ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήση γενετικών αλγορίθμων για τη δημιουργία ομάδων φοιτητών για συνεργασία

Ηλίας Μαστροσαββάκης
Α.Μ. 21013

Εισηγητής: Χρήστος Τρούσσας, Επίκουρος Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήση γενετικών αλγορίθμων για τη δημιουργία ομάδων φοιτητών για συνεργασία

Ηλίας Μαστροσαββάκης
Α.Μ. 21013

Εισηγητής:

Χρήστος Τρούσσας, Επίκουρος Καθηγητής

Εξεταστική Επιτροπή:

Χρήστος Τρούσσας, Επίκουρος Καθηγητής

Christos Troussas Digitally signed by Christos Troussas
Date: 2023.07.19 12:14:44 +03'00'

Ιωάννης Βογιατζής, Καθηγητής

Ioannis Voyiatzis Digitally signed by Ioannis Voyiatzis
Date: 2023.08.31 12:24:21 +03'00'

Φοίβος Μυλωνάς, Αναπληρωτής Καθηγητής

Φοίβος-Απόστολος Μυλωνάς Ψηφιακά υπογεγραμμένο
2023.07.19 23:06:03 +03'00'

Ημερομηνία εξέτασης 17/07/2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου. Επιπλέον βεβαιώνω ότι έχω αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για την συγκεκριμένη διπλωματική εργασία.

Ο Δηλών
Ηλίας Μαστροσαββάκης

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα πολύ ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό των γενετικών αλγορίθμων και του προγραμματισμού σε γλώσσα python. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου κ. Χρήστος Τρούσσας, τον οποίο θα ήθελα να ευχαριστήσω θερμά.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένειά μου για τη συμπαράστασή της κατά τη διάρκεια των σπουδών μου και να αφιερώσω την παρούσα εργασία στην μνήμη του πατέρα μου.

ΠΕΡΙΛΗΨΗ

Ο σχηματισμός ομάδων εργασίας φοιτητών παίζει σημαντικό ρόλο στην αποτελεσματικότητα της μαθησιακής διαδικασίας. Κατάλληλες ομάδες φοιτητών οδηγούν σε αλληλεπιδράσεις μεταξύ τους και αυξάνουν τα μαθησιακά αποτελέσματα. Ωστόσο, ο σχηματισμός ομάδας είναι ένα σύνθετο έργο και απαιτεί αυτόματες προσεγγίσεις που θα παράγουν τα βέλτιστα αποτελέσματα σε σύντομο χρονικό διάστημα. Προς αυτή την κατεύθυνση, η διπλωματική αυτή παρουσιάζει έναν νέο γενετικό αλγόριθμο για την ομαδοποίηση των φοιτητών με σκοπό την εκμάθησή τους με εργασίες. Οι καινοτομίες του αφορούν τις ιδιότητες που χρησιμοποιούνται για τη σύνθεση ομάδων και γενετικούς τελεστές. Ειδικότερα, τα χαρακτηριστικά του φοιτητή αναφέρονται στις τρεις κύριες διαστάσεις της μάθησης, ακαδημαϊκό, γνωστικό και κοινωνικό. Όσον αφορά τους γενετικούς τελεστές, ο αλγόριθμος εκτελεί δύο τελεστές διασταύρωσης, μια τροποποιημένη διασταύρωση δύο σημείων και μια νέα προσέγγιση, που ονομάζεται διασταύρωση σε ένα σημείο ανά ομάδα.

Λέξεις Κλειδιά: Συνεργατική μάθηση, Γενετικός Αλγόριθμος, Σχηματισμός Ομάδας, Ομαδοποίηση με πολλαπλά κριτήρια.

ABSTRACT

The group formation plays an important role to the effectiveness of learning process. Adequate groups foster student interactions and increase learning outcomes. However, group formation is a complex task and requires automatic approaches to produce the optimal results in short time. To this direction, this project presents a novel genetic algorithm for student grouping to facilitate learning process with projects. Its innovations pertain to the attributes used for the composition of groups and genetic operators applied. In particular, student attributes refer to the three main dimensions of learning, academic, cognitive, and social. Regarding genetic operators, the algorithm performs two crossover operators, a modification of two-point crossover and a new approach, called one-point per group crossover.

Index Terms: Collaborative learning, genetic algorithm, group formation, multi-criteria grouping.

Πίνακας Περιεχομένων

Κεφάλαιο 1

1. Εισαγωγή

1.1 Η χρησιμότητα των ομάδων σε φοιτητικές εργασίες

Είναι γενικά γνωστό σε όλους ότι η διαδικασία της μάθησης –όποια και εάν είναι αυτή- και αφορά την ιδιαίτερη κατηγορία των φοιτητών, είναι ζωτικής σημασίας για την μελλοντική τους εξέλιξη και σταδιοδρομία. Αυτό που απαιτείται σε μεγάλο βαθμό και καθορίζει την δυνατότητα των αποφοίτων να απορροφηθούν στους διάφορους κλάδους εργασίας είναι η αποτελεσματική συνεργασία τους με αλλά άτομα, δεδομένου ότι μπορούν να αποτελέσουν υπάλληλοι ή/και ανώτερα στελέχη μιας επιχείρησης. Η ικανότητα των ατόμων να εργάζονται σε ένα ομαδικό περιβάλλον αποτελεσματικά αξιολογείται καλύτερα και εκτιμάται περισσότερο από την γνώση του επαγγελματικού πεδίου, αφού οι σύγχρονες πλέον επιχειρήσεις θέλουν να πετύχουν το βέλτιστο δυνατό αποτελέσματα σε σύντομο χρονικό διάστημα.

Γι’ αυτό το λόγο οι περισσότεροι ακαδημαϊκοί καθηγητές προκειμένου να βοηθήσουν τους σπουδαστές τους αλλά επιπλέον και για να τους διευκολύνουν στην διαδικασία μάθησης που είναι ιδιαίτερα απαιτητική, χρησιμοποιούν διάφορους τρόπους ώστε οι φοιτητές να αλληλεπιδρούν και να συνεργάζονται μεταξύ τους. Ένας από αυτούς τους τρόπους είναι οι ομαδικές εργασίες. Αυτές χρειάζονται προκειμένου οι φοιτητές να αφομοιώσουν καλύτερα και ευκολότερα την ύλη του μαθήματος, και ταυτοχρόνως δύνανται να αναπτύξουν την ικανότητα της αποτελεσματικής συνεργασίας. Οι καθηγητές με τους βοηθούς τους δημιουργούν εργασίες κατάλληλες κυρίως για μικρές ομάδες -συνήθως τριών- φοιτητών.

Προκειμένου να σχηματιστούν οι κατάλληλες ομάδες φοιτητών ώστε να φέρουν όλες σε πέρας την εργασία, είναι αναγκαίο να ληφθούν υπόψη πολλές παράμετροι, για την σωστή επιλογή κάθε μέλους ομάδας. Οι καθηγητές δεν θα ήταν εύκολο να κάνουν κάτι τόσο χρονοβόρο. Έτσι λοιπόν το μόνο που θα τους εξυπηρετούσε είναι ένα αυτοματοποιημένο σύστημα που να χρησιμοποιεί κάποιον ή κάποιους αλγορίθμους που θα δέχονται ένα πλήθος παραμέτρων για κάθε φοιτητή και θα τους εντάσσουν, όσο είναι δυνατόν στην καταλληλότερη ομάδα.

Σκοπός της παρούσας μεταπτυχιακής διπλωματικής είναι η δημιουργία ενός αλγορίθμου σχηματισμού ομάδων εργασίας φοιτητών, με την χρήση γενετικών αλγορίθμων. Η σπουδαιότητα των γενετικών αλγορίθμων στο σχηματισμό ομάδων θα αναλυθεί στην παρακάτω υποενότητα.

1.2 Η σπουδαιότητα γενετικών αλγορίθμων στον σχηματισμό ομάδων

Γενικά, μπορεί κάποιος να δει τα προβλήματα ομαδοποίησης ως εκείνα των οποίων ο στόχος είναι να ομαδοποιηθούν τα μέλη ενός συνόλου σε μία ή περισσότερες ομάδες αντικειμένων, με κάθε αντικείμενο να βρίσκεται σε μια ακριβώς ομάδα, δηλαδή να βρεθεί μια ομαδοποίηση αυτών των αντικειμένων.

Στα περισσότερα από αυτού του είδους τα προβλήματα, δεν επιτρέπονται όλοι οι πιθανοί συνδυασμοί ομαδοποιήσεις, διότι μια λύση στο πρόβλημα πρέπει να συμμορφώνεται με ένα πλήθος από συγκεκριμένους περιορισμούς, διαφορετικά η λύση δεν είναι αποδεκτή. Συνήθως ένα αντικείμενο δεν μπορεί να ομαδοποιηθεί με όλα τα πιθανά υποσύνολα των υπόλοιπων αντικειμένων. Δηλαδή τα προβλήματα ομαδοποίησης είναι μια σημαντική κατηγορία δύσκολων προβλημάτων βελτιστοποίησης. Ο στόχος της ομαδοποίησης είναι να βελτιστοποιήσει μια συνάρτηση κόστους ή συνάρτηση καταλληλότητας που ορίζεται επάνω στο σύνολο όλων των ομαδοποιήσεων που ικανοποιούν τους προαναφερόμενους περιορισμούς.

Πολλά προβλήματα ομαδοποίησης είναι NP-hard (βλέπε Garey and Johnson (1979)), που σημαίνει ότι οποιοσδήποτε γνωστός ακριβής αλγόριθμος θα εκτελεστεί σε χρόνο εκθετικό ως προς το μέγεθος των δεδομένων του προβλήματος. Ένας τέτοιος αλγόριθμος είναι επομένως στις περισσότερες περιπτώσεις άχρηστος για προβλήματα πραγματικού μεγέθους και όχι ενδεικτικού. Ως εκ τούτου οι γενετικοί αλγόριθμοι (GA) καθίστανται υποψήφιοι για την επίλυση των προβλημάτων ομαδοποίησης αφού οδηγούν σε ενθαρρυντικά αποτελέσματα.

Κατά καιρούς και μέχρι σήμερα έχουν αναπτυχθεί πολλοί αλγόριθμοι ομαδοποίησης. Κάποιοι από αυτούς είναι διαμελιστικοί αλγόριθμοι ομαδοποίησης όπως π.χ. οι K-means, k-medoids, fuzzy C, DBSCAN, ενώ άλλοι είναι αλγόριθμοι Ιεραρχικής ομαδοποίησης π.χ. οι BIRCH , Agglomerative Hierarchy. Τέλος υπάρχουν και γενετικοί αλγόριθμοι ομαδοποίησης (GGA) που βασίζονται στην χρήση των γενετικών αλγορίθμων.

Οι αλγόριθμοι που ανήκουν στην κατηγορία των διαμελιστικών αλγορίθμων όπως ο K-means, χρησιμοποιούν μια συνήθης τεχνική ομαδοποίησης βασιζόμενη στην μέθοδο διαχωρισμού σε ομάδες, η οποία προσπαθεί να βρει έναν καθορισμένο από τον χρήστη αριθμό συστάδων (k). Οι συστάδες αυτές αντιπροσωπεύονται από τα κέντρα τους, ελαχιστοποιώντας τη συνάρτηση τετράγωνου του σφάλματος. Αν και αυτοί οι αλγόριθμοι είναι απλοί και μπορούν να χρησιμοποιηθούν σε μια μεγάλη ποικιλία τύπων δεδομένων, είναι αρκετά ευαίσθητοι όσον αφορά την επιλογή των αρχικών θέσεων των κέντρων των συστάδων. Υπάρχουν δύο απλές προσεγγίσεις για την αρχικοποίηση του κέντρου συμπλέγματος, δηλαδή είτε τυχαία επιλογή των αρχικών τιμών είτε να επιλεγούν τα πρώτα k

δείγματα των σημείων δεδομένων. Και οι δύο προσεγγίσεις προκαλούν τη σύγκλιση των αλγόριθμων αυτών σε υποβέλτιστες λύσεις. Δηλαδή μπορούν να κολλήσουν σε μη βέλτιστες λύσεις, ανάλογα με την επιλογή των αρχικών κέντρων συμπλέγματος. Όντας μέθοδοι επαναληπτικές και αναρριχητικές, είναι αρκετά ευαίσθητες στις αρχικές θέσεις των κέντρων των συστάδων-ομάδων. Επιπλέον, δεδομένου ότι οι σχετικές συναρτήσεις κόστους είναι μη γραμμικές και πολυτροπικές, συνήθως αυτοί οι αλγόριθμοι συγκλίνουν σε ένα τοπικό ελάχιστο, δηλαδή παράγουν διαφορετικά συμπλέγματα για διαφορετικά σύνολα τιμών των αρχικών κέντρων.

Από την άλλη πλευρά, ο γενετικός αλγόριθμος, μία τεχνική παράλληλης αναζήτησης και ένας από τους ευρέως χρησιμοποιούμενους εξελικτικούς αλγόριθμους εκτελεί μία γενικευμένη αναζήτηση για να βρει τη λύση σε ένα πρόβλημα ομαδοποίησης, αφού αναζητά μια συνολική κατά προσέγγιση λύση στα προβλήματα ομαδοποίησης μέσω της εφαρμογής των αρχών της εξελικτικής βιολογίας. Ο αλγόριθμος τυπικά ξεκινά με ένα σύνολο τυχαία δημιουργούμενων ατόμων που ονομάζονται πληθυσμός και δημιουργεί διαδοχικές, νέες γενιές του πληθυσμού με γενετικές λειτουργίες όπως η φυσική επιλογή, η διασταύρωση και η μετάλλαξη. Κάθε χρωμόσωμα του πληθυσμού αντιπροσωπεύει τον K αριθμό των κέντρων. Τα βήματα του γενετικού αλγορίθμου εφαρμόζονται επανειλημμένα για έναν αριθμό γενεών προς αναζήτηση κατάλληλων κέντρων συστάδας στο χώρο των χαρακτηριστικών, έτσι ώστε να βελτιστοποιείται η τιμή μίας συνάρτησης που αντιπροσωπεύει την ομοιότητα των συστάδων που προκύπτουν. Το μεγαλύτερο πλεονέκτημα των γενετικών αλγορίθμων είναι ότι η συνάρτηση καταλληλότητας μπορεί να τροποποιηθεί για να αλλάξει τη συμπεριφορά του αλγορίθμου.

Ενώ ο γενετικός αλγόριθμος (GA) είναι μια τυχαιοποιημένη τεχνική αναζήτησης και βελτιστοποίησης που καθοδηγείται από τις αρχές της εξέλιξης και της φυσικής γενετικής και έχει μεγάλο βαθμό σιωπηρού παραλληλισμού, παρέχει σχεδόν βέλτιστες λύσεις για τον στόχο που τίθεται σε ένα πρόβλημα μέσω της βελτιστοποίησης της συνάρτησης καταλληλότητας αυτού. Υπό περιοριστικές συνθήκες, μια τεχνική ομαδοποίησης που βασίζεται σε γενετικό αλγόριθμο (GA) αναμένεται να παρέχει μια βέλτιστη ομαδοποίηση, ανώτερη από αυτή του αλγόριθμου K-Means, αλλά με λίγη περισσότερη πολυπλοκότητα χρόνου.

Για όλους τους παραπάνω λόγους παρουσιάζουμε σε αυτή την διπλωματική έναν Γενετικό Αλγόριθμο Ομαδοποίησης (GGA), ο οποίος είναι ένας Γενετικός Αλγόριθμος (GA) τροποποιημένος για να ταιριάζει στη δομή του προβλήματος ομαδοποίησης φοιτητών σε εργασίες.

Κεφάλαιο 2

2. Συνεργατική μάθηση και συστήματα σχηματισμού ομάδων

Η μάθηση διευκολύνεται όταν οι σπουδαστές έχουν την ευκαιρία να αλληλοεπιδρούν (interact) και να συνεργάζονται με τους άλλους σε ομαδικές εργασίες, και με αυτόν τον τρόπο να κατανοούν και να σέβονται τη διαφορετικότητα που υπάρχει μεταξύ τους. Αυτή η κατανόηση, σύμφωνα με τις αρχές της μαθητοκεντρικής προσέγγισης (APA, 1997), αυξάνει την ευελιξία στη σκέψη, τις κοινωνικές δεξιότητες και την ηθική ανάπτυξη των μαθητών – φοιτητών. Εξάλλου, η ικανότητα των ατόμων να εργάζονται αποτελεσματικά σε ένα ομαδικό περιβάλλον, αναφέρεται συνεχώς ως μία από τις κύριες ιδιότητες – δεξιότητες των εργαζομένων, που αναζητούν οι επιχειρήσεις, και συχνά αξιολογείται υψηλότερα και από τη γνώση του επαγγελματικού πεδίου. Το κλειδί για τη συνεργασία είναι η αποτελεσματική επικοινωνία και το κλειδί για την αποτελεσματική επικοινωνία είναι η επίτευξη αυτογνωσίας και η κατανόηση των άλλων (συνεργατών). Ο σπουδαστής που μαθαίνει να κατανοεί και να εκτιμά όλους τους μαθησιακούς τύπους, προσαρμόζεται ευκολότερα στις νέες προκλήσεις που αντιμετωπίζει στις σπουδές του, στη δουλειά του και στις προσωπικές του σχέσεις.

2.1 Θεωρητική προσέγγιση της συνεργατικής μάθησης

Ο όρος συνεργασία (collaboration / cooperation) στην εκπαίδευση θα μπορούσε να οριστεί ως μία κατάσταση στην οποία οι μαθητές αλληλοεπιδρούν μεταξύ τους, και αυτές οι αλληλεπιδράσεις επηρεάζουν την γνωστική διαδικασία, με δεδομένο ότι είναι ίσοι ή καλύτερα ομότιμοι μεταξύ τους (peers) (Dillenbourg, 1999). Γενικότερα, ο όρος περιλαμβάνει την ικανότητα των ατόμων να εργαστούν αποτελεσματικά με άλλους, συμπεριλαμβανομένων και αυτών από διαφορετικές ομάδες και με αντίθετες απόψεις (American Management Association, 2010). Ο όρος ομάδα (group/team) αναφέρεται σε μία ανεξάρτητη συλλογή ατόμων που μοιράζονται την ευθύνη για συγκεκριμένα αποτελέσματα για την εκπαίδευση ή τις επιχειρήσεις (Sundstrom, DeMeuse & Futrell, 1990). Στη διεθνή βιβλιογραφία απαντώνται δύο αγγλικές λέξεις που περιγράφουν την έννοια της ομάδας εργασίας, work group και work team, και αν και μερικοί συγγραφείς τις διακρίνουν, οι Sundstrom, McIntyre, Halfhill & Richards (2000) υποστηρίζουν ότι η διάκριση δεν είναι ούτε συνεπής ούτε ευρέως αναγνωρίσιμη και έτσι χρησιμοποιούν τον ίδιο ορισμό και για τους δύο όρους. Επίσης, στη διεθνή βιβλιογραφία, απαντώνται δύο αγγλικοί όροι για την έννοια της συνεργασίας, collaboration και cooperation, οι οποίοι πολλές φορές χρησιμοποιούνται σα συνώνυμοι, ενώ άλλες φορές είναι διακριτοί, ανάλογα με το βαθμό επιμερισμού της εργασίας (Dillenbourg, 1999). Στη συνεργασία με τον όρο cooperation, οι συνεργάτες διαχωρίζουν την εργασία, επιλύουν τις επιμέρους εργασίες ατομικά και έπειτα συγκεντρώνουν μαζί τα μερικά

αποτελέσματα σε ένα τελικό, ενώ με τον όρο collaboration οι συνεργάτες εκτελούν την εργασία μαζί (Dillenbourg, 1999). Όμως, ακόμη και όταν δύο άνθρωποι εργάζονται από κοινού, κάποιος διαχωρισμός στην εργασία μπορεί να συμβεί (Miyake, 1986).

