pame=[];
matlabpool(4)
parfor i=1:10
[Accuracy,Overall]=PrimaIndiansDataTest10fold(PopulationSize,EliteCount,MigrationInterval,MigrationFraction
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
matlabpool close force local
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Tournament','NumberTitle','off');
plot(pame(:,1),pame(:,2),'--rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Population Size');
ylabel('%Accuracy');
a=sum(AccuracyAll)/i;
fprintf('Accuracy : %f\n', a);
```

Α΄.5 Sonar Data

Α΄.5.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_SonarData(Data,emax,PopulationSize,EliteCount,MigrationInterval,Mi
%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
nvars=round(Rows*Colls*1);   %The number of Detectors (How many values the genetic algorithm will generate
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
InPop=InitialRange4(Data,nvars,PopulationSize);
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn', @selectionremainder);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.5.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy,OverallC]=SonarDataTest10fold(PopulationSize,EliteCount,MigrationInterval,MigrationFract

load sonar.mat;



Class1=sonar;
```

```
a=find(Class1(:,61)==1);
ClassA=Class1(a,:);
ClassA(:,61)=[];
b=find(Class1(:,61)==0);
ClassB=Class1(b,:);
ClassB(:,61)=[];
[NM1]=get_normalized_matrix(ClassA);
[NM2]=get_normalized_matrix(ClassB);


ClassA=NM1(1:100,:);
ClassB=NM2(1:90,:);


% ClassA=ClassA(1:100,:);
% ClassB=ClassB(1:90,:);

Fold1=length(ClassA)*0.1;
Fold2=length(ClassB)*0.1;
OverallC=zeros(2,2);



for i=1:10


Fs1=Fold1*(i-1);
Fe1=Fold1*i;

Fs2=Fold2*(i-1);
Fe2=Fold2*i;

ClassATrainingData=ClassA;
ClassBTrainingData=ClassB;

ClassATestData=ClassA(Fs1+1:Fe1,:);
ClassATrainingData(Fs1+1:Fe1,:)=[];

ClassBTestData=ClassB(Fs2+1:Fe2,:);
ClassBTrainingData(Fs2+1:Fe2,:)=[];

[F1,D1,nvars,Output]=AirsGenetic_SonarData(ClassATrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F2,D2,nvars,Output]=AirsGenetic_SonarData(ClassBTrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);

[Rows1,Colls]=size(D1);
[Rows2,Colls]=size(D2);
[Rows3,Colls]=size(ClassATestData);
[Rows4,Colls]=size(ClassBTestData);

Sample=[ClassATestData;ClassBTestData];
Training=[D1;D2];
```

```
Group=[ones(Rows1,1);ones(Rows2,1)*2];
G=[ones(Rows3,1);ones(Rows4,1)*2];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
OverallC=OverallC+C;
[' Fold ',num2str(i),' % '];
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/190)*100;
[' Test set Accuracy ',num2str(Accuracy),' % ']

end
```

Α΄.5.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% Sonar Data Test
clear;
clc;
PopulationSize=180;
emax=0.16;
EliteCount=8;
CrossoverFraction = 1;
MigrationInterval =20;
MigrationFraction = 1;
AccuracyAll=[];
Matrixs=[];
pame=[];
k=2;

matlabpool(4)
parfor i=1:10
[Accuracy,Overall]=SonarDataTest10fold(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,Crosso
Matrixs=[Matrixs;Po];
AccuracyAll=[AccuracyAll;Accuracy];
end
matlabpool close force local
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Population Size','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Population Size');
ylabel('%Accuracy');
```

Α΄.6 C1vsC2

Α΄.6.1 Γενετικός Αλγοριθμος

```
function [X]=AirsGenetic_FeaturesData(Data,emax,PopulationSize,...
    EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,...
    Detectors)
%% RUNS THE  GENETIC AIRS OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
```

```
%%
%15-10-12
%% Computes the Dimension of the Data Set
[Rows, Colls]=size(Data);
dim=Colls;                              %Dimensions of the Data Set
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function Fobj which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
%% We call the function InitialRange4 that computes the Initial Population
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
 lb=repmat(Minimum,1,nvars/dim);
 ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,~,~,Output] =ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
end
```

Α΄.6.2 Cross-Validation και K-NN Algorithm

```
function   [Accuracy]=Features_Airs(PopulationSize,EliteCount,...
    MigrationInterval,MigrationFraction,CrossoverFraction,k,...
    emax,Detectors)
load features30.mat
Data=N;
[NM]=normalized(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Fold=10;
OverallC=zeros(2,2);
Call=zeros(2,2);
```

```
for i=1:10
%% For 10-Cross-Validation- Fs1 is the Fold
Fs1=Fold*(i-1);
Fe1=Fold*i;

%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;

%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];

%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];

%% Call Genetic Airs Algorithm
[D1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,PopulationSize,...
    EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,...
    Detectors);
[D2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,PopulationSize,...
    EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,...
    Detectors);
Sample=[Class1TestData;Class2TestData];
Training=[D1;D2];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2];
G=[ones(Fold,1)*1;ones(Fold,1)*2];
Class=knnclassify(Sample,Training,Group,k);
[C] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/200)*100;
['Test set Accuracy ',num2str(Accuracy),'%']
end
```

Α΄.6.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C1vsC2 Data Test
clear;
clc;
tic;
PopulationSize=90;
emax=0.26;
EliteCount=4;
MigrationFraction = 0.9;
MigrationInterval = 48;
CrossoverFraction = 1;
k=5;
AccuracyAll=[];
Matrixs=[];
```

```
Detectors=0.8;
matlabpool(4)
parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations C1vsC2','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
matlabpool close
toc;
```

Α΄.7 C1vsC2vsC3

Α΄.7.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors)
%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
   nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'Vectorized', 'on');
```

```
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

Α΄.7.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy]=Features_Airs(PopulationSize,EliteCount,...
    MigrationInterval,MigrationFraction,CrossoverFraction,k,...
    emax,Detectors)
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Fold=10;
OverallC=zeros(3,3);
Fold1=length(Class1)*0.1;
Fold2=length(Class2)*0.1;
Fold3=length(Class3)*0.1;
Call=zeros(3,3);
for i=1:10
Fs1=Fold1*(i-1);
Fe1=Fold1*i;
Fs2=Fold2*(i-1);
Fe2=Fold2*i;
Fs3=Fold3*(i-1);
Fe3=Fold3*i;
%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs2+1:Fe2,:);
Class2TrainingData(Fs3+1:Fe3,:)=[];
%% Class3
Class3TestData=Class3TrainingData(Fs3+1:Fe3,:);
Class3TrainingData(Fs3+1:Fe3,:)=[];
%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,...
```

```
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
Sample=[Class1TestData;Class2TestData;Class3TestData];
Training=[D1;D2;D3];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/300)*100;
['Test set Accuracy ',num2str(Accuracy),' % ']

end
```

Α΄.7.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C1vsC2vsC3 test
clear;
clc;
PopulationSize=80 ;
emax=0.23;
MigrationFraction = 0.6;
MigrationInterval = 72;
k=9;
EliteCount=5;
CrossoverFraction = 1;
AccuracyAll=[];
Matrixs=[];
pame=[];
Detectors=0.8;
matlabpool(4)

parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations C1vsC2vsC3','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
```

```
matlabpool close
```

A΄.8 C1vsC2vsC3vsC4

A΄.8.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
CrossoverFraction,Detectors)

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);   %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

Α΄.8.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy]=Features_Airs(PopulationSize,EliteCount,...
MigrationInterval,MigrationFraction,CrossoverFraction,k,emax,...
Detectors)
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Fold=10;
OverallC=zeros(4,4);
Fold1=length(Class1)*0.1;
Fold2=length(Class2)*0.1;
Fold3=length(Class3)*0.1;
Fold4=length(Class4)*0.1;
Call=zeros(4,4);
for i=1:10
Fs1=Fold1*(i-1);
Fe1=Fold1*i;
Fs2=Fold2*(i-1);
Fe2=Fold2*i;
Fs3=Fold3*(i-1);
Fe3=Fold3*i;
Fs4=Fold4*(i-1);
Fe4=Fold4*i;
%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs2+1:Fe2,:);
Class2TrainingData(Fs3+1:Fe3,:)=[];
%% Class3
Class3TestData=Class3TrainingData(Fs3+1:Fe3,:);
Class3TrainingData(Fs3+1:Fe3,:)=[];
%% Class4
Class4TestData=Class4TrainingData(Fs4+1:Fe4,:);
Class4TrainingData(Fs4+1:Fe4,:)=[];
%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax, ...
```