Ο γενικότερος ορισμός της συνεργατικής μάθησης (collaborative learning) αφορά μία κατάσταση στην οποία δύο ή περισσότεροι άνθρωποι μαθαίνουν ή επιχειρούν να μάθουν κάτι μαζί (Dillenbourg, 1999). Η συνεργατική μάθηση είναι μια διαδικασία κοινωνικοποίησης που εμπλέκει τις αλληλεπιδράσεις των ατόμων σε ένα ομαδικό πλαίσιο.

Η εκπαιδευτική έρευνα έχει πολλάκις τονίσει τα οφέλη της συνεργατικής μάθησης. Σύμφωνα με ερευνητές του χώρου, όταν οι μαθητές εμπλέκονται ενεργά σε συνεργατικές δραστηριότητες τείνουν να μαθαίνουν καλύτερα και περισσότερα από αυτά που διδάσκονται, τα αφομοιώνουν περισσότερο από τη συμβατική διδασκαλία και εμφανίζονται πιο ικανοποιημένοι από τα αποτελέσματα που οι δημιουργούν οι ίδιοι (πχ. Dillenbourg, 1999; Gross Davis, 1993). Έρευνες έχουν δείξει ότι οι ομαδικές μαθησιακές δραστηριότητες βελτιώνουν τους μαθητές διανοητικά (Petress, 2004) και προωθούν ανώτερες γνωστικές ικανότητες, όπως αναλυτική τεκμηρίωση και σύνθεση πολλαπλών ροών πληροφοριών, σε ένα σύνολο που είναι μεγαλύτερο από το άθροισμα των μερών του (Johnson & Johnson, 1990). Επίσης, η ομαδική δραστηριότητα μπορεί να βελτιώσει την ποιότητα του έργου και τις επιδόσεις της ομάδας (Soliman & Okba, 2006), καθώς μπορεί να οδηγήσει σε πιο βαθιά ανάλυση ή κριτική των τελικών λύσεων (Dillenbourg, Baker, Blaye & O'Malley, 1995). Επιπλέον, σύμφωνα με τον Petress (2004), η ομαδική εργασία οδηγεί σε περισσότερη ποικιλία και δημιουργικότητα και, όταν γίνεται αποτελεσματικά, διεγείρει το ενδιαφέρον και αυξάνει την αυτοπεποίθηση.

Η συνεργατικότητα βοηθά επίσης τους μαθητές να αναπτύξουν κοινωνικές δεξιότητες, όπως η επικοινωνία και η αποσαφήνιση των ιδεών, η υπευθυνότητα και η αλληλεπίδραση με τους άλλους (Schlichter, 1997). Οι συμμετέχοντες σε μια ομάδα μπορούν να μαθαίνουν από την εμπειρία τους να πετύχουν έναν ομαδικό στόχο, και έτσι να βελτιώσουν τις συνεργατικές τους δεξιότητες ώστε να μπορούν να αντιμετωπίσουν μελλοντικές συνεργασίες (Jules, 2007). Σύμφωνα με ερευνητές, το περιβάλλον της συνεργατικής μάθησης, παραλληλίζοντας στενά τις εμπειρίες της ζωής, προετοιμάζει κατάλληλα τους μαθητές για να συναντήσουν επιτυχώς τις προκλήσεις στην μετέπειτα εργασιακή τους ζωή (McLoughlin & Luca, 2002). Γενικά, σύμφωνα με τους Johnson και Johnson (1990), η συνεργασία, συγκρινόμενη με τις ατομικές και ανταγωνιστικές προσπάθειες, έχει σαν αποτέλεσμα (α) υψηλές επιδόσεις και μεγαλύτερη παραγωγικότητα, (β) περισσότερο υποστηρικτικές σχέσεις και (γ) μεγαλύτερη ψυχολογική υγεία, κοινωνική επάρκεια και αυτοεκτίμηση. Επιπλέον επιτυγχάνεται (δ) εποικοδομητική κριτική και αυτοκριτική , όταν οι μαθητές μαθαίνουν να συνεργάζονται (Miyake, 1986).

Παρόλα αυτά, μερικοί μαθητές αισθάνονται άβολα με τις συνεργατικές μαθησιακές δραστηριότητες και συχνά τις αντιμετωπίζουν σαν απειλή για τις επιδόσεις τους (Ellis & Hafner, 2008). Πολλοί μαθητές υπολείπονται των κοινωνικών δεξιοτήτων που είναι προϋπόθεση για την επιτυχία σε συνεργατικές δραστηριότητες, και η προσαρμογή τους στις νέες απαιτήσεις μπορεί να είναι «απειλητική» γι' αυτούς (Herreid, 1998). Σύμφωνα με τον Petress (2004), υπάρχουν κάποιες προϋποθέσεις που θα πρέπει να ισχύουν για την επιτυχημένη συνεργασία μίας ομάδας. Δηλαδή: (1) ο κύριος λόγος που οι μαθητές σχηματίζουν μια ομάδα πρέπει να είναι η μάθηση, (2) οι μαθητές οφείλουν να είναι υπεύθυνοι και να παρακολουθούν συχνά τις ομαδικές συναντήσεις, να είναι στην ώρα τους, προετοιμασμένοι και με διάθεση για εργασία, (3) οι μαθητές πρέπει να είναι πρόθυμοι και ικανοί να συμμετέχουν ενεργά στις εργασίες της ομάδας και (4) θα πρέπει να είναι ανεκτικοί με τις ιδέες των άλλων, τους μαθησιακούς τους τύπους και τις απόψεις τους. Μία παρόμοια αντίληψη έχει εκφράσει και ο Hoffman (1959), ο οποίος υποστηρίζει ότι η ποικιλομορφία των απόψεων θα πρέπει να συνοδεύεται και από ανεκτικότητα στις διαφορετικές απόψεις, προκειμένου η ομάδα να διερευνήσει τη δυνητική της δημιουργικότητα. Επίσης, σύμφωνα με τον Petress (2004), η επίγνωση της διαφορετικότητας, η ανεκτικότητα και η αποδοχή είναι μερικά επιπλέον κέρδη από την ομαδική συνεργασία.

2.2 Σχηματισμός ομάδων

Από τα παραπάνω γίνεται φανερό ότι η ενθάρρυνση των μαθητών να εργαστούν μαζί δεν οδηγεί από μόνη της σε παραγωγική συνεργασία. Σύμφωνα με ερευνητές, η παραγωγικότητα και η επιτυχία μίας ομάδας μάθησης καθορίζεται από τους ρόλους και τις αλληλεπιδράσεις των μελών της και από το πόσο καλά συνεργάζονται (Martin & Paredes, 2004; Webster & Sudweeks, 2006). Ειδικότερα, ο τρόπος που οι μαθητές ομαδοποιούνται μπορεί να επηρεάσει τα αποτελέσματα της μαθησιακής εμπειρίας (Johnson & Johnson, 1990; Slavin, 1995; Webster & Sudweeks, 2006). Ως εκ τούτου, ο κατάλληλος σχηματισμός της ομάδας (group formation) είναι μία πολύ σημαντική διαδικασία που μπορεί να λύσει πολλά προβλήματα πριν αυτά αναδυθούν (Muehlenbrock, 2006). Επιπλέον η κατάλληλη συγκρότηση της ομάδας μπορεί να αυξήσει την πιθανότητα η αλληλεπίδραση των μελών της να ενεργοποιήσει τους μηχανισμούς μάθησης (Daradoumis, Guitert, Gimenez, Marques & Lloret, 2002). Ο όρος μηχανισμός μάθησης (learning mechanism) αναφέρεται στην ενεργοποίηση από τους μαθητές συγκεκριμένων γνωστικών διαδικασιών επεξεργασίας της πληροφορίας (όπως επαγωγή (induction), απαγωγή (deduction), επεξεργασία (compilation) κλπ). Η αλληλεπίδραση των μαθητών σε συνεργατικά περιβάλλοντα αναμένεται να δώσει ώθηση σε μηχανισμούς μάθησης και να προάγει τη γνώση (Dillenbourg, 1999). Επειδή όμως δεν υπάρχει εγγύηση ότι τελικά θα συμβούν οι αναμενόμενες αλληλεπιδράσεις, θα πρέπει να

δημιουργηθούν οι κατάλληλες αρχικές συνθήκες που αφορούν στο μέγεθος της ομάδας και στην επιλογή των μελών της (Dillenbourg, 1999).

2.2.1 Μέθοδοι σχηματισμού ομάδων

Η έρευνα πάνω στη συνεργατική μάθηση εξετάζει τελευταία τεχνικές οργάνωσης ομάδων, ώστε να επιτευχθούν τα καλύτερα μαθησιακά οφέλη, ακαδημαϊκά και κοινωνικά. Οι τρεις μέθοδοι σχηματισμού ομάδων που συχνά εφαρμόζονται από τους καθηγητές είναι οι αυτοδημιούργητες ομάδες (self-selected teams), οι τυχαία καθορισμένες ομάδες (randomly assigned teams) και οι καθορισμένες από τον καθηγητή ομάδες (instructor-assigned teams) (Layton et al., 2010).

Οι αυτοδημιούργητες ομάδες, δηλ. οι ομάδες που δημιουργούνται από τους ίδιους τους μαθητές, μπορεί να είναι βολικές για τα μέλη τους, αλλά συνήθως βασίζονται σε φιλίες και είναι αμφίβολο αν είναι αποτελεσματικές. Πρόσφατη εμπεριστατωμένη ανασκόπηση των ερευνών των τριών μεθόδων από τους Layton et al. (2010) αναφέρει ότι οι αυτοδημιούργητες ομάδες έχουν πλεονεκτήματα και μειονεκτήματα. Σύμφωνα με τους ερευνητές, αφενός υπάρχουν μελέτες που δείχνουν ότι οι ομάδες αυτές έχουν αυξημένη συνοχή, συνεργατικότητα, υπευθυνότητα και αυξημένη ικανοποίηση των μελών τους (Strong & Anderson, 1990; Neal, 1997; Bacon, Stewart, & Silver, 1999), αφετέρου σε άλλες μελέτες αναφέρεται ότι τα μέλη αυτών των ομάδων, σε μεγάλο ποσοστό, εκφράζονται πολύ αρνητικά για την εμπειρία τους, το μάθημα, τις εργασίες ή τους καθηγητές (Brickell, Porter, Reynolds & Cosgrove, 1994). Επίσης, η αυτό-επιλογή μπορεί να οδηγήσει σε ακραία ομοιογένεια (Jalajas & Sutton, 1985), ώστε να μην συμπεριλαμβάνονται στην ομάδα όλες οι δεξιότητες που απαιτούνται για την ομαδική εργασία (Mello, 1993). Ένας ακόμη κίνδυνος αυτών των ομάδων είναι η ανάπτυξη συμπεριφορών «κλίκας» που διαβρώνουν τη συνοχή της ομάδας και την απόδοσή της (Daly & Worrell, 1993). Οι Cavanaugh, Ellis, Layton & Ardis (2004) υποστηρίζουν ότι υπάρχει ρίσκο στην ελεύθερη επιλογή ομάδων από τους φοιτητές, ειδικότερα σε τάξεις με μικρή εμπειρία και μικρή γνωριμία των φοιτητών μεταξύ τους και ότι αυτή η ελεύθερη επιλογή δυσκολεύει τους μαθητές που δεν έχουν γνωστούς στην τάξη. Η παρέμβαση του καθηγητή, με την εισαγωγή κάποιων περιορισμών, μπορεί να μειώσει τυχόν μειονεκτήματα των αυτοδημιούργητων ομάδων (Bacon et al., 1999).

Οι τυχαία καθορισμένες ομάδες παρουσιάζουν επίσης πολλαπλά μειονεκτήματα, σύμφωνα με την ανασκόπηση ερευνών από τους Layton et al. (2010). Η τυχαία κατανομή των μαθητών δε λαμβάνει άμεσα υπόψη την κοινωνική δυναμική των ομάδων (Gogoulou, Gouli, Boas, Liakou & Grigoriadou, 2007). Επίσης, δεν έχει απαραίτητα σαν αποτέλεσμα την ποικιλομορφία, τις εξισορροπημένες δεξιότητες ή την ανάμειξη των προσωπικοτήτων (Cook, 1981). Ακόμη, εγείρει θέματα δικαιοσύνης και πιθανόν να μην αποτελεί θετική εμπειρία για

τους μαθητές (Bacon et al., 1999). Οι καθηγητές συχνά χρησιμοποιούν την τυχαία κατανομή για εργασίες μικρής διάρκειας ή για ερευνητικούς σκοπούς.

Τέλος, ο καθοδηγούμενος σχηματισμός ομάδων θεωρείται ιδιαίτερα χρήσιμος στην περίπτωση που οι μαθητές δε γνωρίζονται μεταξύ τους ή όταν τείνουν να σχηματίζουν ομάδες που δεν οδηγούν σε καρποφόρες συνεργασίες (Alfonseca et al., 2006). Οι καθορισμένες από τον καθηγητή ομάδες επιτρέπουν στους καθηγητές να ελέγχουν διάφορα κριτήρια σχηματισμού τους, σε μία προσπάθεια να δημιουργήσουν θετικές μαθησιακές εμπειρίες στους εκπαιδευόμενους, και έρευνες επιβεβαιώνουν ότι ο έλεγχος των κριτηρίων βελτιώνει τις επιδόσεις των μαθητών (πχ. Oakley et al., 2004). Παρόλα τα πλεονεκτήματα της κατανομής των μαθητών σε ομάδες με βάση κάποια κριτήρια, οι καθηγητές συχνά δεν προβαίνουν σε αυτή τη μέθοδο, λόγω της πολυπλοκότητας και του χρόνου που απαιτείται, ειδικά όταν το μέγεθος της τάξης είναι μεγάλο ή αυξάνεται ο αριθμός των μεταβλητών (Layton et al., 2010).

Ο σχηματισμός ομάδων με τη βοήθεια υπολογιστή (Computer-aided team formation)) αποτελεί μία σημαντική λύση στο παραπάνω πρόβλημα, καθότι αφενός επιτρέπει στον καθηγητή να ελέγχει τη διαδικασία σχηματισμού ομάδων, θέτοντας δικά του κριτήρια, και αφετέρου μειώνει το χρόνο που απαιτείται. Προκειμένου να διευκολυνθεί ο καθηγητής στη διαδικασία κατανομής μαθητών σε ομάδες, τα τελευταία χρόνια έχουν αναπτυχθεί διάφορα συστήματα σχηματισμού ομάδων.

Με βάση τις παραπάνω μελέτες, προκύπτει το συμπέρασμά ότι ένα ενδιαφέρον πεδίο έρευνας για την υποστήριξη της συνεργατικής μάθησης των εκπαιδευόμενων, είναι ο καθοδηγούμενος σχηματισμός ομάδων με τη βοήθεια υπολογιστή.

2.2.2 Μαθησιακά χαρακτηριστικά για το σχηματισμό ομάδων

Ένα σημαντικό θέμα στη δημιουργία ομάδων είναι ο προσδιορισμός συγκεκριμένων χαρακτηριστικών των μαθητών. Πολλοί παράγοντες έχουν ερευνηθεί για την επιρροή τους στη δυναμική των ομάδων και στις επιδόσεις τους. Συνήθεις πρακτικές σχηματισμού ομάδων βασίζονται στις ικανότητες ή στις επιδόσεις των μαθητών με βάση ένα αρχικό τεστ. Όμως, ερευνητές του πεδίου δίνουν έμφαση στη σπουδαιότητα των προσωπικών χαρακτηριστικών (προσωπικότητας ή κοινωνικών) στη σύνθεση των ομάδων (Bradley & Herbert, 1997; Martin & Paredes, 2004). Προτείνουν ότι εκτός από το επίπεδο γνώσεων των μαθητών, κοινωνικά χαρακτηριστικά (όπως το φύλο, η εθνικότητα), το κίνητρο, τα ενδιαφέροντα, στοιχεία προσωπικότητας (όπως οι μαθησιακοί τύποι) και το μαθησιακό πλαίσιο, θα έπρεπε να λαμβάνονται υπόψη στη διαδικασία σχηματισμού ομάδων (ποικιλομορφία (diversity)).

Έρευνα αποκαλύπτει ότι ενώ τα δημογραφικά χαρακτηριστικά όπως η ηλικία, η φυλή, το γένος, προκαλούν διαδικασίες κοινωνικής κατηγοριοποίησης, που μπορεί να οδηγήσουν σε δυσμενείς ομαδικές διαδικασίες και αποτελέσματα, τα ψυχολογικά χαρακτηριστικά όπως οι μαθησιακοί τύποι, μπορούν να προκαλέσουν ποικιλομορφία στη σκέψη και να οδηγήσουν σε ενισχυμένη απόδοση της ομάδας (Jules, 2007). Ειδικότερα, όπως πολλοί ερευνητές υποστηρίζουν, οι μαθησιακοί τύποι φαίνεται να αποτελούν ένα αξιόπιστο εργαλείο για το σχηματισμό ομάδων (πχ. Martin & Paredes, 2004; Alfonseca et al., 2006; Grigoriadou et al., 2006; Muehlenbrock, 2006; Al-Dujaily & Ryu, 2007; Wang et al., 2007).

2.2.3 Κατάλληλο πρότυπο (pattern) για το σχηματισμό ομάδων – Η σημασία της ετερογένειας

Ένα σημαντικό θέμα στο σχηματισμό ομάδων είναι ο προσδιορισμός του κατάλληλου προτύπου (ομοιογενούς ή ετερογενούς) για το σχηματισμό ομάδων. Μελετώντας τη βιβλιογραφία, παρότι έχει αναφερθεί ότι μερικές φορές οι ομοιογενείς ομάδες (homogeneous groups) (όσον αφορά τις ικανότητες, τις εμπειρίες και τα ενδιαφέροντα) τείνουν να είναι καλύτερες στην επίτευξη συγκεκριμένων στόχων (Johnson & Johnson, 1994), στην πλειονότητα των μελετών, οι ετερογενείς ομάδες (heterogeneous groups) είχαν καλύτερες επιδόσεις από τις ομοιογενείς σε ένα ευρύ πεδίο εργασιών (πχ. Johnson & Johnson, 1994; Alfonseca et al., 2006; De Faria, Adan-Coello & Yamanaka, 2006; Grigoriadou et al., 2006; Al-Dujaily & Ryu, 2007; Wang et al., 2007).