```
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData];
Training=[D1;D2;D3;D4];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/400)*100;
['Test set Accuracy ',num2str(Accuracy),'%']
end
```

Α΄.8.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C1234 test
clear;
clc;
PopulationSize=40;
emax=0.18;
MigrationFraction = 0.6;
MigrationInterval = 24;
k=9;
CrossoverFraction = 0.8;
EliteCount=3;
AccuracyAll=[];
Matrixs=[];
pame=[];
Detectors=0.8;
matlabpool(4)
parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and PopulationSize C1vsC2vsC3vsC4','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
matlabpool close
```

Α΄.9 C1vsC2vsC3vsC4vsC5

Α΄.9.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors)
%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                          %Dimensions of the Dataset

%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
%nvars=compute_nvars(Data,emax,dim);
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn', @selectionremainder);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);

%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.9.2 Cross-Validation και K-NN Algorithm

```
function   [Accuracy]=Features_Airs(PopulationSize,EliteCount,...
```

```
MigrationInterval,MigrationFraction,CrossoverFraction,k,...
emax,Detectors)
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Fold=10;

OverallC=zeros(5,5);
Call=zeros(5,5);
%% Folds
Fold1=length(Class1)*0.1;



for i=1:10
%% Start and End
Fs1=Fold1*(i-1);
Fe1=Fold1*i;

%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];

%% Class3
Class3TestData=Class3TrainingData(Fs1+1:Fe1,:);
Class3TrainingData(Fs1+1:Fe1,:)=[];

%% Class4
Class4TestData=Class4TrainingData(Fs1+1:Fe1,:);
Class4TrainingData(Fs1+1:Fe1,:)=[];

%% Class5
Class5TestData=Class5TrainingData(Fs1+1:Fe1,:);
Class5TrainingData(Fs1+1:Fe1,:)=[];

%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax, ...
```

```
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);
[~,D5,~,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax, ...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors);

Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData; ...
    Class5TestData];
Training=[D1;D2;D3;D4;D5];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4; ...
    ones(Rows,1)*5];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4; ...
    ones(Fold,1)*5];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/500)*100;
['Test set Accuracy ',num2str(Accuracy),' % ']

end
```

Α΄.9.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C12345 test
PopulationSize=80;
emax=0.1;
MigrationFraction = 0.7;
MigrationInterval = 15;
k=3;
CrossoverFraction = 1;
EliteCount=5;
AccuracyAll=[];
Matrixs=[];
pame=[];
Detectors=0.8;
parfor w2=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;MigrationInterval];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
```

```
xlabel('Iterations');
ylabel('%Accuracy');
toc;
matlabpool close
```

Α΄.10 C1vsC2vsC3vsC4vsC5vsC6

Α΄.10.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction, ...
CrossoverFraction,Detectors,r)

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize,r);
if(mod(nvars,2)==1)
   nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);

options = gaoptimset(options,'SelectionFcn', {  @selectiontournament,4 });
options = gaoptimset(options,'Vectorized', 'on');

%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);

%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
```

```
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.10.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverF
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Class6=Data(501:600,:);

Fold=10;
OverallC=zeros(6,6);
Fold1=length(Class1)*0.1;

Call=zeros(6,6);
for i=1:10

Fs1=Fold1*(i-1);
Fe1=Fold1*i;

%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
Class6TrainingData=Class6;

%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];

%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];

%% Class3
Class3TestData=Class3TrainingData(Fs1+1:Fe1,:);
Class3TrainingData(Fs1+1:Fe1,:)=[];

%% Class4
Class4TestData=Class4TrainingData(Fs1+1:Fe1,:);
Class4TrainingData(Fs1+1:Fe1,:)=[];
```

```
%% Class5
Class5TestData=Class5TrainingData(Fs1+1:Fe1,:);
Class5TrainingData(Fs1+1:Fe1,:)=[];

%% Class5
Class6TestData=Class6TrainingData(Fs1+1:Fe1,:);
Class6TrainingData(Fs1+1:Fe1,:)=[];

%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,PopulationSize,EliteCount,MigrationInterv
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,PopulationSize,EliteCount,MigrationInterv
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax,PopulationSize,EliteCount,MigrationInterv
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax,PopulationSize,EliteCount,MigrationInterv
[~,D5,~,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax,PopulationSize,EliteCount,MigrationInterv
[~,D6,~,Output6]=AirsGenetic_FeaturesData(Class6TrainingData,emax,PopulationSize,EliteCount,MigrationInterv

Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData;Class5TestData;Class6TestData];
Training=[D1;D2;D3;D4;D5;D6];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4;ones(Rows,1)*5;ones(Rows,1)*6];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4;ones(Fold,1)*5;ones(Fold,1)*6];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/600)*100;
['Test set Accuracy ',num2str(Accuracy),'%']
end
```

Α΄.10.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C123456 test
r=0.08;
PopulationSize=50;
emax=0.1;
MigrationFraction = 0.6;
MigrationInterval = 32;
k=6;
EliteCount=1;
CrossoverFraction = 1;
Detectors=0.9;
matlabpool(4)
AccuracyAll=[];
Matrixs=[];
pame=[];
tic
parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
```

```
figure('Name','Accuracy and Iterations C123456','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
toc
matlabpool close
```

Α΄.11 C1vsC2vsC3vsC4vsC5vsC6vsC7

Α΄.11.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
CrossoverFraction,Detectors)

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn', {  @selectiontournament,3 });
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
```

```
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.11.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy]=Features_Airs(PopulationSize,EliteCount,...
MigrationInterval,MigrationFraction,CrossoverFraction,k,...
emax,Detectors)
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Class6=Data(501:600,:);
Class7=Data(601:700,:);

Fold=10;

OverallC=zeros(7,7);
Call=zeros(7,7);
%% Folds
Fold1=length(Class1)*0.1;



for i=1:10
%% Start and End
Fs1=Fold1*(i-1);
Fe1=Fold1*i;

%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
Class6TrainingData=Class6;
Class7TrainingData=Class7;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];
```

```
%% Class3
Class3TestData=Class3TrainingData(Fs1+1:Fe1,:);
Class3TrainingData(Fs1+1:Fe1,:)=[];

%% Class4
Class4TestData=Class4TrainingData(Fs1+1:Fe1,:);
Class4TrainingData(Fs1+1:Fe1,:)=[];

%% Class5
Class5TestData=Class5TrainingData(Fs1+1:Fe1,:);
Class5TrainingData(Fs1+1:Fe1,:)=[];

%% Class6
Class6TestData=Class6TrainingData(Fs1+1:Fe1,:);
Class6TrainingData(Fs1+1:Fe1,:)=[];
%% Class7
Class7TestData=Class7TrainingData(Fs1+1:Fe1,:);
Class7TrainingData(Fs1+1:Fe1,:)=[];

%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,...
    emax,PopulationSize,EliteCount,MigrationInterval,...
    MigrationFraction,CrossoverFraction,Detectors);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D5,~,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D6,~,Output5]=AirsGenetic_FeaturesData(Class6TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D7,~,Output5]=AirsGenetic_FeaturesData(Class7TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);

Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData;...
    Class5TestData;Class6TestData;Class7TestData];
Training=[D1;D2;D3;D4;D5;D6;D7];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4;...
    ones(Rows,1)*5;ones(Rows,1)*6;ones(Rows,1)*7];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4;...
    ones(Fold,1)*5;ones(Fold,1)*6;ones(Fold,1)*7];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
```

```
Accuracy=(OverallSum/700)*100;
[' Test set Accuracy ',num2str(Accuracy),' % ']

end
```