Σύμφωνα με τους Jackson et al. (2003), μία ποικιλία από παράγοντες του ευρύτερου πλαισίου (όπως ο τύπος της ίδιας της εργασίας) μπορεί να διαμορφώσει τα αποτελέσματα της ποικιλομορφίας, που παρατηρούνται σε διάφορες έρευνες. Η γενικότερα αποδεκτή υπόθεση είναι ότι τα δυνητικά οφέλη της ποικιλομορφίας στην απόδοση της ομάδας, είναι μεγαλύτερα όταν η εργασία (task) απαιτεί δημιουργικότητα και καινοτομία, ενώ αντίθετα όταν η εργασία είναι ρουτίνα, ή όταν ο στόχος είναι η ταχύτητα, η ποικιλομορφία μιας ομάδας μπορεί να επηρεάσει την απόδοση (Williams & O'Reilly, 1998). Σύμφωνα με τον Herrmann (1989), οι ετερογενείς ομάδες μπορεί να βιώνουν αρχικά δυσκολίες στην επίτευξη συναίνεσης, αλλά εξαιτίας της ποικιλομορφίας τους μπορεί να είναι ιδανικές για δημιουργικές και καινοτόμες αναθέσεις εργασιών.

Όσον αφορά το μέγεθος των ομάδων, οι έρευνες έχουν δείξει ότι οι επιτυχημένες ομάδες δεν υπερβαίνουν κανονικά τα πέντε μέλη, παρότι εξαιρέσεις μπορεί να υπάρξουν (Petress, 2004). Το πολύ μεγάλο μέγεθος ομάδας ενθαρρύνει άδικες και εξουθενωτικές διαιρέσεις της εργασίας και επιτρέπει σε μερικά μέλη να αναλάβουν όλη την ευθύνη για το έργο.

2.2.4 Έρευνες σχετικά με την ομοιογενή / ετερογενή σύνθεση ομάδων

Όσον αφορά τις επιδόσεις ετερογενών και ομοιογενών ομάδων, ανατρέχοντας στη βιβλιογραφία εντοπίστηκαν έρευνες που συγκρίνουν τα αποτελέσματα ετερογενών και ομοιογενών ομάδων με βάση τους μαθησιακούς τύπους των μελών τους. Πιο αναλυτικά:

Σε μία έρευνα των Alfonseca et al. (2006) με 166 μαθητές που σχημάτισαν ομάδες των δύο, τα αποτελέσματα έδειξαν ότι οι μαθησιακοί τύποι φαίνεται να επηρεάζουν την απόδοση των μαθητών όταν δουλεύουν μαζί. Η ίδια έρευνα έδειξε ότι οι ετερογενείς ομάδες είχαν καλύτερα αποτελέσματα από τις ομοιογενείς.

Σε ότι αφορά στο επίπεδο γνώσεων των μαθητών, μία εμπειρική έρευνα από τους De Faria et al. (2006) που αξιολόγησε ομάδες για συνεργατική μάθηση σε τμήματα προγραμματισμού με βάση τις προγραμματιστικές δεξιότητες των μαθητών, έδειξε ότι η συνεργατική μάθηση ήταν πολύ αποτελεσματική στη βελτίωση των προγραμματιστικών δεξιοτήτων των μαθητών, ειδικότερα γι' αυτούς που εργάστηκαν σε ετερογενείς ομάδες με βάση το επίπεδο γνώσεών τους. Η ίδια μελέτη ανέδειξε τον βασικότατο ρόλο του συντονιστή/μεσολαβητή (moderator) στις αλληλεπιδράσεις της ομάδας.

Ομοίως, μία μελέτη των Al-Dujaily και Ryu (2007) ερεύνησε τους τύπους προσωπικότητας 40 μαθητών που δούλευαν σε ζεύγη σε ένα από απόσταση μαθησιακό περιβάλλον. Τα αποτελέσματα και σε αυτή την έρευνα έδειξαν ότι η μαθησιακή συνεργατική απόδοση των ετερογενών ομάδων ήταν καλύτερη από αυτή των ομοιογενών, με την προϋπόθεση ότι η συνεργασία θα μπορούσε να διευκολυνθεί και να παρακινηθεί αποτελεσματικά από κάποιον εκπαιδευτικό ή άλλο πρόσωπο ως υπεύθυνο ομάδας.

Μία άλλη έρευνα, των Wang et al. (2007), έδειξε ότι οι δώδεκα ετερογενείς ομάδες των τριών μαθητών που δημιουργήθηκαν από υπολογιστή με βάση τους τύπους σκέψης τους, αποδείχθηκαν πιο ικανές να ολοκληρώσουν τα απαιτούμενα από τη συνεργατική δραστηριότητα σε σύγκριση με τις δέκα ομάδες των τριών μαθητών που σχηματίστηκαν με τυχαία επιλογή. Παράλληλα οι ετερογενείς ομάδες παρουσίασαν μικρότερες αποκλίσεις στην απόδοση των μελών τους, υψηλότερα επίπεδα ικανοποίησης από τις στάσεις των άλλων μελών της ομάδας τους και υψηλότερα ομαδικά αποτελέσματα.

Αντίθετα, οι Liu και Tsai (2008) διενήργησαν μία ανάλυση των προτύπων που ακολουθούσαν μικρές ομάδες μαθητών προκειμένου να αλληλοεπιδράσουν σε μία on-line ομαδική δραστηριότητα για την επίλυση προβλήματος. Η ανάλυση έδειξε ότι οι ομάδες, που ήταν ετερογενείς ως προς το επίπεδο δεξιοτήτων, και περιλάμβαναν μέλη υψηλών αποδόσεων, δεν ήταν απαραίτητα και επιτυχείς στην ομαδική εργασία. Επιπλέον, η μελέτη έδειξε ότι πολλές ομάδες χρειάζονταν δασκάλους ή συντονιστές για να υποστηρίξουν τη διαδικασία των αλληλεπιδράσεων και της μάθησης των μελών τους.

Κινούμενα προς αυτή την κατεύθυνση, συστήματα αυτόματης ομαδοποίησης συμπεριέλαβαν καθορισμό των κριτηρίων για τον προσδιορισμό ενός μέλους ως συντονιστή ομάδας (π.χ. Gogoulou et al., 2007).

Συμπερασματικά, και με βάση τις παραπάνω έρευνες, προκύπτει ότι:

• Η ετερογένεια των μαθησιακών τύπων στο σχηματισμό ομάδων βοηθά τις ομάδες να είναι πιο αποδοτικές από τις ομοιογενείς και να έχουν καλύτερα γνωστικά και κοινωνικά αποτελέσματα (Alfonseca et al., 2006; De Faria et al., 2006; Grigoriadou et al., 2006; Al-Dujaily & Ryu, 2007; Wang et al., 2007).

• Η ετερογένεια των μαθησιακών τύπων παράγει διαφορετικές προοπτικές και αυξάνει τη δημιουργικότητα των ομάδων (Herrmann, 1987; Johnson & Johnson, 1994; Kolb, 1999; Robinson, 2001; De Abreu Dos Reis et al., 2007).

• Οι ετερογενείς ομάδες μπορεί να βιώνουν αρχικά δυσκολίες στην επίτευξη συναίνεσης, αλλά εξαιτίας της ποικιλομορφίας τους φαίνεται να είναι ιδανικές για δημιουργικές και καινοτόμες αναθέσεις εργασιών (Herrmann 1989; Williams & O'Reilly, 1998; De Abreu Dos Reis et al., 2007).

• Σχετικά μικρές ομάδες θεωρούνται πιο αποδοτικές (Petress, 2004).

• Σε πολλές περιπτώσεις συστήνεται από τους ερευνητές η ανάδειξη του ρόλου του συντονιστή ομάδας, είτε εξωτερικού είτε από τα μέλη της (De Faria et al., 2006; Al-Dujaily and Ryu, 2007; Gogoulou et al., 2007; Liu και Tsai, 2008).

Με βάση τα παραπάνω, φαίνεται πως αξίζει να ερευνηθεί ο σχηματισμός ομάδων με βάση την ετερογένεια των μαθησιακών τύπων και η επίδρασή του στη συνεργασία των εκπαιδευόμενων. Επίσης, φαίνεται πως θα ήταν βοηθητικό να ενθαρρυνθεί η ανάδειξη συντονιστή από τα μέλη των ομάδων ή και η υποστήριξη των ομάδων από εξωτερικό συντονιστή.

2.3 Συστήματα σχηματισμού ομάδων με την υποστήριξη υπολογιστή

Ο καθοδηγούμενος σχηματισμός ομάδων από τον διδάσκοντα απαιτεί υπερβολικό χρόνο και προσπάθεια, ειδικά για το σχηματισμό ετερογενών ομάδων, επειδή οι συνδυασμοί των χαρακτηριστικών που πρέπει να ληφθούν υπόψη είναι πάρα πολλοί, και σε πολλές περιπτώσεις (πολλών ατόμων ή/και χαρακτηριστικών), είναι σχεδόν αδύνατο να γίνουν «με το χέρι». Οι προσπάθειες των ερευνητών εστιάζουν στην ανάπτυξη εργαλείων βασισμένων σε υπολογιστή (computer-based tools) που να υποστηρίζουν τον αυτόματο σχηματισμό ομάδων βάσει κάποιων χαρακτηριστικών των μαθητών.

Παρακάτω αναφέρονται συνοπτικά έρευνες που αφορούν συστήματα υποστήριξης της συνεργατικής μάθησης με τη βοήθεια υπολογιστή (Computer Supported Collaborative Learning (CSCL) systems).

Οι ερευνητές, προκειμένου να προσδιορίσουν το γνωστικό προφίλ των μαθητών επέλεξαν τρία μοντέλα: (α) το μοντέλο γνωστικών τύπων Riding's Cognitive Styles Analysis (Riding & Rayner, 1998), (β) το μοντέλο μαθησιακών τύπων ASSIST (Tait, Entwistle & McCune, 1998) και (γ) το μοντέλο τύπων προσωπικότητας Myers-Briggs Type Inventory (Myers, McCaulley, Quenk & Hammer, 1998). Το ερευνητικό πλαίσιο περιλάμβανε: (α) την ανάπτυξη και εφαρμογή της έννοιας του γνωστικού προφίλ, (β) τη χρήση του γνωστικού προφίλ των μαθητών σαν ένα πλαίσιο για το δομημένο στοχασμό πάνω στα ατομικά και ομαδικά μαθησιακά χαρακτηριστικά, (γ) τον προσδιορισμό της επιρροής των μαθησιακών χαρακτηριστικών στη δυναμική της ομάδας, (δ) τον προσδιορισμό προφίλ-κλειδιών για την ανάπτυξη επιτυχημένων μαθησιακών ομάδων και (ε) την ανάπτυξη μιας μεθοδολογίας για τη διευκόλυνση της μεταγνωστικής επίγνωσης της ατομικής και ομαδικής μάθησης. Η μεθοδολογία χρησιμοποιήθηκε προκειμένου να ενθαρρυνθούν οι μαθητές στο στοχασμό σχετικά με τη συμμετοχή τους στη διαδικασία του σχεδιασμού και της ανάπτυξης ενός εικονικού ομαδικού μαθησιακού περιβάλλοντος, με ομαδικές συζητήσεις και διαπραγματεύσεις.

Οι ερευνητές προσπάθησαν να αναπτύξουν τεχνολογικά εργαλεία υποστήριξης της συνεργατικής μάθησης προκειμένου να οργανώσουν τους μαθητές σε ομάδες (πχ. Yang et al., 2003), να ανιχνεύσουν τις κατάλληλες συνθήκες για να ενταχθεί ένας μαθητής σε μια ομάδα (Inaba et al., 2000) ή να προτείνουν ομότιμους βοηθούς για συνεργασία (Muehlenbrock, 2006). Επιπλέον οι ερευνητές προσπάθησαν να ενθαρρύνουν το στοχασμό των μαθητών σχετικά με το γνωστικό τους προφίλ και την ομαδική μάθηση (Webster & Sudweeks, 2006). Τα παραπάνω συστήματα υποστήριξης της συνεργατικής μάθησης έχουν αποδειχθεί κατάλληλα σε πολλά πλαίσια, όμως οι μελέτες δεν αναλύουν απαραίτητα πώς οι ομάδες αρχικά σχηματίστηκαν.

Παρακάτω αναφέρεται ένα παράδειγμα συστήματος αυτόματου σχηματισμού ομάδων.

2.3.1 Σύστημα “DIANA” (Wang, Lin & Sun, 2007)

Η δουλειά των Wang, Lin και Sun (2007) βασίζεται σε ένα σύστημα ομαδοποίησης, που λέγεται DIANA, το οποίο χρησιμοποιεί γενετικούς αλγόριθμους για να δημιουργήσει ετερογενείς ομάδες φοιτητών.

Το σύστημα DIANA δημιουργεί ομάδες των 3-7 ατόμων, με βάση τρεις τύπους σκέψης (thinking styles) σύμφωνά με το μοντέλο του Sternberg (1999), τον Εκτελεστικό (Executive), το Νομοθετικό (Legislative) και τον Κριτικό (Judicial). Σύμφωνα με τους

ερευνητές, οι δημιουργημένες ομάδες επιδεικνύουν εσωτερική ποικιλομορφία και εξωτερική ισορροπία με τις άλλες ομάδες. Το σύστημα στην τρέχουσα μορφή του λαμβάνει υπόψη του το πολύ επτά μεταβλητές για να δημιουργήσει ομάδες. Η μέθοδος χρησιμοποιεί μια βελτιωμένη παραλλαγή του αλγόριθμου «Random Mutation Hill Climbing» (Russell & Norvig, 1995), η οποία δημιουργεί κατά το δυνατόν όμοια σε σχήμα και μέγεθος τρίγωνα (ομάδες των τριών φοιτητών) με βάση τις Ευκλείδειες αποστάσεις μεταξύ των ατομικών σκορ των φοιτητών. Η μέθοδος κατανέμει τους φοιτητές γύρω από κέντρα κατηγοριών, τα οποία αναπροσαρμόζονται με την εισαγωγή φοιτητών. Το τριγωνικό σχήμα των τελικών κέντρων κατηγοριών θεωρείται πρωτότυπο και αντιπροσωπεύει τη δομή όλων των ετερογενών ομάδων. Από τη στιγμή που ορίζεται η δομή των ομάδων, όλα τα σχήματα των ομάδων θα πρέπει να είναι ίδια ή πολύ όμοια με το πρωτότυπο τρίγωνο. Στη συνέχεια το σύστημα χρησιμοποιεί ένα γενετικό αλγόριθμο (Genetic algorithm) (Holland, 1975) για να παράγει συγγενείς λύσεις για βέλτιστη ομαδοποίηση.

Η έρευνα που πραγματοποιήθηκε αφορούσε στην εφαρμογή του συστήματος DIANA σε πραγματικές συνθήκες και ήταν διάρκειας 13 εβδομάδων. Τμήμα 66 προπτυχιακών φοιτητών πληροφορικής κατανεμήθηκε σε ομάδες των τριών, είτε τυχαία (10 ομάδες) είτε με τη χρήση του αλγόριθμου (12 ομάδες). Σύμφωνα με τους ερευνητές, τα αποτελέσματα έδειξαν ότι (α) οι καθορισμένες από τον αλγόριθμο ομάδες ήταν πιο ικανές να συμπληρώσουν ό,τι τους ζητήθηκε να κάνουν, σε ένα στατιστικά σημαντικό επίπεδο, (β) και οι δύο ομάδες ήταν εξίσου ικανές να επιλύσουν σχεδόν το 80% από αυτά που επέλεξαν να κάνουν, και (γ) οι καθορισμένες από τον αλγόριθμο ομάδες είχαν μικρότερη διαφοροποίηση στην επίδοση μέσα στην ομάδα. Τα επίπεδα ικανοποίησης με τη στάση των μελών της ομάδας καθώς και η συνεργατική διαδικασία και τα αποτελέσματα της ομάδας ήταν υψηλότερα για τα μέλη των καθορισμένων από τον αλγόριθμο ομάδων.

Σύμφωνα με τους ερευνητές της μελέτης (Wang, Lin & Sun, 2007), το σύστημα μπορεί να είναι εξαιρετικά χρήσιμο σε καθηγητές που αρχίζουν να καταλαβαίνουν τις μοναδικές δεξιότητες των φοιτητών τους ή ενδιαφέρονται να περιλάβουν πιο σύνθετους παράγοντες στη σύνθεση των ομάδων. Επίσης, το σύστημα προτείνεται για χρήση στην εκπαίδευση από απόσταση για τη σύνθεση εικονικών ομάδων (virtual groups).

2.4 Συμπεράσματα

Από την επισκόπηση των ερευνών προκύπτει ότι, ο καθοδηγούμενος σχηματισμός ομάδων είναι σημαντικός προκειμένου να αποφευχθούν προβλήματα που προκύπτουν από την τυχαία κατανομή των φοιτητών ή από τις αυτοδημιούργητες ομάδες (Dillenbourg, 1999; Alfonseca et al., 2006; Layton et al., 2010). Η συγκρότηση ομάδων από συστήματα μπορεί να

βελτιώσει σε πολλές περιπτώσεις τη μαθησιακή διαδικασία (πχ. Cavanaugh et el, 2004; Graf & Bekele, 2006).

Σύμφωνα με πολλούς ερευνητές, η ετερογένεια των μαθησιακών τύπων αυξάνει τη δημιουργικότητα των ομάδων (Herrmann, 1987; Johnson & Johnson, 1994; Kolb, 1999; Robinson, 2001; De Abreu Dos Reis et al., 2007) και σε πολλές έρευνες που μελετήθηκαν (πχ. Alfonseca et al., 2006; De Faria et al., 2006; Grigoriadou et al., 2006; Al-Dujaily and Ryu, 2007; Wang et al., 2007), η ετερογένεια και η ποικιλομορφία στη συγκρότηση των ομάδων ήταν βοηθητική για την απόδοση των ομάδων. Οι μαθησιακοί τύποι φαίνεται ότι επηρεάζουν δυναμικά τα αποτελέσματα των ομάδων (Sternberg, 1994; Wang et al, 2007) και σε πολλές έρευνες αποτελούν ένα αξιόπιστο εργαλείο για το σχηματισμό ομάδων (πχ. Martin & Paredes, 2004; Alfonseca et al., 2006; Grigoriadou et al., 2006; Muehlenbrock, 2006; Al-Dujaily & Ryu, 2007; Wang et al., 2007; Christodoulopoulos, & Papanikolaou, 2007b).

Κεφάλαιο 3

3. Γενετικοί Αλγόριθμοι

3.1 Εισαγωγή

Οι γενετικοί αλγόριθμοι (genetic algorithms: GA) είναι ευριστικοί αλγόριθμοι αναζήτησης (heuristic search algorithms) που προσομοιώνουν τις εξελικτικές διαδικασίες που παρατηρούνται στη φύση και οι οποίες βασίζονται στη φυσική επιλογή και στη φυσική εξέλιξη όπως αυτές περιγράφονται στη Δαρβινική εξελικτική θεωρία και στη γενετική.