Α΄.11.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% Test C1234567
PopulationSize=50;
emax=0.07;
EliteCount=5;
MigrationFraction = 0.2;
MigrationInterval = 56;
k=5;
CrossoverFraction = 1;
AccuracyAll=[];
Matrixs=[];
pame=[];
Detectors=0.9;
parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,...
MigrationInterval,MigrationFraction,CrossoverFraction,...
k,emax,Detectors);
Matrixs=[Matrixs;MigrationInterval];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations C1234567','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('MigrationInterval');
ylabel('%Accuracy');
toc;
matlabpool close
```

Α΄.12 C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8

Α΄.12.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,PopulationSize,EliteCount,MigrationInterval,

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                          %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
```

```
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);   %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize,r);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

A΄.12.2 Cross-Validation και K-NN Algorithm

```
function   [Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverF
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Class6=Data(501:600,:);
Class7=Data(601:700,:);
Class8=Data(701:800,:);
Fold=10;
OverallC=zeros(8,8);
Call=zeros(8,8);
%% Folds
Fold1=length(Class1)*0.1;
for i=1:10
%% Start and End
Fs1=Fold1*(i-1);
```

```
Fe1=Fold1*i;
%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
Class6TrainingData=Class6;
Class7TrainingData=Class7;
Class8TrainingData=Class8;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];

%% Class3
Class3TestData=Class3TrainingData(Fs1+1:Fe1,:);
Class3TrainingData(Fs1+1:Fe1,:)=[];

%% Class4
Class4TestData=Class4TrainingData(Fs1+1:Fe1,:);
Class4TrainingData(Fs1+1:Fe1,:)=[];

%% Class5
Class5TestData=Class5TrainingData(Fs1+1:Fe1,:);
Class5TrainingData(Fs1+1:Fe1,:)=[];

%% Class6
Class6TestData=Class6TrainingData(Fs1+1:Fe1,:);
Class6TrainingData(Fs1+1:Fe1,:)=[];
%% Class7
Class7TestData=Class7TrainingData(Fs1+1:Fe1,:);
Class7TrainingData(Fs1+1:Fe1,:)=[];

%% Class 8
Class8TestData=Class8TrainingData(Fs1+1:Fe1,:);
Class8TrainingData(Fs1+1:Fe1,:)=[];
%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D5,~,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D6,~,Output6]=AirsGenetic_FeaturesData(Class6TrainingData,emax,...
```

```
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D7,~,Output7]=AirsGenetic_FeaturesData(Class7TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);
[~,D8,~,Output8]=AirsGenetic_FeaturesData(Class8TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors,r);

Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData;...
    Class5TestData;Class6TestData;Class7TestData;Class8TestData];
Training=[D1;D2;D3;D4;D5;D6;D7;D8];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4;...
    ones(Rows,1)*5;ones(Rows,1)*6;ones(Rows,1)*7;ones(Rows,1)*8];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4;...
    ones(Fold,1)*5;ones(Fold,1)*6;ones(Fold,1)*7;ones(Fold,1)*8];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/800)*100;
['Test set Accuracy ',num2str(Accuracy),' % ']
end
```

Α΄.12.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% First
r=0.08;

emax=0.21;
MigrationFraction = 0.1;
MigrationInterval = 24;
k=6;
EliteCount=1;
PopulationSize=90;
CrossoverFraction = 1;
Detectors=1;

AccuracyAll=[];
Matrixs=[];
pame=[];

matlabpool(4)

parfor i=1:10

[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverFraction,k,
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations','NumberTitle','off');
```

```
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
matlabpool close
```

A΄.13 C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9

A΄.13.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,PopulationSize,EliteCount,MigrationInterval,

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
%% Computes the Dimension of the Data
[Rows,Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
a=Detectors;
nvars=round(Rows*Colls*a);    %The number of Detectors (How many values the genetic algorithm will generate
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
lb=repmat(Minimum,1,nvars/dim);
ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
```

```
X=flipud(rot90(ds2));
F=fval;
end
```

Α΄.13.2 Cross-Validation και K-NN Algorithm