Σύμφωνα με τη θεωρία της φυσικής επιλογής του Δαρβίνου [1] οι οργανισμοί που δεν είναι κατάλληλοι για το εκάστοτε περιβάλλον δεν επιβιώνουν, ενώ αυτοί που είναι, ζουν και αναπαράγουν. Συνεπώς, η ικανότητα (fitness) ενός οργανισμού μετριέται από την επιτυχία επιβίωσής του. Οι απόγονοι (offsprings) μοιάζουν στους γονείς τους (parents) και συνεπώς κάθε νέα γενιά αποτελείται από οργανισμούς που μοιάζουν στα καταλληλότερα μέλη της προηγούμενης γενιάς, με αποτέλεσμα κάθε καινούρια γενιά να έχει περισσότερες πιθανότητες επιβίωσης από την προηγούμενη. Εάν το περιβάλλον αλλάζει αργά, τα διάφορα είδη προλαβαίνουν να εξελίσσονται παράλληλα με αυτό. Εάν όμως, παρουσιαστεί μια ξαφνική αλλαγή στο περιβάλλον, πιθανώς τα διάφορα είδη να αφανιστούν. Περιστασιακά μπορεί να συμβούν διάφορες μεταλλάξεις (mutations), οι περισσότερες των οποίων οδηγούν στον γρήγορο θάνατο του μεταλλαγμένου οργανισμού, ενώ κάποιες από αυτές οδηγούν σε ένα νέο καλύτερο είδος [2].

Στις αρχές του 1950 εμπνευσμένοι από τη Δαρβινική θεωρία άρχισαν να προτείνονται διάφοροι εξελικτικοί αλγόριθμοι (evolutionary algorithms). Αρκετοί ερευνητές, ανεξάρτητα μεταξύ τους, άρχισαν να μελετούν εξελικτικά συστήματα (evolutionary systems) βασισμένοι στην ιδέα ότι οι εξελικτικοί αλγόριθμοι θα μπορούσαν να χρησιμοποιηθούν ως εργαλεία σε προβλήματα βελτιστοποίησης στη μηχανική. Οι εξελικτικοί αλγόριθμοι αποσκοπούν στην εξέλιξη ενός πληθυσμού πιθανών λύσεων σε ένα συγκεκριμένο πρόβλημα, με τη χρήση τελεστών (operators) εμπνευσμένων από τη φυσική γενετική ποικιλότητα και τη φυσική επιλογή [3]. Στις δεκαετίες του 1950 και 1960 αναπτύχθηκαν αρκετά εξελικτικά συστήματα από πολλούς ερευνητές. Βασικές υποκατηγορίες τους είναι ο εξελικτικός προγραμματισμός (evolutionary programming), οι στρατηγικές εξέλιξης (evolution strategies), τα συστήματα ταξινόμησης (classifier systems) και ο γενετικός προγραμματισμός (genetic programming) .

Οι γενετικοί αλγόριθμοι είναι μια ειδική κατηγορία εξελικτικών αλγορίθμων που αρχικά είχαν επινοηθεί από τον J. Holland και τους συνεργάτες του στο University of Michigan στις αρχές της δεκαετίας του 1960 και αφού αναπτύχθηκαν είχαν προταθεί με την

τελική τους μορφή το 1975 [4]. Σε αντίθεση με τους προηγούμενους ερευνητές εξελικτικών αλγορίθμων, ο αρχικός στόχος του Holland δεν ήταν ο σχεδιασμός αλγορίθμων για την επίλυση συγκεκριμένων προβλημάτων, αλλά η φορμαλιστική μελέτη του φαινομένου της προσαρμογής, όπως αυτή γίνεται στη φύση, και η ανάπτυξη μεθόδων με τις οποίες οι φυσικοί αυτοί μηχανισμοί, θα μπορούσαν να εισαχθούν σε υπολογιστικά συστήματα [3]. Αποσκοπούσε στη δημιουργία υπολογιστικών προγραμμάτων που θα «εξελίσσονταν» με τρόπο που προσομοιώνει τη φυσική επιλογή και θα ήταν ικανά να λύσουν πολύπλοκα προβλήματα τα οποία ούτε οι δημιουργοί τους δεν είναι ικανοί να κατανοήσουν πλήρως (“Computer programs that “evolve” in ways that resemble natural selection can solve complex problems even their creators do not fully understand.”) [4] . Σήμερα, ωστόσο η πιο ευρεία τους χρήση, είναι η αναζήτηση βέλτιστων λύσεων σε συστήματα που ανάγονται σε μαθηματικά προβλήματα. Ο Holland χρησιμοποίησε τεχνικές που προσομοιώνουν τη φυσική επιλογή και τελεστές από τη γενετική όπως η διασταύρωση (crossover), η μετάλλαξη (mutation) και η αντιστροφή (inversion) έτσι ώστε ένας πληθυσμός χρωματοσωμάτων (chromosomes) να μπορεί να εξελιχθεί σε έναν καινούριο πληθυσμό. Ο γενετικός αλγόριθμος του Holland με την εισαγωγή της έννοιας του πληθυσμού και των τελεστών διασταύρωσης, μετάλλαξης και αντιστροφής αποτελούσε μεγάλη καινοτομία στους εξελικτικούς αλγόριθμους. Η ορολογία που χρησιμοποιείται στους γενετικούς αλγόριθμους είναι δανεισμένη από τη βιολογία και πιο συγκεκριμένα από τη γενετική.

Κάθε λύση σε ένα γενετικό αλγόριθμο αναπαρίσταται με ένα χρωμόσωμα (chromosome) και κάθε χρωμόσωμα αποτελείται από έναν αριθμό γονιδίων (genes). Γονίδιο (gene) είναι κάθε σύμβολο που χρησιμοποιείται για την αναπαράσταση μιας υποψήφιας λύσης. Μια ομάδα συγκεκριμένου αριθμού γονιδίων σχηματίζει ένα χρωμόσωμα (chromosome) και αντιπροσωπεύει μια πιθανή λύση. Μια ομάδα χρωματοσωμάτων συγκροτεί έναν πληθυσμό (population): μια «δεξαμενή» από πιθανές λύσεις για ένα πρόβλημα. Οι γενετικοί αλγόριθμοι χρησιμοποιούν βασικούς μηχανισμούς της εξέλιξης: κληρονομικότητα (inheritance), επιλογή (selection), διασταύρωση (crossover), μετάλλαξη (mutation), εφαρμόζοντάς τους ως τελεστές πάνω στα χρωμοσώματα. Κάθε μηχανισμός πραγματοποιείται με μια συγκεκριμένη πιθανότητα. Υπάρχει μία συνάρτηση, που ονομάζεται συνάρτηση καταλληλότητας και η οποία θα εξηγηθεί λεπτομερειακά πιο κάτω, που χαρακτηρίζει τα μέλη του πληθυσμού. Αυτά τα μέλη του πληθυσμού με υψηλή τιμή συνάρτησης καταλληλόλητας θεωρούνται επιλέξιμα για αναπαραγωγή και ονομάζονται γονείς (parents). Με τη χρήση τελεστών στους γονείς κατά την αναπαραγωγή, προκύπτουν νέα χρωμοσώματα, τα οποία καλούνται απόγονοι (offsprings). Κάθε νέα γενιά πληθυσμού αντιστοιχεί με ακόμα μια επανάληψη του γενετικού αλγόριθμου.

Η όλη διαδικασία της εξέλιξης προς τη βέλτιστη λύση υλοποιείται σε 3 βασικά στάδια: Στο πρώτο στάδιο (αρχικοποίηση) δημιουργείται τυχαία ένας αρχικός πληθυσμός απαρτιζόμενος από κάποια χρωμοσώματα (chromosomes). Κάθε χρωμόσωμα αντιπροσωπεύει μια λύση στο πρόβλημα, όπως αναφέρθηκε πιο πάνω, και χρησιμοποιεί μια μορφή κωδικοποίησης. Ένα χρωμόσωμα μπορεί να κωδικοποιηθεί ως ένα διάνυσμα στοιχείων. Κάθε στοιχείο του χρωμοσώματος αντιστοιχεί σε ένα γονίδιο (gene). Μια ομάδα κωδικοποιημένων γονιδίων συνθέτει ένα κωδικοποιημένο χρωμόσωμα.

Ο αρχικός πληθυσμός (initial population) αποτελεί την πρώτη γενιά (first generation). Κάθε επανάληψη του αλγόριθμου δημιουργεί μια καινούρια γενιά. Ο γενετικός αλγόριθμος αξιολογεί κάθε μεμονωμένη λύση (χρωμόσωμα) χρησιμοποιώντας μια συνάρτηση καταλληλόλητας (fitness function).

Στο δεύτερο στάδιο, το στάδιο της αναπαραγωγής (reproduction) εφαρμόζεται ο τελεστής της επιλογής (selection operator). Διάφορα άτομα από τον τρέχων πληθυσμό επιλέγονται στοχαστικά βάσει της τιμής της συνάρτησης καταλληλότητάς τους. Ακολούθως, εφαρμόζεται ο τελεστής της διασταύρωσης (crossover) στα επιλεγμένα χρωμοσώματα (τα επιλεγμένα χρωμοσώματα ανασυνδυάζονται μεταξύ τους παράγοντας καινούρια χρωμοσώματα), καθώς και ο τελεστής της μετάλλαξης (mutation) (διάφορα γονίδια κάποιων χρωματοσωμάτων μεταλλάσσονται τυχαία). Με την εφαρμογή των τελεστών αυτών σχηματίζεται ένας νέος πληθυσμός που είναι η επόμενη γενιά χρωματοσωμάτων.

Στο τρίτο στάδιο ο νέος πληθυσμός χρησιμοποιείται στην επόμενη επανάληψη του αλγορίθμου και η διαδικασία επαναλαμβάνεται για ένα αριθμό επαναλήψεων. Ο αλγόριθμος τερματίζεται σύμφωνα με ένα κριτήριο τερματισμού (για παράδειγμα όταν ο μέγιστος αριθμός γενεών έχει επιτευχθεί). Εν τέλει ο γενετικός αλγόριθμος επιστρέφει τον καλύτερο πληθυσμό χρωματοσωμάτων, τα οποία έχουν τις καλύτερες τιμές καταλληλόλητας. Αυτός ο πληθυσμός θα είναι η καλύτερη «δεξαμενή» λύσεων του προβλήματος.

‘Ψευδοκώδικας Γενετικού αλγόριθμου’

ΑΡΧΙΚΟΠΟΙΗΣΕ ΤΟΝ ΠΛΥΘΗΣΜΟ (ΠΙΘΑΝΕΣ ΛΥΣΕΙΣ) ΜΕ ΤΥΧΑΙΑ ΧΡΩΜΟΣΩΜΑΤΑ

ΑΞΙΟΛΟΓΗΣΕ ΚΑΘΕ ΧΡΩΜΟΣΩΜΑ (ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ)

ΕΠΑΝΕΛΑΒΕ

ΕΦΑΡΜΟΣΕ ΕΛΙΤΙΣΜΟ

ΕΠΕΛΕΞΕ ΓΟΝΕΙΣ

ΔΙΑΣΤΑΥΡΩΣΕ ΤΟΥΣ ΓΟΝΕΙΣ ΚΑΙ ΔΗΜΙΟΥΡΓΗΣΕ ΑΠΟΓΟΝΟΥΣ

ΜΕΤΑΛΛΑΞΕ ΧΡΩΜΟΣΩΜΑΤΑ ΤΟΥ ΠΛΥΘΗΣΜΟΥ ΚΑΙ ΑΠΟΓΟΝΩΝ

ΕΠΕΛΕΞΕ ΤΑ ΚΑΤΑΛΛΗΛΟΤΕΡΑ ΧΡΩΜΟΣΩΜΑΤΑ ΓΙΑ ΤΗΝ ΕΠΟΜΕΝΗ ΓΕΝΙΑ

ΜΕΧΡΙΣ_ΟΤΟΥ ΙΚΑΝΟΠΟΙΗΘΟΥΝ ΤΑ ΚΡΙΤΗΡΙΑ ΤΕΡΜΑΤΙΣΜΟΥ ΠΟΥ ΕΧΟΥΝ ΟΡΙΣΤΕΙ

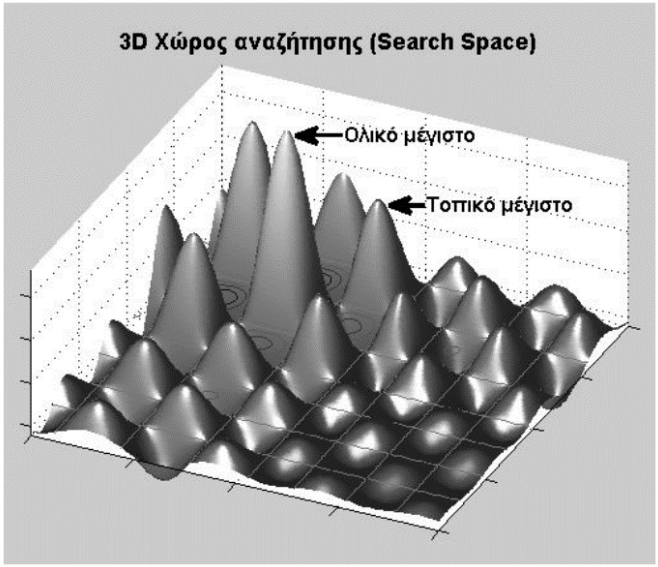
Η πιο πάνω διαδικασία θα περιγράφει αναλυτικότερα στα επόμενα υποκεφάλαια.

3.2 Χώρος αναζήτησης (search space)

Όταν λύνουμε ένα πρόβλημα συνήθως ψάχνουμε για κάποια λύση, η οποία θα είναι η βέλτιστη των διαφόρων πιθανών άλλων λύσεων. Ο χώρος όλων των εφικτών πιθανών λύσεων αποτελεί τον χώρο αναζήτησης λύσεων του προβλήματος (search space).

Στόχος είναι η εύρεση της βέλτιστης λύσης (ή των βέλτιστων λύσεων), η οποία αντιστοιχεί σε ένα σημείο (ή περισσότερα) στο χώρο αναζήτησης. Έτσι, η αναζήτηση λύσης ανάγεται στην αναζήτηση κάποιου ολικού μέγιστου ή ελάχιστου - ανάλογα με το πρόβλημα - στο χώρο αναζήτησης. Ο χώρος αναζήτησης είναι δυνατό να είναι όλος γνωστός την ώρα επίλυσης του προβλήματος, αλλά συνήθως, εκ των προτέρων, γνωρίζουμε μόνο λίγα σημεία του και παράγουμε τα υπόλοιπα καθώς η διαδικασία αναζήτησης λύσεων εξελίσσεται.

Το πρόβλημα είναι ότι η αναζήτηση μπορεί να είναι περίπλοκη. Οι γενετικοί αλγόριθμοι χρησιμοποιούνται ως μια μέθοδος αναζήτησης μιας κατάλληλης λύσης (κατά προσέγγιση βέλτιστης).



Εικόνα 3.1: Παράδειγμα τρισδιάστατου χώρου αναζήτησης (search space)

3.3 Αναπαράσταση πιθανών λύσεων – Κωδικοποίηση χρωμοσωμάτων (Encoding)

Πριν το στάδιο της αρχικοποίησης απαιτείται τα χαρακτηριστικά κάθε πιθανής λύσης να κωδικοποιηθούν στα χρωμοσώματα. Κάθε παράμετρος του προβλήματος αντιστοιχεί σε ένα γονίδιο του χρωμοσώματος και πρέπει να κωδικοποιηθεί. Συνεπώς, πριν από οποιαδήποτε διαδικασία, πρέπει πρωταρχικά, να επιλεγεί μια μέθοδος αναπαράστασης των πιθανών λύσεων και κωδικοποίησής τους στο χρωμόσωμα σε μορφή επεξεργάσιμη από τον υπολογιστή, δηλαδή με μαθηματικό τρόπο. Το είδος της κωδικοποίησης των χρωμοσωμάτων προσδιορίζει το πεδίο τιμών το οποίο ερευνά ο αλγόριθμος.

Με την έννοια χρωμόσωμα, εννοούμε μια συμβολοσειρά σταθερού, συνήθως, μήκους όπου αποθηκεύεται όλη η γενετική πληροφορία ενός χρωμοσώματος. Τα γονίδια είναι τα δομικά στοιχεία του χρωμοσώματος, όπως έχει προαναφερθεί.

Ανάλογα με το πρόβλημα, τους περιορισμούς και τις απαιτήσεις του επιλέγεται η αντίστοιχη μορφή κωδικοποίησης. Είναι πιθανό ένα πρόβλημα να επιδέχεται περισσότερες από μια κωδικοποιήσεις. Κάποιες βασικές μορφές κωδικοποίησης που έχουν ήδη χρησιμοποιηθεί είναι οι ακόλουθες: Δυαδική κωδικοποίηση (Binary encoding), Οκταδική κωδικοποίηση (Octal encoding), δεκαεξαδική κωδικοποίηση (Hexadecimal encoding), κωδικοποίηση με δομές ακέραιων τιμών στο δεκαδικό σύστημα (decimal integer encoding), κωδικοποίηση με δομές δεκαδικών τιμών στο δεκαδικό σύστημα (decimal float encoding), κωδικοποίηση μεταλλαγής (Permutation encoding), κωδικοποίηση τιμής (Value encoding), υβριδική κωδικοποίηση, κωδικοποίηση σε μορφή αλφαριθμητικής ακολουθίας και κωδικοποίηση δέντρου.

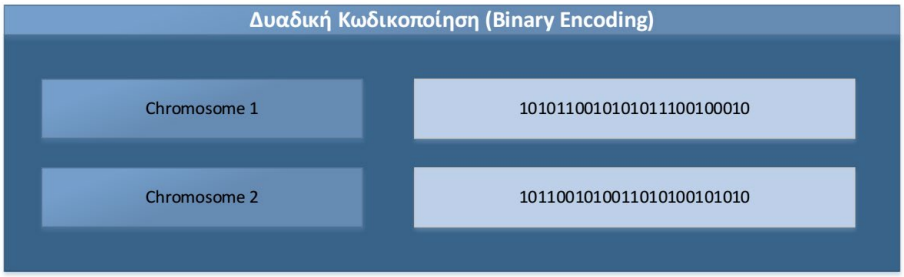
Είναι πιθανό ένα πρόβλημα να επιδέχεται περισσότερες από μία κωδικοποιήσεις και η επιλογή της καταλληλότερης αποτελεί κρίσιμο βήμα σε ένα γενετικό αλγόριθμο. Κάθε χρωμόσωμα αναπαρίσταται ως ένα διάνυσμα και υλοποιείται ως ένας πίνακας - γραμμή πολλών μεταβλητών.

Το χρωμόσωμα κατασκευάζεται με την τοποθέτηση όλων των κωδικοποιημένων χαρακτηριστικών το ένα πλησίον του άλλου. Έτσι ώστε κατά την αποκωδικοποίηση (decoding) του χρωμοσώματος γίνεται εύκολα η εύρεση των τιμών των χαρακτηριστικών.

3.3.1 Δυαδική κωδικοποίηση (Binary Encoding)

Η πιο τυπική προσέγγιση και πιο ευρέως διαδεδομένη είναι η δυαδική κωδικοποίηση (binaryencoding), δηλαδή η κωδικοποίηση με χρήση των δυαδικών ψηφίων 0 και 1. Η δυαδική αναπαράσταση {0,1} προτάθηκε από το Holland το 1975. Οι πρώτες μελέτες σχετικά με γενετικούς αλγόριθμους χρησιμοποίησαν αυτόν τον τύπο κωδικοποίησης εξαιτίας της

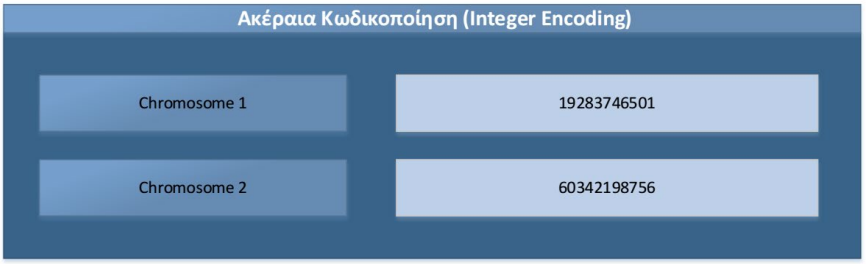
απλότητάς της καθώς και του γεγονότος ότι είναι πιο κοντά στη γλώσσα μηχανής. Κάθε πιθανή λύση αναπαρίσταται στο χρωμόσωμα ως μια ακολουθία 0 και 1 (binary string {0,1}) και κάθε bit δείχνει ένα χαρακτηριστικό της λύσης. Ας σημειωθεί, ότι όσο πιο μεγάλη είναι η ακολουθία των 0 και 1, (δηλαδή όσες περισσότερες είναι οι μεταβλητές απόφασης και αντίστοιχα τα γονίδια του χρωμοσώματος) τόσο περισσότερο μεγαλώνει ο χώρος αναζήτησης με εκθετικούς ρυθμούς [6]



Εικόνα 3.2: Παράδειγμα δυαδικής κωδικοποίησης σε χρωμοσώματα 25 γονιδίων

3.3.2 Κωδικοποίηση με δομές ακεραίων (integer encoding)

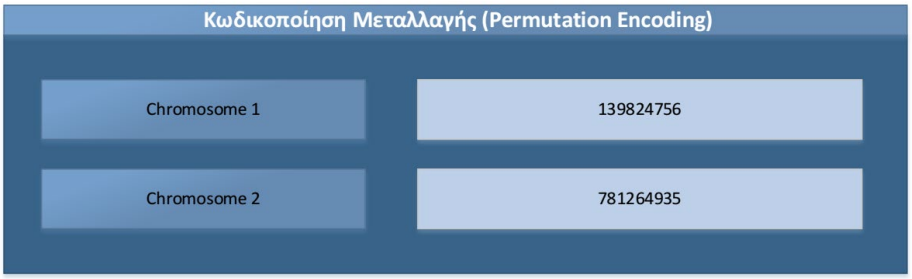
Η κωδικοποίηση των λύσεων σαν δομές ακεραίων τιμών (integer encoding) χρησιμοποιεί μόνο ακέραιες τιμές από το δεκαδικό αριθμητικό σύστημα {0,1,2,3,4,5,6,7,8,9}.



Εικόνα 3.3: Παράδειγμα ακέραιας κωδικοποίησης

3.3.3 Κωδικοποίηση μεταλλαγής (Permutation encoding)

Η κωδικοποίηση μεταλλαγής χρησιμοποιείται σε προβλήματα διάταξης, όπως για παράδειγμα το πρόβλημα του πλανόδιου πωλητή ή προβλήματα καθορισμού σειράς στόχων. Στην κωδικοποίηση αυτή, κάθε χρωμόσωμα είναι μια ακολουθία ακέραιων αριθμών, η οποία αντιπροσωπεύει μια σειρά. Κάθε αριθμός αντιπροσωπεύει μια θέση σε μια σειρά (γι' αυτό πρέπει να εμφανίζεται μόνο μια φορά).



Εικόνα 3.4: Παράδειγμα κωδικοποίησης μεταλλαγής

3.4 Αρχικοποίηση (Initialization)

Στο στάδιο της αρχικοποίησης, το πρώτο στάδιο ενός γενετικού αλγορίθμου, παράγεται ένας αρχικός πληθυσμός από μεμονωμένα χρωματοσώματα-λύσεις, που δημιουργούνται τυχαία. Προτού γίνει αυτό, οι μεταβλητές απόφασης του προβλήματος έχουν ήδη κωδικοποιηθεί σε χρωμοσώματα. Αυτός ο πληθυσμός περιέχει πιθανές λύσεις του προβλήματος, λύσεις δηλαδή, που βρίσκονται στο χώρο αναζήτησης λύσεων του προβλήματος (search space). Το μέγεθος του πληθυσμού αυτού, εξαρτάται από τη φύση του εκάστοτε προβλήματος, αλλά τυπικά περιέχει έναν μεγάλο αριθμό πιθανών λύσεων (συνήθως μεταξύ 50 με 500 λύσεις). Περιστασιακά, οι λύσεις μπορούν να αναζητηθούν στις περιοχές, στις οποίες οι βέλτιστες λύσεις είναι πιθανό να βρεθούν.

Με τη δημιουργία της πρώτης γενιάς ξεκινά η διαδικασία. Σε κάθε χρωμόσωμα εφαρμόζεται η χρήση της συνάρτησης καταλληλόλητας (fitness function) που υποδεικνύει πόσο κατάλληλο είναι το υπό αξιολόγηση χρωμόσωμα, ως λύση για το εξεταζόμενο πρόβλημα.

3.4.1 Αξιολόγηση χρωμοσωμάτων - Συνάρτηση καταλληλόλητας (Fitness function)

Στους γενετικούς αλγορίθμους η επιλογή των χρωμοσωμάτων -πιθανών λύσεων- προς αναπαραγωγή γίνεται με καθορισμένο τρόπο και γι’ αυτό είναι αναγκαία η χρήση μιας μεθόδου η οποία θα είναι υπεύθυνη για την συγκριτική αξιολόγηση των χρωμοσωμάτων. Ο γενετικός αλγόριθμος αφού αποκωδικοποιήσει κάθε χρωμόσωμα του τρέχοντος πληθυσμού κάνει χρήση της λεγόμενης συνάρτησης καταλληλόλητας (fitness function) που θα αναθέσει μια τιμή καταλληλόλητας σε κάθε χρωμόσωμα του τρέχοντος πληθυσμού[3].

Επομένως, παίρνει ως είσοδο την αποκωδικοποιημένη συμβολοσειρά (τα γονίδια κάθε χρωμοσώματος) και επιστρέφει την τιμή καταλληλόλητας της. Η τιμή αυτή αποτελεί και τον καθοριστικό παράγοντα επιβίωσης και πολλαπλασιασμού ή όχι του χρωμοσώματος. Η

καταλληλόλητα ενός χρωμοσώματος εξαρτάται από το πόσο καλά ένα χρωμόσωμα επιλύει το υπό εξέταση πρόβλημα. Η συνάρτηση καταλληλόλητας δίνει ένα ποσοτικό μέτρο αυτής της ικανότητας, της εγγύτητας δηλαδή στη βέλτιστη λύση (απόσταση ή ομοιότητα από την ιδανική λύση), αντιστοιχίζοντας τα σημεία του χώρου των υποψήφιων λύσεων. Είναι σημαντικό η συνάρτηση καταλληλόλητας να είναι εύκολα υπολογίσιμη έτσι ώστε να μην επιβραδύνεται η εκτέλεση του γενετικού αλγορίθμου.

Μια συνήθης εφαρμογή των γενετικών αλγορίθμων είναι η βελτιστοποίηση συνάρτησης, όπου στόχος είναι η εύρεση ενός συνόλου τιμών παραμέτρων που ελαχιστοποιούν ή μεγιστοποιούν μια πολύπλοκη πολυπαραμετρική συνάρτηση. Βελτιστοποίηση είναι η διαδικασία ρύθμισης των εισόδων ενός προβλήματος έτσι ώστε να βρεθεί η ελάχιστη ή μέγιστη έξοδος. Η είσοδος αποτελείται από μεταβλητές. Η διαδικασία είναι η συνάρτηση καταλληλόλητας (fitness function) και η έξοδος είναι η καταλληλόλητα. Εάν η καταλληλόλητα πρέπει να ελαχιστοποιηθεί, η βελτιστοποίηση γίνεται πρόβλημα ελαχιστοποίησης, ενώ αν πρέπει να μεγιστοποιηθεί γίνεται πρόβλημα μεγιστοποίησης.

Σε μαθηματικά προβλήματα βελτιστοποίησης συνάρτησης –έστω f(x)- είναι προφανές ότι η συνάρτηση καταλληλόλητας θα πρέπει να είναι η ίδια η συνάρτηση f. Συνεπώς σε κάθε υποψήφια λύση, δηλαδή σε κάθε πιθανή τιμή της μεταβλητής x, θα αντιστοιχεί μια τιμή καταλληλόλητας που θα αξιολογεί την εκάστοτε πιθανή λύση και που στην περίπτωση αυτή θα ταυτίζεται με την ίδια την τιμή της από τη συνάρτηση f.

Όταν το πρόβλημα έχει μόνο ένα κριτήριο, διέπεται από μια μόνο αντικειμενική συνάρτηση και η επιλογή συνάρτησης καταλληλόλητας είναι εύκολη αφού θα είναι ίδια με την αντικειμενική συνάρτηση ή ένα μετασχηματισμό της. Στην περίπτωση όμως προβλημάτων βελτιστοποίησης με πολλαπλά κριτήρια απαιτείται η ταυτόχρονη βελτιστοποίηση δύο ή περισσότερων αντικρουόμενων ζητημάτων με διάφορους περιορισμούς. Στο πρόβλημα βελτιστοποίησης με πολλαπλά κριτήρια κάθε κριτήριο έχει τη δική του συνάρτηση καταλληλόλητας. Έτσι το ολικό πρόβλημα ανάγεται σε ένα συνδυασμό όλων των συναρτήσεων σε μια μόνο συναρτησιακή μορφή που θα αποτελεί και τη συνάρτηση καταλληλόλητας. Εάν το πρόβλημα βελτιστοποίησης με πολλαπλά κριτήρια είναι καλά ορισμένο δεν θα υπάρχει μια μοναδική λύση η οποία θα βελτιστοποιεί κάθε υποστόχο. Σε αυτές τις περιπτώσεις υπάρχει δυσκολία στον καθορισμό μιας ενδεχόμενης λύσης ως ιδανικότερης από κάποιες άλλες, αφού μπορεί να είναι καλύτερη ως προς ένα κριτήριο και χειρότερη ως προς κάποιο άλλο. Πρέπει σε κάθε περίπτωση ένα κριτήριο να έχει φτάσει σε ένα τέτοιο σημείο, ώστε κάθε επιπλέον προσπάθεια βελτιστοποίησής του να συνεπάγεται την υποβάθμιση άλλων κριτηρίων. Ο στόχος είναι η εύρεση μιας τέτοιας λύσης –ανάμεσα σε πολλές που μπορεί να υπάρχουν- η οποία να είναι καλύτερη με αυτή την έννοια σε σχέση με τις υπόλοιπες. Τέτοιου είδους προβλήματα είναι γνωστά ως προβλήματα πολλαπλών στόχων

(multi-objective problems). Οι γενετικοί αλγόριθμοι που επιλύουν προβλήματα αυτής της κατηγορίας καλούνται γενετικοί αλγόριθμοι πολλαπλών στόχων (multi-objective GAs - MOGAs) και συνήθως εφαρμόζουν μαθηματικές θεωρίες βελτιστοποίησης πολλών κριτηρίων (π.χ. βελτιστοποίηση Pareto) για την αξιολόγηση των ατόμων.

Η μέτρηση καταλληλόλητας είναι η κινητήρια δύναμη του γενετικού αλγόριθμου. Ανάλογα με το πρόβλημα πρέπει να γίνεται και η σχεδίαση της πιο ικανοποιητικής συνάρτησης καταλληλόλητας. Ένα φαινομενικά δυσεπίλυτο πρόβλημα καθίσταται προσπελάσιμο από ένα γενετικό αλγόριθμο με τη δημιουργία της κατάλληλης συνάρτησης καταλληλόλητας. Ο σχεδιασμός συναρτήσεων καταλληλόλητας που να πληρούν τις ανάλογες προϋποθέσεις, ακόμα και για κατηγορίες όμοιων προβλημάτων, δεν είναι πάντα εύκολος και για το λόγο αυτό συχνά απαιτείται εξειδικευμένη γνώση για το πρόβλημα.

Η αξιολόγηση καθώς και τα υπόλοιπα στάδια του γενετικού αλγόριθμου αποκαλύπτουν τη μεγάλη χρησιμότητα των γενετικών αλγορίθμων στην επίλυση δυσνόητων προβλημάτων. Ο προγραμματιστής αρκεί να σχεδιάσει την αναπαράσταση των ενδεχόμενων λύσεων και τη συνάρτηση αξιολόγησής τους χωρίς να χρειάζεται να έχει περαιτέρω γνώση των εσωτερικών διεργασιών του προβλήματος. Τα υπόλοιπα τα αναλαμβάνει ο γενετικός αλγόριθμος.

3.5 Ελιτισμός – Διατήρηση των ικανών (Elitism)

Ο ελιτισμός έχει προταθεί από τον Kenneth De Jong το 1975 και εξασφαλίζει ότι ένα ποσοστό από τα καλύτερα χρωμοσώματα της κάθε γενιάς θα μεταπηδήσουν στην επόμενη γενιά χωρίς να περάσουν από το στάδιο της αναπαραγωγής. Αυτή η τεχνική είναι σημαντική επειδή εγγυάται ότι οι ενδεχόμενα καλές λύσεις μιας γενιάς δεν θα χαθούν ούτε θα αλλοιωθούν εξαιτίας προσμίξεων με άλλες λύσεις.

Το καλύτερο, μέχρι μια δεδομένη στιγμή του αλγορίθμου, χρωμόσωμα, (ή μερικά από τα καλύτερα μέχρι στιγμής χρωμοσώματα) μπορεί να μην έχει επιλεγεί να επιβιώσει στην επόμενη γενεά, οπότε κατά τη δημιουργία της νέας γενεάς υπάρχει μεγάλη πιθανότητα να χαθεί. Ωστόσο, το χρωμόσωμα αυτό μπορεί να είναι μια υψηλής ποιότητας λύση, ίσως ακόμα και η ολικά βέλτιστη λύση, και ως αποτέλεσμα να μην συμπεριληφθεί στον πληθυσμό. Για να αποφευχθεί η απώλεια αυτή, εφαρμόζεται ο ελιτισμός. Σε κάθε γενιά τα χρωμοσώματα με την υψηλότερη τιμή καταλληλόλητας αντιγράφονται κατευθείαν στην επόμενη γενιά αυτούσια, χωρίς καμιά τροποποίηση των γενετικών χαρακτηριστικών τους. Με αυτό τον τρόπο διασφαλίζεται ότι η τελική λύση θα είναι η ολικά βέλτιστη λύση, αυξάνοντας έτσι, την απόδοση του αλγορίθμου.

Συνήθως, η τιμή της παραμέτρου του ελιτισμού ανήκει στο διάστημα [0.05-0.20]. Για παράδειγμα, αν το μέγεθος του πληθυσμού μιας γενεάς χρωμοσωμάτων είναι N=200, τα 10-

40 καλύτερα χρωμοσώματα από αυτά θα μεταφερθούν κατευθείαν στον πληθυσμό της επόμενης γενιάς.

Ο ελιτισμός παρουσιάζεται ιδιαίτερα χρήσιμος στις περιπτώσεις που η πιθανότητα των γενετικών τελεστών έχουν υψηλή τιμή. Σε αυτές τις περιπτώσεις, χρωμοσώματα που ανήκουν στο σύνολο των «υποψήφιων γονέων», έχουν μεγάλη πιθανότητα να μην προχωρήσουν στην επόμενη γενιά και γι' αυτό κρίνεται αναγκαίο να γίνει χρήση μιας σχετικά μεγάλης πιθανότητας ελιτισμού για τη διατήρηση των καλών λύσεων.

3.6 Επιλογή (Selection)

Στο στάδιο της επιλογής, επιλέγονται τα πιο κατάλληλα άτομα από τον τρέχων πληθυσμό για το επόμενο στάδιο, το στάδιο της αναπαραγωγής (reproduction). Τα επιλεγμένα άτομα αναφέρονται ως «γονείς» και θα χρησιμοποιηθούν στην αναπαραγωγή για τον σχηματισμό απογόνων (offsprings). Η επιλογή είναι κομβικής σημασίας στην επιτυχή περάτωση του γενετικού αλγορίθμου, αποσκοπώντας παράλληλα και στην επιλογή καλύτερων χρωμοσωμάτων, αλλά και στην επιλογή χρωμοσωμάτων που βρίσκονται πλησιέστερα στη βέλτιστη λύση.

Υπάρχουν πολλές διαθέσιμες μέθοδοι για την επιλογή χρωμοσωμάτων. Ανάλογα με τις παραμέτρους και τις απαιτήσεις του προβλήματος μπορεί να χρησιμοποιηθεί η καταλληλότερη εκ των διαφόρων αυτών τεχνικών ή κάποια παραλλαγή τους. Παρακάτω θα αναφερθούν δυό συνήθεις μέθοδοι επιλογής.

3.6.1 Επιλογή ρουλέτας (Roulette wheel selection)

Η επιλογή ρουλέτα, αποτελεί την πιο ευρέως χρησιμοποιούμενη στοχαστική μέθοδο επιλογής και είναι μια απλή μέθοδος υλοποίησης της αναλογικής επιλογής (fitness-proportionate selection) [5]. Τα άτομα επιλέγονται με πιθανότητα επιλογής σύμφωνα με την παρακάτω εξίσωση.