```
function [Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverF
load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Class6=Data(501:600,:);
Class7=Data(601:700,:);
Class8=Data(701:800,:);
Class9=Data(801:900,:);
Fold=10;
OverallC=zeros(9,9);
Call=zeros(9,9);
%% Folds
Fold1=length(Class1)*0.1;
for i=1:10
%% Start and End
Fs1=Fold1*(i-1);
Fe1=Fold1*i;
%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
Class6TrainingData=Class6;
Class7TrainingData=Class7;
Class8TrainingData=Class8;
Class9TrainingData=Class9;
%% Class 1
Class1TestData=Class1TrainingData(Fs1+1:Fe1,:);
Class1TrainingData(Fs1+1:Fe1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(Fs1+1:Fe1,:);
Class2TrainingData(Fs1+1:Fe1,:)=[];
%% Class3
Class3TestData=Class3TrainingData(Fs1+1:Fe1,:);
Class3TrainingData(Fs1+1:Fe1,:)=[];
%% Class4
Class4TestData=Class4TrainingData(Fs1+1:Fe1,:);
Class4TrainingData(Fs1+1:Fe1,:)=[];
%% Class5
Class5TestData=Class5TrainingData(Fs1+1:Fe1,:);
Class5TrainingData(Fs1+1:Fe1,:)=[];
```

```
%% Class6
Class6TestData=Class6TrainingData(Fs1+1:Fe1,:);
Class6TrainingData(Fs1+1:Fe1,:)=[];
%% Class7
Class7TestData=Class7TrainingData(Fs1+1:Fe1,:);
Class7TrainingData(Fs1+1:Fe1,:)=[];
%% Class 8
Class8TestData=Class8TrainingData(Fs1+1:Fe1,:);
Class8TrainingData(Fs1+1:Fe1,:)=[];
%% Class 9
Class9TestData=Class9TrainingData(Fs1+1:Fe1,:);
Class9TrainingData(Fs1+1:Fe1,:)=[];
%% Airs Algorithm All Classes
[~,D1,~,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D2,~,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D3,~,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D4,~,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D5,~,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D6,~,Output6]=AirsGenetic_FeaturesData(Class6TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D7,~,Output7]=AirsGenetic_FeaturesData(Class7TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D8,~,Output8]=AirsGenetic_FeaturesData(Class8TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
[~,D9,~,Output8]=AirsGenetic_FeaturesData(Class9TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,Detectors);
Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData;...
    Class5TestData;Class6TestData;Class7TestData;Class8TestData;...
    Class9TestData];
Training=[D1;D2;D3;D4;D5;D6;D7;D8;D9];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4;...
    ones(Rows,1)*5;ones(Rows,1)*6;ones(Rows,1)*7;ones(Rows,1)*8;...
    ones(Rows,1)*9];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4;...
    ones(Fold,1)*5;ones(Fold,1)*6;ones(Fold,1)*7;ones(Fold,1)*8;...
    ones(Fold,1)*9];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;
end
```

```
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/900)*100;
[' Test set Accuracy ',num2str(Accuracy),' % ']
end
```

Α΄.13.3 Επαναλήψεις με καλύτερες παραμέτρους

```
clear;
clc;
r=0.08;
emax=0.04;
MigrationFraction = 0.4;
MigrationInterval = 56;
k=9;
EliteCount=9;
PopulationSize=80;
CrossoverFraction = 1;
Detectors=0.8;

AccuracyAll=[];
Matrixs=[];
pame=[];

matlabpool(4)

parfor i=1:10

[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,...
    MigrationFraction,CrossoverFraction,k,emax,Detectors);
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations','NumberTitle','off');
plot(pame(:,1),pame(:,2),'—rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('Iterations');
ylabel('%Accuracy');
matlabpool close
```

Α΄.14 C1vsC2vsC3vsC4vsC5vsC6vsC7vsC8vsC9vsC10

Α΄.14.1 Γενετικός Αλγοριθμος

```
function [F,X,nvars,Output]=AirsGenetic_FeaturesData(Data,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction,~)