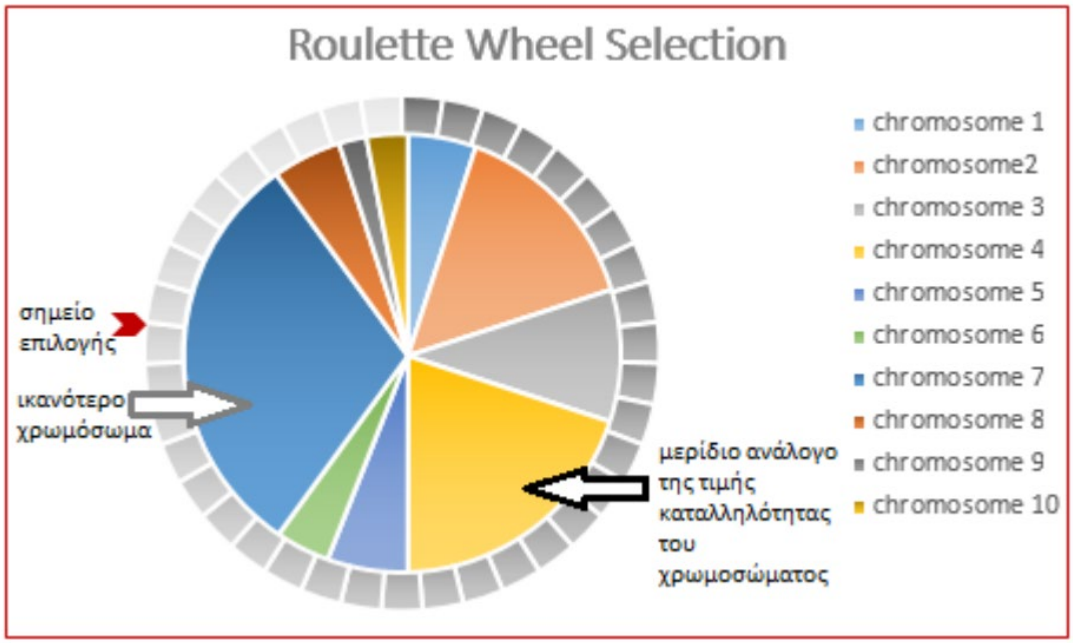
$$P_{selection} = \frac{f(parent_i)}{\sum_i f(parent_i)} \quad (3.1)$$

όπου fi είναι η τιμή καταλληλόλητας (fitness value) του χρωμοσώματος i του πληθυσμού.

Τα χρωμοσώματα επιλέγονται για γονείς ανάλογα με την καταλληλόλητας τους: όσο πιο κατάλληλο είναι ένα χρωμόσωμα τόσες περισσότερες πιθανότητες έχει να επιλεγεί.

Ας υποθέσουμε μια ρουλέτα όπου τοποθετούνται όλα τα χρωμοσώματα του πληθυσμού, κάθε ένα εκ των οποίων παίρνει ένα κομμάτι της ρουλέτας μεγέθους ανάλογου

με την τιμή καταλληλόλητας του. Ακολούθως, ρίχνεται μια μπίλια και επιλέγεται ένα χρωμόσωμα το οποίο εντάσσεται στο σύνολο των υποψήφιων γονέων. Η ρουλέτα γυρίζει τόσες φορές όσες ο αριθμός των χρωμοσωμάτων του πληθυσμού. Το χρωμόσωμα με την μεγαλύτερη ικανότητα θα επιλεχθεί περισσότερες φορές. Αυτό σχηματικά παρουσιάζεται στην εικόνα 3.5.



Εικόνα 3.5: Σχήμα επιλογής με ρουλέτα

Η επιλογή με ρουλέτα μπορεί να υλοποιηθεί με τον ακόλουθο αλγόριθμο:

1. Υπολογισμός του αθροίσματος των τιμών καταλληλόλητας όλων των χρωμοσωμάτων του πληθυσμού (έστω άθροισμα S).

2. Επιλογή τυχαίου αριθμού στο διάστημα [0,S], έστω r.

3. Αρχικοποίηση μεταβλητής d = 0 και σάρωση του πληθυσμού προσθέτοντας κάθε φορά την τιμή καταλληλόλητας του τρέχοντος χρωμοσώματος στη μεταβλητή d. Όταν το άθροισμα d γίνει μεγαλύτερο του r, σταμάτα και επέστρεψε το τρέχον χρωμόσωμα.

4. Επανάληψη της διαδικασίας από το 2ο έως το 4ο βήμα N φορές (όπου N ο πληθάριθμος του συνόλου του πληθυσμού) .

3.6.2 Επιλογή τουρνουά (Tournament selection)

Πραγματοποιούνται διάφορα «τουρνουά» ανάμεσα σε διαφορετικές ομάδες k τυχαία επιλεγμένων χρωμοσωμάτων του πληθυσμού, όπου k το μέγεθος του τουρνουά (tournament size). Τα k επιλεγμένα χρωμοσώματα κάθε ομάδας αναμετρώνται ανά δύο και το ικανότερο

χρωμόσωμα συνεχίζει να συμμετέχει στο τουρνουά της ομάδας ενώ το λιγότερο ικανό απορρίπτεται. Οι αναμετρήσεις μεταξύ των χρωμοσωμάτων της ομάδας συνεχίζονται έως ότου να αναδειχθεί ένα χρωμόσωμα νικητής. Κατά την πραγματοποίηση ενός τουρνουά ανάμεσα στα k τυχαία επιλεγμένα χρωμοσώματα μιας ομάδας επιλέγεται ο νικητής-χρωμόσωμα ως υποψήφιος γονέας, ενώ τα υπόλοιπα χρωμοσώματα της ομάδας, καθώς και τα μη επιλεγμένα στην ομάδα χρωμοσώματα συμμετέχουν εκ νέου στη διαδικασία.

3.7 Αναπαραγωγή (Reproduction)

Στη φύση τα άτομα του πληθυσμού τα οποία έχουν καλύτερη ικανότητα επιβίωσης έχουν την τάση να αναπαράγονται με μεγαλύτερη συχνότητα από ότι τα υπόλοιπα άτομα του είδους. Οι οργανισμοί μέσω της αναπαραγωγής μεταβιβάζουν πολλά από τα χαρακτηριστικά τους στους απογόνους τους. Αυτά τα χαρακτηριστικά δεν μεταβιβάζονται πάντα ως πιστά αντίτυπα με αποτέλεσμα να υφίστανται παραλλαγές τους μέσα στον πληθυσμό. Οι παραλλαγές αυτές οδηγούν σε περαιτέρω διαφοροποίηση, όσον αφορά την ικανότητα των ατόμων του πληθυσμού να ανταγωνίζονται και να επιβιώνουν.

Στους γενετικούς αλγορίθμους μετά το στάδιο της αρχικοποίησης, ακολουθεί το στάδιο της αναπαραγωγής. Στις περισσότερες υλοποιήσεις γενετικών αλγορίθμων στο στάδιο της αναπαραγωγής χρησιμοποιούνται δύο τελεστές: ένας τελεστής διασταύρωσης (crossover), όπου δύο ή περισσότερα άτομα του πληθυσμού ανταλλάζουν γενετική πληροφορία και ένας τελεστής μετάλλαξης (mutation), όπου η γενετική πληροφορία ενός ατόμου μεταβάλλεται χωρίς να πραγματοποιείται ανταλλαγή γενετικής πληροφορίας. Η παραγωγή καινούριου απογόνου από ένα νέο ζευγάρι επιλεγμένων γονέων κάθε φορά, συνεχίζεται έως ότου επιτευχθεί ο επιθυμητός πληθυσμός λύσεων.

3.7.1 Διασταύρωση (Crossover)

Διασταύρωση (crossover) είναι η διαδικασία συνδυασμού 2 ή περισσότερων λύσεων-γονέων (parent solutions) έτσι ώστε να παραχθεί μια λύση-παιδί (child solution) από αυτές. Διαισθητικά, θα λέγαμε ότι η διασταύρωση εξυπηρετεί ανταλλαγή πληροφοριών ανάμεσα σε υποψήφιες λύσεις ή αλλιώς ανταλλαγή γενετικού υλικού ανάμεσα σε χρωμοσώματα αποσκοπώντας σε ενδεχόμενες νέες καλύτερες λύσεις.

Μια συνήθης διασταύρωση πραγματοποιείται σε 3 βασικά βήματα:

1. Ο τελεστής αναπαραγωγής επιλέγει τυχαία ένα ζεύγος ατόμων για αναπαραγωγή από το σύνολο των «υποψήφιων γονέων».

2. Ένα ή περισσότερα σημεία διασταύρωσης Σ.Δ. (crossover points) επιλέγονται τυχαία κατά μήκος των ατόμων.

3. Δημιουργούνται οι δύο απόγονοι. Ξεκινώντας από την πρώτη θέση και κινούμενοι κατά μήκος των ατόμων, ο πρώτος απόγονος αντιγράφει τα γονίδια του πρώτου γονέα, ενώ ο δεύτερος απόγονος τα γονίδια του δευτέρου γονέα. Όταν βρεθεί σημείο διασταύρωσης, η διαδικασία αντιστρέφεται: ο πρώτος απόγονος αντιγράφει τα γονίδια του δεύτερου γονέα και ο δεύτερος απόγονος τα γονίδια του πρώτου γονέα. Η ίδια διαδικασία ακολουθείται και για τα υπόλοιπα σημεία διασταύρωσης έως ότου φτάσει το τέλος των χρωμοσωμάτων. Έτσι προκύπτουν δύο νέα χρωμοσώματα που αποτελούν έναν ανασυνδυασμό των δύο γονέων.

Στόχος της διασταύρωσης είναι η νέα γενιά που θα προκύψει μετά την εφαρμογή της, να αποτελείται από χρωμοσώματα που θα διαφέρουν από τους γονείς τους και θα φέρουν τον συνδυασμό των ικανότερων χαρακτηριστικών τους. Έτσι, μπορούν να προκύψουν επιτυχημένοι συνδυασμοί υψηλής ικανότητας. Ωστόσο, είναι πιθανόν, η διασταύρωση να παραγάγει απογόνους λιγότερο κατάλληλους από ότι ήταν οι γονείς τους, αλλά αυτοί δεν θα έχουν μεγάλη πιθανότητα αναπαραγωγής στον επόμενο κύκλο επειδή θα έχουν χαμηλή καταλληλόλητα.

Η διασταύρωση είναι μια απαραίτητη λειτουργία που συμβάλλει καθοριστικά στην απόδοση του γενετικού αλγορίθμου. Ανάλογα με τον τύπο του προβλήματος επιλέγεται η πιο κατάλληλη μέθοδος διασταύρωσης. Η διασταύρωση είναι χρήσιμη και για την ανακατεύθυνση της αναζήτησης της βέλτιστης λύσης σε ανεξερεύνητες περιοχές του χώρου αναζήτησης. Με αυτό τον τρόπο διευρύνεται το εύρος του γενετικού αλγορίθμου και αυξάνονται οι πιθανότητες για διεκπεραίωση του στόχου του.

Η διασταύρωση λαμβάνει χώρα με μια πιθανότητα pc, η οποία καλείται πιθανότητα διασταύρωσης (crossover probability) και είναι αναγκαίος ο προσδιορισμός της. Η πιθανότητα διασταύρωσης καθορίζει το ποσοστό των υποψήφιων γονέων στους οποίους θα εφαρμοστεί διασταύρωση και κατ' επέκταση και του ποσοστού του τελικού πληθυσμού που θα προέρχεται από διασταύρωση. Η πιθανότητα διασταύρωσης ποικίλει ανάλογα με το πρόβλημα και είναι δυνατό να μεταβληθεί κατά τη διάρκεια εκτέλεσης του γενετικού αλγορίθμου. Τιμές που έχουν προταθεί από ερευνητές είναι pc=0.6 , pc=0.95 , pc=1 και pc=[0.75,0.95]. Η πιθανότητα διασταύρωσης επηρεάζει το ρυθμό σύγκλισης και το χρόνο εκτέλεσης του γενετικού αλγορίθμου. Αν είναι ίση με 1, τότε θα εφαρμόζεται συνεχώς η λειτουργία της διασταύρωσης, δηλαδή όλα τα ζεύγη του συνόλου των υποψήφιων γονέων κάθε πληθυσμού θα υπόκεινται διασταύρωση κι έτσι η αναζήτηση θα γίνει σε όλο το χώρο αναζήτησης με αποτέλεσμα τη σύγκλιση του γενετικού αλγορίθμου στη βέλτιστη λύση αλλά με χρονοτριβή. Αν έχει μικρές τιμές, η αναζήτηση θα γίνεται με μεγάλο βήμα κι έτσι ο αλγόριθμος είναι πιθανότερο να συγκλίνει πιο σύντομα. Αν όμως το βήμα είναι πολύ μεγάλο είναι πιθανό ο γενετικός αλγόριθμος να προσπεράσει τη βέλτιστη λύση και να ξεκινήσει να αποκλίνει. Η συνήθης τακτική που ακολουθείται είναι η επιλογή μεγάλου βήματος στην αρχή

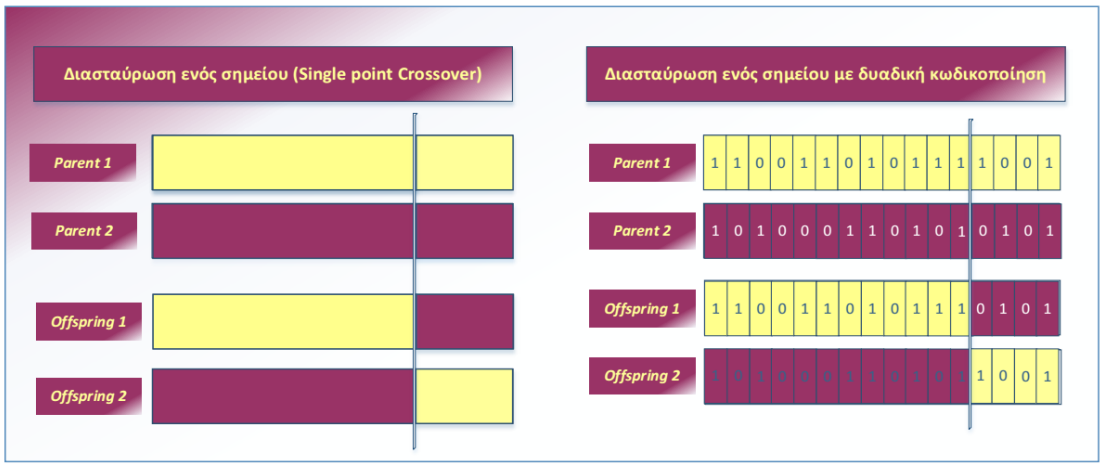
του γενετικού αλγορίθμου και όταν ο αλγόριθμος αρχίσει να πλησιάζει τη βέλτιστη λύση τότε επανακαθορίζεται η πιθανότητα διασταύρωσης επιβάλλοντας μικρό βήμα αναζήτησης. Με αυτή την τακτική, μειώνεται ο κίνδυνος απόκλισης του γενετικού αλγορίθμου και είναι δυνατό να αυξηθεί η ταχύτητα σύγκλισης.

Ακολούθως, παρατίθενται ενδεικτικά βασικές μέθοδοι διασταύρωσης (crossover).

3.7.1.1 Διασταύρωση ενός σημείου (One/Single Point crossover)

Η διασταύρωση ενός σημείου θεωρείται η απλούστερη μορφή του τελεστή διασταύρωσης και έχει προταθεί από τον Holland [15]. Από κάθε ζεύγος γονέων παράγεται ένα νέο ζεύγος απογόνων.

Επιλέγεται ένα μοναδικό κοινό σημείο διασταύρωσης και στους δύο γονείς-χρωμοσώματα, το αρχικό τμήμα του πρώτου απογόνου προκύπτει από την αντιγραφή της ακολουθίας του πρώτου γονέα από το σημείο έναρξης έως το σημείο διασταύρωσης, και το υπόλοιπό του τμήμα από την αντιγραφή της ακολουθίας του δεύτερου γονέα από το σημείο διασταύρωσης μέχρι το τέλος της ακολουθίας. Αντίστοιχα δημιουργείται και ο δεύτερος απόγονος παίρνοντας το αρχικό του τμήμα από το δεύτερο γονέα και το υπόλοιπο από τον πρώτο γονέα.



Εικόνα 3.6: Παράδειγμα διασταύρωσης ενός σημείου με δυαδική κωδικοποίηση

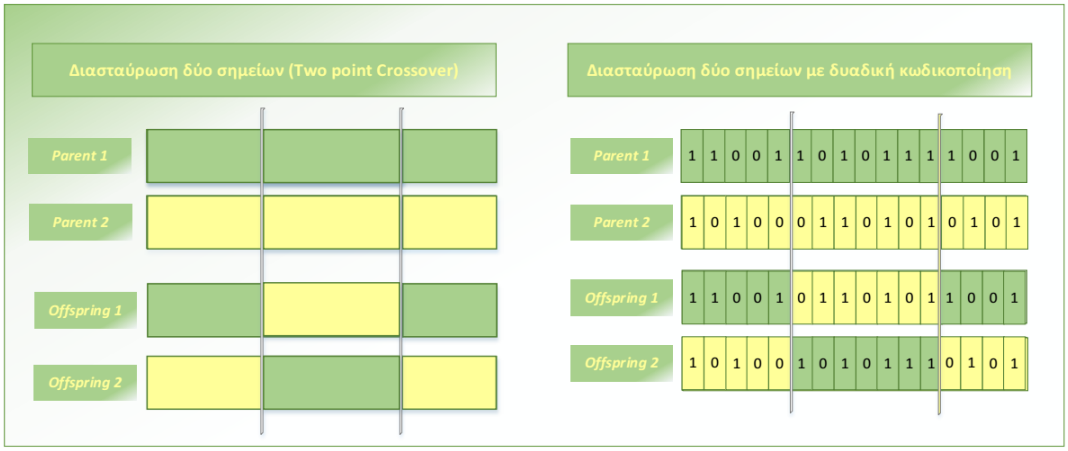
Το τυχαία επιλεγμένο, κοινό και για τους δύο γονείς, σημείο διαχωρισμού, τους χωρίζει σε δύο τμήματα. Αυτή η μέθοδος δίνει περισσότερες πιθανότητες σε γειτονικά γονίδια από κάθε άτομο-γονέα να κληρονομηθούν στον ίδιο απόγονο από ότι σε γονίδια με μια απόσταση μεταξύ τους. Σε περίπτωση που και τα δύο άκρα ενός χρωμοσώματος-γονέα περιέχουν καλό γενετικό υλικό θα παρουσιαστεί πρόβλημα, αφού κανένας απόγονός του δεν θα έχει τα καλά χαρακτηριστικά και των δύο άκρων [16]. Το φαινόμενο αυτό καλείται από

τους Eshelman, Caruana και Schaffer πόλωση θέσεως (positional bias) και αποτελεί το κύριο μειονέκτημα της διασταύρωσης ενός σημείου, καθώς περιορίζει την αποτελεσματικότητα του γενετικού αλγορίθμου. Όσο αυξάνονται τα σημεία διασταύρωσης τόσο η επίδραση της πόλωσης θέσεως μειώνεται. Για αποφυγή των αρνητικών συνεπειών της πόλωσης θέσης, οι εξαρτώμενες μεταβλητές του προβλήματος μπορούν να τοποθετηθούν σε κοντινές θέσεις στην αναπαράσταση του χρωμοσώματος.

Παρόλα αυτά, σε προβλήματα παραμετρικής βελτιστοποίησης, ο τελεστής διασταύρωσης ενός σημείου φαίνεται να είναι ανώτερος από τους υπόλοιπους τελεστές διασταύρωσης ως προς την απόδοση του γενετικού αλγορίθμου [15].

3.7.1.2 Διασταύρωση δύο σημείων (Two point crossover)

Στη διασταύρωση δύο σημείων επιλέγονται δύο σημεία διασταύρωσης στους δύο γονείς-χρωμοσώματα. Τα γονίδια αριστερά και δεξιά των σημείων διασταύρωσης αντιγράφονται από τους γονείς στους απογόνους τους όπως έχουν, ενώ όλα τα γονίδια μεταξύ των δύο σημείων διασταύρωσης στα χρωμοσώματα-γονείς ανταλλάσσονται μεταξύ τους για τη δημιουργία των απογόνων. Το πρόβλημα της πόλωσης θέσης που παρατηρείται στη διασταύρωση ενός σημείου, το υπερβαίνει σε ένα βαθμό η διασταύρωση δύο σημείων αφού αντί για ένα, υπάρχουν δύο σημεία διαχωρισμού, τα οποία διαιρούν το χρωμόσωμα σε τρία τμήματα.



Εικόνα 3.7: Παράδειγμα διασταύρωσης δύο σημείων με δυαδική κωδικοποίηση

3.7.1.3 Διασταύρωση πολλαπλών σημείων (Multi-point crossover)

Η μετάλλαξη συνήθως Η μέθοδος αυτή, για κάθε ζεύγος γονέων επιλέγει τυχαία έναν προκαθορισμένο αριθμό σημείων διασταύρωσης, και καθένας από τους δύο απογόνους που

παράγονται παίρνει ένα τμήμα από κάθε γονέα εναλλάξ. Μπορεί να επιλεγεί είτε μονός αριθμός σημείων διασταύρωσης είτε ζυγός αριθμός. Εάν είναι ζυγός, τα σημεία διασταύρωσης επιλέγονται τυχαία κυκλικά (τα μισά γονίδια από κάθε γονιό διατηρούνται) και η πληροφορία ανταλλάσσεται χωρίς προβλήματα. Στην περίπτωση που είναι μονός αριθμός, διαπιστώνεται πρόβλημα πόλωσης θέσης αφού η πιθανότητα ανταλλαγής ενός γονιδίου εξαρτάται από τη θέση του γονιδίου [15]. Έτσι, εάν επιλεγεί μονός αριθμός, ένα επιπλέον διαφορετικό σημείο διασταύρωσης θεωρείται στην αρχή της ακολουθίας [16]. Με τη διασταύρωση πολλαπλών σημείων, μειώνεται η επίδραση της πόλωσης θέσης ακόμα περισσότερο από ότι με τη διασταύρωση δύο σημείων.

3.7.2 Μετάλλαξη (Mutation)

Η μετάλλαξη συνήθως εφαρμόζεται μετά το στάδιο της διασταύρωσης. Σε αντίθεση με τη διασταύρωση εφαρμόζεται σε ένα μόνο χρωμόσωμα κάθε φορά. Στη βιολογία, η μετάλλαξη είναι μια μόνιμη αλλαγή στη σειρά των γονιδίων στο DNA ή γενικά οποιαδήποτε μεταβολή μπορεί να συμβεί στο γενετικό υλικό ενός οργανισμού. Στους γενετικούς αλγορίθμους κατ' αναλογία, ο τελεστής της μετάλλαξης αλλάζει τυχαία την τιμή ενός ή περισσοτέρων γονιδίων σε ένα χρωμόσωμα. Μπορεί να εφαρμοστεί, είτε σε κάποιο αντίγραφο ενός γονέα, έτσι ώστε να δημιουργήσει ένα νέο άτομο, είτε σε κάποιο απόγονο ενός ζεύγους γονέων για να τον μεταλλάξει.

Ο στόχος της μετάλλαξης είναι η εξερεύνηση του προηγουμένως απρόσιτου χώρου αναζήτησης. Η μετάλλαξη λειτουργεί ως ασφαλιστική δικλείδα σε περίπτωση που η επιλογή και η διασταύρωση ενδεχομένως χάσουν πολύτιμες γενετικές πληροφορίες. Αντίθετα με τον τελεστή της διασταύρωσης, ο τελεστής της μετάλλαξης μπορεί να εισάγει νέες τιμές στα γονίδια των χρωμοσωμάτων, οι οποίες δεν ήταν μέχρι τώρα παρούσες στον πληθυσμό. Έτσι, εισάγοντας καινούρια πληροφορία στον πληθυσμό, λειτουργεί ως τελεστής διατάραξης του πληθυσμού και συμβάλλει στην ποικιλομορφία των παραγόμενων γενεών. Οι γενετικοί αλγόριθμοι, εξαιτίας της στοχαστικής φύσης των τελεστών μετάλλαξης, μπορούν να διαφύγουν από τοπικά ελάχιστα / μέγιστα στο χώρο αναζήτησης, σε αντίθεση με άλλους αλγόριθμους που μένουν παγιδευμένοι εκεί.

Ο τελεστής της μετάλλαξης εφαρμόζεται με μια πιθανότητα μετάλλαξης pm. Η πιθανότητα μετάλλαξης υποδεικνύει πόσο συχνά θα μεταλλαχθούν τα γονίδια ενός χρωμοσώματος. Εάν δεν υπάρχει καθόλου μετάλλαξη, οι απόγονοι θα παραχθούν αμέσως μετά τη διασταύρωση ή θα αντιγραφούν κατευθείαν χωρίς καμία αλλαγή. Εάν εφαρμοστεί μετάλλαξη, ένα ή περισσότερα γονίδια του χρωμοσώματος θα αλλάξουν. Εάν η πιθανότητα μετάλλαξης είναι 1, τότε όλα τα γονίδια του χρωμοσώματος θα μεταλλαχθούν, ενώ εάν είναι 0, κανένα γονίδιο δεν θα μεταλλαχθεί [16]. Απαιτείται προσοχή στην επιλογή της τιμής της

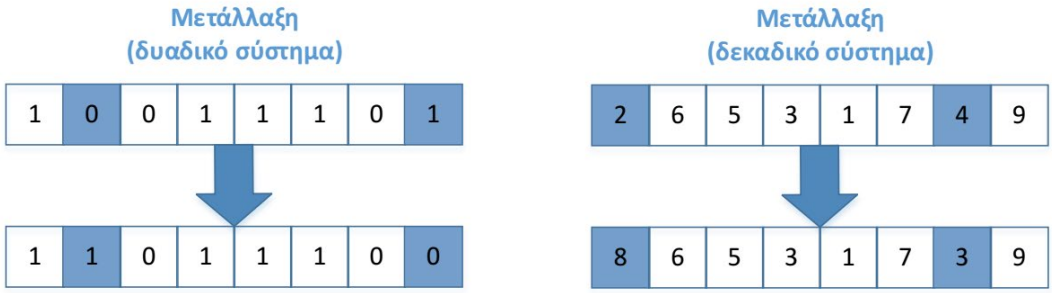
πιθανότητας μετάλλαξης, αφού αν είναι μεγάλη τα χρωμοσώματα δεν θα μπορούν να διατηρήσουν τα καλά δομικά στοιχεία τους, με κίνδυνο ο γενετικός αλγόριθμος να μετατραπεί σε αλγόριθμο τυχαίας αναζήτησης και να μη συγκλίνει σε κανένα ακρότατο. Η τιμή που της ανατίθεται συνήθως είναι σχετικά χαμηλή. Για αλγόριθμους δυαδικής αναπαράστασης η τιμή της πιθανότητας μετάλλαξης επιλέγεται συνήθως στο διάστημα [0.01, 0.1]. Σε περίπτωση άλλων αναπαραστάσεων πολλές φορές είναι αρκετά μεγαλύτερη. Οι Schaffer et al. αναφέρουν ότι η απόδοση των γενετικών αλγορίθμων μειώνεται, είτε όταν είναι ταυτόχρονα μεγάλος και ο πληθυσμός (περισσότερα από 200 άτομα) και η πιθανότητα μετάλλαξης (μεγαλύτερη από 0.05), είτε όταν συνδυάζεται μικρός πληθυσμός (λιγότερα από 20 άτομα) και μικρή πιθανότητα μετάλλαξης (μικρότερη από 0.002) [20].

Η διαδικασία ενός κλασσικού παραδείγματος απλής μορφής μετάλλαξης αναλύεται στα παρακάτω βήματα:

1. Επιλέγεται ένα χρωμόσωμα από το σύνολο των απογόνων που προέκυψαν μετά το στάδιο της διασταύρωσης (ή εναλλακτικά κατευθείαν από το σύνολο των υποψήφιων γονέων που προέκυψαν από το στάδιο της επιλογής).

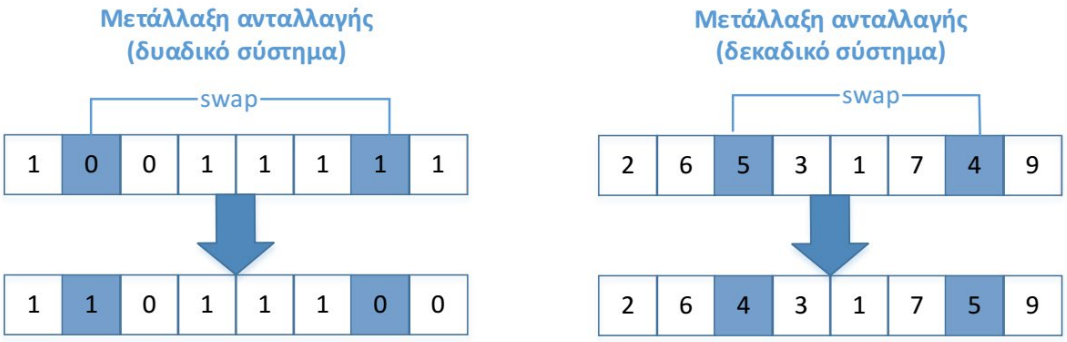
2. Με δεδομένη την πιθανότητα μετάλλαξης p_m για κάθε γονίδιο του χρωμοσώματος, επιλέγεται με ομοιόμορφη κατανομή μια τιμή στο διάστημα [0,1]. Εάν αυτή η τιμή είναι μεγαλύτερη από την πιθανότητα μετάλλαξης, η διαδικασία συνεχίζεται για το επόμενο γονίδιο. Εάν είναι μικρότερη ή ίση, εφαρμόζεται μετάλλαξη στο τρέχον γονίδιο.

3. Στην περίπτωση που πραγματοποιείται μετάλλαξη θα δοθεί μια τυχαία, αλλά διαφορετική τιμή από την τρέχουσα στο γονίδιο αυτό: Αν η τιμή του γονιδίου είναι σε δυαδική μορφή τότε η τιμή του απλά αντιστρέφεται. Αν η τιμή του γονιδίου είναι σε ακέραια ή δεκαδική μορφή τότε παίρνει μια τυχαία τιμή από τις υπόλοιπες που ανήκουν στο επιτρεπτό διάστημα τιμών του γονιδίου.



Εικόνα 3.8: Μετάλλαξη σε χρωμόσωμα δυαδικής και δεκαδικής αναπαράστασης

Ανάλογα με τον τύπο των δεδομένων χρησιμοποιείται κι η ανάλογη τεχνική μετάλλαξης. Μια συνηθισμένη τεχνική μετάλλαξης είναι η μετάλλαξη ανταλλαγής (swap), όπου δύο τυχαίες θέσεις της ακολουθίας επιλέγονται και τα αντίστοιχα σε αυτές τις θέσεις στοιχεία ανταλλάσσονται [16].



Εικόνα 3.9: Μετάλλαξη ανταλλαγής σε χρωμόσωμα δυαδικής και δεκαδικής αναπαράστασης

3.8 Κριτήριο τερματισμού (Termination criterion)

Κριτήριο τερματισμού είναι η συνθήκη που πρέπει να ικανοποιηθεί, προκειμένου να σταματήσει η εκτέλεση του αλγόριθμου [23].

Κάποιες από τις συνθήκες τερματισμού είναι οι ακόλουθες:

- Εύρεση λύσης που πληροί κάποια ελάχιστα κριτήρια.
- Μέγιστος αριθμός γενεών (Maximum Generations): ο αλγόριθμος σταματά με την επίτευξη συγκεκριμένου αριθμού γενεών. Αυτό είναι και το σύνηθες κριτήριο που χρησιμοποιείται είτε μόνο του είτε σε συνδυασμό με κάποιο άλλο κριτήριο.
- Κατακερματισμός διαθέσιμων πόρων για την επίλυση (ειδικά χρόνος): ορισμός κάποιου μέγιστου επιτρεπόμενου χρονικού ορίου και τερματισμός της γενετικής διαδικασίας όταν ο συγκεκριμένος χρόνος λειτουργίας παρέλθει. Σε περίπτωση συνδυασμού με το κριτήριο μέγιστου αριθμού γενεών (όπως συνηθίζεται), όταν ο μέγιστος αριθμός γενεών έχει συμπληρωθεί πριν τον προκαθορισμένο χρόνο τερματισμού η διαδικασία τερματίζεται τότε.
- Καμία αλλαγή στην καταλληλόλητα – Στασιμότητα της βελτίωσης της καταλληλόλητας του πληθυσμού κάτω από ένα προκαθορισμένο κατώφλι για κάποιο δεδομένο αριθμό επαναλήψεων. Η λύση με την υψηλότερη καταλληλόλητα (πρόβλημα μεγιστοποίησης), είτε προσεγγίζεται, είτε έχει ήδη εντοπιστεί, οπότε, αφού άλλες επαναλήψεις δεν θα παράγουν πλέον καλύτερα αποτελέσματα, δεν υπάρχει λόγος για συνέχιση της εκτέλεσης του γενετικού αλγορίθμου.

- Γενεές αναβολής (Stall generations): ο αλγόριθμος σταματά εάν δεν υπάρχει βελτίωση στη συνάρτηση καταλληλόλητας για μια ακολουθία διαδοχικών γενεών μήκους ίσου με τον αριθμό των γενεών αναβολής.
- Όριο χρόνου αναβολής (Stall time limit): ο αλγόριθμος σταματά εάν δεν υπάρχει βελτίωση στη συνάρτηση καταλληλόλητας κατά τη διάρκεια μιας περιόδου χρόνου ίσης σε δευτερόλεπτα με το όριο χρόνου αναβολής.
- Άνω όριο στον αριθμό αξιολογήσεων της συνάρτησης καταλληλόλητας: όταν επιτευχθεί ένας συγκεκριμένος αριθμός αξιολογήσεων της συνάρτησης καταλληλόλητας ο γενετικός αλγόριθμος σταματά.
- Παράθυρο μεταβολής: ο γενετικός αλγόριθμος τερματίζει αν για καθορισμένο αριθμό γενεών η μέση τιμή της καταλληλόλητας του πληθυσμού δεν έχει αισθητή βελτίωση.
- Κριτήριο ποιότητας - Καλύτερο άτομο (Best individual): ο γενετικός αλγόριθμος σταματά αν το ικανότερο χρωμόσωμα μιας γενεάς έχει καταλληλόλητα καλύτερη από ένα προκαθορισμένο κατώφλι καταλληλόλητας (Fitness threshold). Έτσι η αναζήτηση γίνεται πιο γρήγορα και εγγυείται τουλάχιστον μια καλή λύση.
- Μείωση της ποικιλομορφίας του πληθυσμού κάτω από ένα προκαθορισμένο κατώφλι.
- Σύγκλιση γονιδίου (Gene convergence): ο γενετικός αλγόριθμος σταματά όταν ένα προκαθορισμένο ποσοστό των γονιδίων των χρωμοσωμάτων θεωρούνται ότι έχουν συγκλίνει. Ένα γονίδιο θεωρείται ότι έχει συγκλίνει όταν η μέση τιμή αυτού του γονιδίου σε όλα τα χρωμοσώματα του τρέχοντος πληθυσμού είναι μικρότερη κατά ένα συγκεκριμένο ποσοστό από τη μέγιστη τιμή του γονιδίου αυτού σε όλα τα χρωμοσώματα.
- Σύγκλιση πληθυσμού (Population Convergence) : ο γενετικός αλγόριθμος σταματά όταν ο πληθυσμός θεωρείται ότι έχει συγκλίνει. Ο πληθυσμός θεωρείται ότι έχει συγκλίνει όταν η μέση τιμή καταλληλόλητας στον τρέχων πληθυσμό είναι μικρότερη κατά ένα προκαθορισμένο ποσοστό από την καλύτερη καταλληλόλητα του τρέχοντος πληθυσμού.
- Χειροκίνητη παρεμβολή.
- Κάποιος συνδυασμός των προαναφερθέντων κριτηρίων τερματισμού.

Κεφάλαιο 4

4. Σχηματισμός ετερογενών ομάδων εργασίας φοιτητών με τη χρήση γενετικού αλγορίθμου βασισμένου σε πολλαπλά χαρακτηριστικά

4.1 Εισαγωγή

Ο σχηματισμός ομάδας είναι ένα κρίσιμο έργο στη συνεργατική μάθηση, καθώς ο τρόπος με τον οποίο δημιουργούνται οι ομάδες και η σύνθεση των φοιτητών τους επηρεάζουν τα μαθησιακά αποτελέσματα, και κατά συνέπεια, την αποδοτικότητα και την αποτελεσματικότητα των φοιτητών. Οι περισσότερες μελέτες επικεντρώνονται στην ομαδοποίηση με βάση τις πληροφορίες του προφίλ αυτών και την απόδοσή τους σε κάποιο μάθημα [8]. Ωστόσο, η υιοθέτηση των κοινωνικών δεξιοτήτων των φοιτητών ως πρόσθετο χαρακτηριστικό για το σχηματισμό ομάδας μπορεί να οδηγήσει σε επαρκείς ομάδες στις οποίες βελτιώνεται η αλληλεπίδραση των μελών, ενισχύεται η συμμετοχή των φοιτητών και αυξάνεται το δυναμικό της ομάδας.

Υπάρχουν τέσσερις διαφορετικοί τύποι ομάδων στο πρόβλημα σχηματισμού ομάδας. Πρώτον οι ομοιογενείς ομάδες, οι οποίες περιλαμβάνουν μέλη με κοινά χαρακτηριστικά, δεύτερον οι ετερογενείς ομάδες, οι οποίες περιλαμβάνουν μέλη με τιμές διαφορικών χαρακτηριστικών, τρίτον οι μικτές ομάδες, οι οποίες περιλαμβάνουν μέλη με κάποια κοινά χαρακτηριστικά, αλλά όχι όλα και τέταρτον οι ισορροπημένες ομάδες, οι οποίες περιλαμβάνουν μέλη με κατανεμημένα χαρακτηριστικά, δηλαδή π.χ. οι ισχυροί φοιτητές ομαδοποιούνται με αδύναμους και μέσους φοιτητές, και ως εκ τούτου, δεν υπάρχουν «ισχυρές» ή «αδύναμες» ομάδες [8]. Η επιλογή του κατάλληλου τύπου ομάδας εξαρτάται από το μαθησιακό πλαίσιο και τους εκπαιδευτικούς στόχους. Ωστόσο, ο τύπος της ομάδας είναι ένας άλλος παράγοντας που πρέπει να ληφθεί υπόψη, καθώς παίζει σημαντικό ρόλο στη μέθοδο που χρησιμοποιείται για τη σύνθεση ομάδων.