%% RUNS THE AIRS GENETIC OPTIMIZATION PROBLEM WITH THE DATASET
%% OF N-D DIMENSION
%
%%
%15-10-12
```

```
%% Computes the Dimension of the Data
[Rows, Colls]=size(Data);
dim=Colls;                              %Dimensions of the Dataset
%% Computes the maximum and minimum value of the Dataset per Collumn so we
%% will have the maximum and minimum value of x,y so we can set the search
%% space between these values
Maximum=max(Data);
Minimum=min(Data);
%% We call the auxiliary function ... which then computes the objective
%% function
Fobj = @(X)Fitness_Evaluation_Vec(X,Data,emax,dim);
%% We set the options of the genetic algorithm
nvars=round(Rows*Colls*0.8);    %The number of Detectors (How many values the genetic algorithm will generat
InPop=InitialRange4(Data,nvars,PopulationSize);
if(mod(nvars,2)==1)
    nvars=nvars+1;
end
options = gaoptimset;
options=gaoptimset(options,'InitialPopulation',InPop);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'CrossoverFraction', CrossoverFraction);
options = gaoptimset(options,'MigrationInterval', MigrationInterval);
options = gaoptimset(options,'MigrationFraction', MigrationFraction);
options = gaoptimset(options,'PopulationSize', PopulationSize);
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'Vectorized', 'on');
%% We set the lb=Lower Bound and ub= Upper Bound of the Genetic Algorithm.
%% So the Algorithm will generate x so that xmin<x<xmax and y so that
%% ymin<y<ymax. lb is a matrix 1xlength(X) and ub is a matrix 1xlength(X)
 lb=repmat(Minimum,1,nvars/dim);
 ub=repmat(Maximum,1,nvars/dim);
%% Runs the genetic algorithm
[x,fval,exitflag,Output,population,score] = ...
ga(Fobj,nvars,[],[],[],[],lb,ub,[],options);
X=[];
ds2=reshape(x,dim,length(x)/dim);
X=flipud(rot90(ds2));
F=fval;
end
```

Α΄.14.2 Cross-Validation και K-NN Algorithm

```
function  [Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,MigrationFraction,CrossoverF

load features30.mat
Data=N;
[NM]=get_normalized_matrix(Data);
Data=NM;
%% Initialize Classes
Class1=Data(1:100,:);
Class2=Data(101:200,:);
Class3=Data(201:300,:);
Class4=Data(301:400,:);
Class5=Data(401:500,:);
Class6=Data(501:600,:);
```

```
Class7=Data(601:700,:);
Class8=Data(701:800,:);
Class9=Data(801:900,:);
Class10=Data(901:1000,:);


Fold=10;

OverallC=zeros(10,10);
Call=[];


for i=1:Fold:length(Class1)-1
%% Initialize The Training Classes
Class1TrainingData=Class1;
Class2TrainingData=Class2;
Class3TrainingData=Class3;
Class4TrainingData=Class4;
Class5TrainingData=Class5;
Class6TrainingData=Class6;
Class7TrainingData=Class7;
Class8TrainingData=Class8;
Class9TrainingData=Class9;
Class10TrainingData=Class10;

%% Class 1
Class1TestData=Class1TrainingData(i:i+Fold-1,:);
Class1TrainingData(i:i+Fold-1,:)=[];
%% Class2
Class2TestData=Class2TrainingData(i:i+Fold-1,:);
Class2TrainingData(i:i+Fold-1,:)=[];
%% Class3
Class3TestData=Class3TrainingData(i:i+Fold-1,:);
Class3TrainingData(i:i+Fold-1,:)=[];

%% Class4
Class4TestData=Class4TrainingData(i:i+Fold-1,:);
Class4TrainingData(i:i+Fold-1,:)=[];
%% Class5
Class5TestData=Class5TrainingData(i:i+Fold-1,:);
Class5TrainingData(i:i+Fold-1,:)=[];
%% Class6
Class6TestData=Class6TrainingData(i:i+Fold-1,:);
Class6TrainingData(i:i+Fold-1,:)=[];

%% Class7
Class7TestData=Class7TrainingData(i:i+Fold-1,:);
Class7TrainingData(i:i+Fold-1,:)=[];
%% Class8
Class8TestData=Class8TrainingData(i:i+Fold-1,:);
Class8TrainingData(i:i+Fold-1,:)=[];
%% Class9
Class9TestData=Class9TrainingData(i:i+Fold-1,:);
Class9TrainingData(i:i+Fold-1,:)=[];

%% Class10
```

```
Class10TestData=Class10TrainingData(i:i+Fold-1,:);
Class10TrainingData(i:i+Fold-1,:)=[];

%% Airs Algorithm All Classes

[F1,D1,nvars1,Output1]=AirsGenetic_FeaturesData(Class1TrainingData,...
    emax,PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);

[F2,D2,nvars2,Output2]=AirsGenetic_FeaturesData(Class2TrainingData,...
    emax,PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);

[F3,D3,nvars3,Output3]=AirsGenetic_FeaturesData(Class3TrainingData,...
    emax,PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F4,D4,nvars4,Output4]=AirsGenetic_FeaturesData(Class4TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F5,D5,nvars5,Output5]=AirsGenetic_FeaturesData(Class5TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F6,D6,nvars6,Output6]=AirsGenetic_FeaturesData(Class6TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F7,D7,nvars7,Output7]=AirsGenetic_FeaturesData(Class7TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F8,D8,nvars8,Output8]=AirsGenetic_FeaturesData(Class8TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F9,D9,nvars9,Output9]=AirsGenetic_FeaturesData(Class9TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);
[F10,D10,nvars10,Output10]=AirsGenetic_FeaturesData(Class10TrainingData,emax,...
    PopulationSize,EliteCount,MigrationInterval,MigrationFraction,...
    CrossoverFraction);


Sample=[Class1TestData;Class2TestData;Class3TestData;Class4TestData;...
    Class5TestData;Class6TestData;Class7TestData;Class8TestData;...
    Class9TestData;Class10TestData];
Training=[D1;D2;D3;D4;D5;D6;D7;D8;D9;D10];
[Rows,Colls]=size(D1);
Group=[ones(Rows,1)*1;ones(Rows,1)*2;ones(Rows,1)*3;ones(Rows,1)*4;...
    ones(Rows,1)*5;ones(Rows,1)*6;ones(Rows,1)*7;ones(Rows,1)*8;...
    ones(Rows,1)*9;ones(Rows,1)*10];
G=[ones(Fold,1)*1;ones(Fold,1)*2;ones(Fold,1)*3;ones(Fold,1)*4;...
    ones(Fold,1)*5;ones(Fold,1)*6;ones(Fold,1)*7;ones(Fold,1)*8;...
    ones(Fold,1)*9;ones(Fold,1)*10];
Class=knnclassify(Sample,Training,Group,k);
[C,order] = confusionmat(G,Class);
Call=[Call;C];
OverallC=OverallC+C;

end
```

```
OverallSum=sum(diag(OverallC));
Accuracy=(OverallSum/1000)*100;
[' Test set Accuracy ',num2str(Accuracy),' % ']


end
```

Α΄.14.3 Επαναλήψεις με καλύτερες παραμέτρους

```
%% C12345678910 Test
PopulationSize=80;
emax=0.1;
CrossoverFraction = 1;
MigrationFraction = 0.7;
MigrationInterval = 15;
k=3;
EliteCount=6;
AccuracyAll=[];
Matrixs=[];
pame=[];
matlabpool(4)
parfor i=1:10
[Accuracy]=Features_Airs(PopulationSize,EliteCount,MigrationInterval,...
    MigrationFraction,CrossoverFraction,k,emax);
Matrixs=[Matrixs;i];
AccuracyAll=[AccuracyAll;Accuracy];
end
pame=[Matrixs AccuracyAll];
figure('Name','Accuracy and Iterations C12345678910','NumberTitle','off');
plot(pame(:,1),pame(:,2),'--rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerSize',10)
xlabel('emax');
ylabel('%Accuracy');
a=sum(AccuracyAll)/10;
fprintf('Accuracy : %d\n', a);
```