Υπάρχουν διάφοροι τρόποι σχηματισμού ομάδων: α. τυχαία επιλογή: οι ομάδες δημιουργούνται τυχαία, β. αυτό-επιλογή: οι φοιτητές επιλέγουν τους συμφοιτητές τους, γ. επιλογή καθηγητή: ο καθηγητής ορίζει χειροκίνητα τη σύνθεση της ομάδας και δ. αυτόματη επιλογή: το σύστημα δημιουργεί αυτόματα τις ομάδες χρησιμοποιώντας μια μέθοδο βάσει αλγορίθμου [12]. Στις τρεις πρώτες περιπτώσεις υπάρχουν πολλά προβλήματα, όπως για παράδειγμα η τυχαία επιλογή δημιουργεί μη ισορροπημένες ομάδες, η αυτοεπιλογή απαιτεί μεγάλες κοινωνικές σχέσεις και η επιλογή δασκάλου δεν μπορεί να χειριστεί μεγάλο αριθμό φοιτητών και πολύπλοκα κριτήρια ομαδοποίησης. Ως εκ τούτου, για να ξεπεραστούν αυτά τα ζητήματα, η αυτόματη επιλογή είναι μια λύση, η οποία είναι η πιο χρησιμοποιούμενη

στρατηγική ομαδοποίησης. Ωστόσο ο σχηματισμός ομάδας πραγματοποιείται κυρίως από καθηγητές ή φοιτητές, όπου οι ομάδες δομούνται χειροκίνητα λαμβάνοντας υπόψη τις στρατηγικές διδασκαλίας ή τα ενδιαφέροντα των φοιτητών. Επομένως, η εφαρμογή μιας αυτόματης προσέγγισης που θα λαμβάνει υπόψη μια σειρά από χαρακτηριστικά των φοιτητών και θα χρησιμοποιεί έξυπνες τεχνικές για τη δομή των βέλτιστων ομάδων, είναι απαραίτητη για την προώθηση της αποτελεσματικής συνεργατικής μάθησης.

Η υιοθέτηση ενός αποτελεσματικού αλγορίθμου για τη δόμηση ομάδων φοιτητών είναι μια πρόκληση και η ερευνητική κοινότητα έχει εισαγάγει διάφορες αλγοριθμικές προσεγγίσεις σε αυτό το πεδίο. Ο γενετικός αλγόριθμος (GA) είναι μια από τις πιο ευρέως χρησιμοποιούμενες προσεγγίσεις για προβλήματα σχηματισμού ομάδων τα τελευταία χρόνια, καθώς μπορεί να χειριστεί πολλές μεταβλητές και να δημιουργήσει βέλτιστες λύσεις σε σύντομο χρονικό διάστημα [13]. Ο γενετικός αλγόριθμος (GA) είναι ένας μεταευρετικός αλγόριθμος βασισμένος στην έννοια της θεωρίας της εξέλιξης του Δαρβίνου, που εισήχθη επίσημα από τον John Holland το 1975, ενώ το 1992, ο Emmanuel Falkenauer πρότεινε τον Γενετικό Αλγόριθμο Ομαδοποίησης (GGA), ξεπερνώντας τις δυσκολίες του παραδοσιακού γενετικού αλγόριθμου σε ζητήματα ομαδοποίησης [17].

Υπάρχουν διάφορες μελέτες που υιοθετούν προσεγγίσεις γενετικού αλγορίθμου με σκοπό τη δομή αποτελεσματικών ομάδων συνεργασίας μεγιστοποιώντας τα μαθησιακά αποτελέσματα των φοιτητών. Εστιάζουν κυρίως σε συγκροτημένες ετερογενείς [18]–[21] ή μικτές ομάδες [22]. Το βασικό χαρακτηριστικό βάσει του οποίου σχηματίστηκαν οι ομάδες είναι το επίπεδο γνώσης του φοιτητή [18], [19], [21], [22], ενώ άλλα χαρακτηριστικά των φοιτητών που χρησιμοποιούνται για αυτήν την εργασία είναι: στυλ μάθησης, ηγετικές/επικοινωνιακές δεξιότητες [18], [20] και προτιμήσεις συνεργάτη [22]. Όσον αφορά τις γενετικές ρυθμίσεις, οι περισσότερες από τις μελέτες εφαρμόζουν γνωστούς τελεστές διασταύρωσης [19], [20], [22], όπως τάξης, ενός σημείου ή μερικώς χαρτογραφημένη, ενώ άλλες μελέτες εισάγουν νέες προσεγγίσεις διασταύρωσης [18], [21]. Επιπλέον, ο πιο συχνά χρησιμοποιούμενος τελεστής μετάλλαξης είναι η μετάλλαξη ανταλλαγής (swap) [21], [22] και η μετατόπιση [18], [20].

Λαμβάνοντας υπόψη τα παραπάνω, παρουσιάζεται στην συνέχεια ένας νέος γενετικός αλγόριθμος για το σχηματισμό ομάδων φοιτητών εργασίας, βελτιώνοντας τη συνεργασία της ομάδας και ενισχύοντας την παραγωγικότητα. Οι καινοτομίες της προτεινόμενης προσέγγισης είναι τα χαρακτηριστικά που χρησιμοποιούνται για την ομαδοποίηση και οι λειτουργίες που εφαρμόζονται. Ειδικότερα, τα χαρακτηριστικά με βάση τα οποία σχηματίζονται οι ομάδες, αφορούν τις τρεις βασικές διαστάσεις της μάθησης: ακαδημαϊκή, γνωστική και κοινωνική. Όσον αφορά τους γενετικούς τελεστές, ο αλγόριθμος εκτελεί δύο νέους τελεστές

διασταύρωσης: μια τροποποίηση διασταύρωσης 2 σημείων και μια νέα προσέγγιση, δηλαδή διασταύρωση ενός σημείου ανά ομάδα.

4.2 Διατύπωση προβλήματος

Μέσω αυτής της διαδικασίας, γίνεται μοντελοποίηση του προβλήματος και καθορίζονται τα συστατικά μέρη του, δηλαδή οι μεταβλητές, οι περιορισμοί και οι στόχοι του. Ακολούθως ορίζεται ο σχηματισμός ομάδας.

Σκεφτείτε μια διαδικτυακή τάξη S που αποτελείται από n φοιτητές που θα πρέπει να χωριστούν σε ομάδες των m φοιτητών. Κάθε φοιτητής χαρακτηρίζεται από k χαρακτηριστικά, τα οποία αναπαρίστανται σε ένα πολυδιάστατο διάνυσμα. Έστω $S = \{s_1, s_2, ..., s_n\}$ το σύνολο των n φοιτητών της τάξης, $s_i = (c_{i1}, c_{i2}, ..., c_{ik})$ το διάνυσμα των k χαρακτηριστικών του φοιτητή i, και $G = \{g_1, g_2, ..., g_z\}$ το σύνολο των ομάδων z. Στόχος είναι να σχηματιστούν ομάδες z = n / m με βάση τα ακόλουθα κριτήρια: α. κάθε φοιτητής θα πρέπει να τοποθετηθεί σε μία ακριβώς ομάδα, β. κάθε ομάδα θα πρέπει να έχει μια ετερογενή σύνθεση που να σχετίζεται με τα χαρακτηριστικά των μελών, δηλαδή τα μέλη να έχουν όσο το δυνατόν περισσότερα διαφορετικά χαρακτηριστικά, γ. οι ομάδες θα πρέπει να έχουν την υψηλότερη ομοιογένεια μεταξύ τους. Επομένως, ο στόχος βελτιστοποίησης αυτού του προβλήματος είναι να ελαχιστοποιήσει την ολική συνάρτηση καταλληλόλητας ομάδας και ορίζεται ως:

$$ChromosomeFitness = \sum_{j=1}^{z} GroupFitness(g_j) \quad (4.1)$$

όπου GroupFitness είναι η συνάρτηση καταλληλόλητας που εκχωρεί μια τιμή σε κάθε ομάδα λαμβάνοντας υπόψη τα χαρακτηριστικά των μελών της ομάδας, όπως περιγράφονται παρακάτω.

Όπως αναφέρθηκε παραπάνω, η ετερογενής ομαδοποίηση κατανέμει τους φοιτητές μεταξύ των ομάδων με τρόπο ώστε τα μέλη κάθε ομάδας να έχουν διαφορετικά χαρακτηριστικά. Αυτή η ποικιλομορφία μπορεί να είναι ευεργετική για τη βελτίωση των μαθησιακών αποτελεσμάτων, καθώς το δυναμικό της ομάδας μπορεί να αυξηθεί μέσω των διαφορετικών χαρακτηριστικών των μελών [23], [24]. Για παράδειγμα, η ομαδοποίηση φοιτητών με διαφορετικό βαθμό λανθασμένης αντίληψης μπορεί να βοηθήσει στην επιτυχή υλοποίηση της εργασίας από το να υπάρχουν φοιτητές με τον ίδιο βαθμό λανθασμένης αντίληψης σε μια ομάδα ή η ομαδοποίηση φοιτητών που είναι εσωστρεφείς μπορεί να προκαλέσει αποτυχία της εργασίας. Η βασική υπόθεση είναι ότι οι ομάδες έχουν καλύτερο αποτέλεσμα όταν οι φοιτητές που τις συνθέτουν έχουν διαφορετικά επιτεύγματα, διαφορετικό βαθμό λανθασμένης αντίληψης, άλλους λειτουργικούς ρόλους, άλλες προσωπικότητες κ.λπ.

Επιπλέον, η ετερογενής ομαδοποίηση παρέχει μεγαλύτερες ευκαιρίες μάθησης σε φοιτητές με χαμηλό επίπεδο ικανότητας σε σύγκριση με την ομοιογενή ομαδοποίηση, χωρίς να είναι επιζήμια για φοιτητές με υψηλό επίπεδο ικανότητας [8].

4.2.1 Αναπαράσταση φοιτητών

Κάθε φοιτητής i αναπαρίσταται χρησιμοποιώντας ένα διάνυσμα, το μέγεθος του οποίου εξαρτάται από τον αριθμό των χαρακτηριστικών που χρησιμοποιούνται για να σχηματιστούν οι ομάδες:

$s_i = (c_{i1}, c_{i2}, \dots , c_{ik})$, (4.2)

i: το αναγνωριστικό φοιτητή, k: ο αριθμός των χαρακτηριστικών, και c_{ij}: η τιμή του χαρακτηριστικού j

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΦΟΙΤΗΤΩΝ ΓΙΑ ΣΧΗΜΑΤΙΣΜΟ ΟΜΑΔΑΣ

ΜΑΘΗΣΙΑΚΗ ΔΙΑΣΤΑΣΗ	ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ	ΒΑΘΜΟΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
ΑΚΑΔΗΜΑΪΚΗ	1.ΔΗΜΙΟΥΡΓΙΚΗ ΔΕΞΙΟΤΗΤΑ 2.ΔΕΞΙΟΤΗΤΑ ΑΞΙΟΛΟΓΗΣΗΣ 3.ΑΝΑΛΥΤΙΚΗ ΔΕΞΙΟΤΗΤΑ 4.ΔΕΞΙΟΤΗΤΑ ΕΦΑΡΜΟΓΗΣ 5.ΔΕΞΙΟΤΗΤΑ ΚΑΤΑΝΝΟΫΣΗΣ 6.ΔΕΞΙΟΤΗΤΑ ΜΝΗΜΗΣ	ΠΟΛΥ ΧΑΜΗΛΗ, ΧΑΜΗΛΗ, ΚΑΛΗ, ΠΟΛΥ ΚΑΛΗ, ΕΞΑΙΡΕΤΙΚΗ
	7.ΓΝΩΣΤΙΚΟ ΕΠΙΠΕΔΟ	ΠΡΩΤΟ ΕΠΙΠΕΔΟ, ΔΕΥΤΕΡΟ ΕΠΙΠΕΔΟ, ΤΡΙΤΟ ΕΠΙΠΕΔΟ
ΓΝΩΣΤΙΚΗ	8.ΣΤΥΛ ΜΑΘΗΣΗΣ	ΑΚΤΙΒΙΣΤΗΣ, ΣΤΟΧΑΣΤΗΣ, ΘΕΩΡΙΤΙΚΟΣ, ΠΡΑΓΜΑΤΙΣΤΗΣ
	9.ΒΑΘΜΟΣ ΛΑΝΘΑΣΜΕΝΗΣ ΑΝΤΙΛΗΨΗΣ	ΑΠΡΟΣΕΚΤΟΣ, ΠΙΘΑΝΩΣ ΑΠΡΟΣΕΚΤΟΣ,ΠΙΘΑΝΩΣ ΑΔΙΑΒΑΣΤΟΣ, ΑΔΙΑΒΑΣΤΟΣ
	10.ΔΙΑΓΡΑΜΜΑ ΠΕΡΙΠΤΩΣΗΣ ΧΡΗΣΗΣ 11.ΔΙΑΓΡΑΜΜΑ ΤΑΞΗΣ 12.ΔΙΑΓΡΑΜΜΑ ΚΑΤΑΣΤΑΣΗΣ 13. ΔΙΑΓΡΑΜΜΑ ΑΚΟΛΟΥΘΙΑΣ	(ΛΑΝΘΑΣΜΕΝΗ ΑΝΤΙΛΗΨΗ) ΣΕ ΚΑΠΟΙΑ ΥΠΟΕΝΟΤΗΤΑ, ΣΕ ΚΑΜΙΑ ΥΠΟΕΝΟΤΗΤΑ
ΚΟΙΝΩΝΙΚΗ	14.ΕΠΙΠΕΔΟ ΑΝΑΓΝΩΡΙΣΗΣ 15.ΕΠΙΠΕΔΟ ΣΥΝΕΙΣΦΟΡΑΣ 16.ΕΠΙΠΕΔΟ ΔΗΜΙΟΥΡΓΙΑΣ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ 17. ΚΟΙΝΩΝΙΚΗ ΣΥΜΜΕΤΟΧΗ	ΠΡΩΤΟ ΕΠΙΠΕΔΟ, ΔΕΥΤΕΡΟ ΕΠΙΠΕΔΟ, ΤΡΙΤΟ ΕΠΙΠΕΔΟ

Πίνακας 4.1

Αυτά τα χαρακτηριστικά θα μπορούσαν να έχουν διαφορετική φύση. Σε παρούσα εργασία, τα χαρακτηριστικά που λαμβάνονται υπόψη για τη σύνθεση της ομάδας προκύπτουν από τις τρεις κύριες διαστάσεις της μάθησης: ακαδημαϊκή, γνωστική και κοινωνική (Πίνακας 4.1). Ειδικότερα, τα χαρακτηριστικά περιλαμβάνουν:

Ακαδημαϊκή μάθηση: απόδοση βασισμένη σε δεξιότητες σκέψης αναθεωρημένης ταξινόμησης Bloom.

Γνωστική μάθηση: στυλ μάθησης Honey & Mumford, βαθμός λανθασμένης αντίληψης.

Κοινωνική μάθηση: σχετίζεται με κοινωνικές δεξιότητες.

Από ακαδημαϊκής πλευράς, η απόδοση των φοιτητών στις αξιολογήσεις μαθημάτων είναι βασικό χαρακτηριστικό σε προβλήματα σχηματισμού ομάδας [25]. Η αξιολόγηση των φοιτητών αναφέρει το επίπεδό τους για κάθε δεξιότητα σκέψης με βάση την αναθεωρημένη ταξινόμηση Bloom, δηλαδή δημιουργία, αξιολόγηση, ανάλυση, εφαρμογή, κατανόηση και μνήμη. Αυτά τα έξι χαρακτηριστικά χρησιμοποιούνται για τον χαρακτηρισμό της απόδοσης των φοιτητών και οι τιμές που μπορούν να τεθούν σε καθεμία είναι πολύ χαμηλή, χαμηλή, καλή, πολύ καλή και εξαιρετική. Επιπλέον, υπάρχουν τρία επίπεδα γνώσης και ο κάθε φοιτητής ανήκει σε κάποιο από τα τρία αυτά επίπεδα, όπου το επίπεδο-1 είναι το χαμηλότερο και το επίπεδο-3 το υψηλότερο.

Λαμβάνοντας υπόψη τη γνωστική διάσταση, το μαθησιακό στυλ είναι ένα από τα κύρια χαρακτηριστικά του φοιτητή σε ένα εκπαιδευτικό περιβάλλον και μπορεί να επηρεάσει τη συλλογική εργασία. Η ομαδοποίηση φοιτητών με διαφορετικά στυλ μάθησης δημιουργεί διαφορετικές προοπτικές για την αποτελεσματική συνεργασία στην ομάδα [26]. Σε αυτά που ακολουθούν, το στυλ μάθησης Honey & Mumford δεν χρησιμοποιείται μόνο για την παροχή εξατομικευμένης διδασκαλίας, αλλά και για τη δημιουργία ομάδων. Οι τύποι φοιτητών είναι ακτιβιστές, στοχαστές, θεωρητικοί και πραγματιστές. Επιπλέον η ενότητα αξιολόγησης σπουδαστών αναφέρεται στον βαθμό της λανθασμένης αντίληψης, δηλαδή εάν οφείλεται σε απροσεξία ή σε κενό γνώσης. Έτσι, κάθε φοιτητής χαρακτηρίζεται από ένα βαθμό λανθασμένης αντίληψης ως απρόσεκτος, πιθανώς απρόσεκτος, πιθανώς αδιάβαστος και αδιάβαστος. Επιπλέον, αυτή η ενότητα προσδιορίζει για κάθε κεφάλαιο των προγραμμάτων σπουδών (π.χ. διάγραμμα περίπτωσης χρήσης, διάγραμμα τάξης, διάγραμμα κατάστασης και διάγραμμα ακολουθίας που αφορούν την UML) την ενότητα όπου ο φοιτητής έχει μια εσφαλμένη αντίληψη. Ο βαθμός λανθασμένης αντίληψης των φοιτητών και οι παρανοήσεις είναι απαραίτητες για τη δημιουργία ετερογενών ομάδων, καθώς η ομαδοποίηση φοιτητών με διαφορετικά γνωστικά κενά μπορεί να ενισχύσει το δυναμικό της ομάδας. Διαφορετικά, για παράδειγμα, εάν οι φοιτητές μιας ομάδας έχουν τον ίδιο βαθμό λανθασμένης αντίληψης στο διάγραμμα της τάξης, μπορεί να αντιμετωπίσουν δυσκολίες στην κατασκευή αυτού του διαγράμματος κατά την εκπόνηση της εργασίας.

Όσον αφορά την κοινωνική πτυχή, οι κοινωνικές αλληλεπιδράσεις και η συμμετοχή των φοιτητών είναι σημαντικές για την προώθηση της μαθησιακής διαδικασίας. Στο

σχηματισμό ομάδας, η επιτυχία της παράδοσης της εργασίας που εκτελείται, εξαρτάται από την αποτελεσματικότητα της συνεργασίας των φοιτητών. Εάν οι εσωστρεφείς φοιτητές ομαδοποιηθούν, αυτό μπορεί να προκαλέσει προβλήματα στην επικοινωνία τους. Ενώ, εάν ένας κοινωνικός φοιτητής ανήκει σε αυτήν την ομάδα, μπορεί να ενθαρρύνει άλλους να εμπλακούν στη διαδικασία του έργου. Ως εκ τούτου, η ομαδοποίηση φοιτητών με διαφορετικές κοινωνικές δεξιότητες οδηγεί σε καλύτερες αλληλεπιδράσεις. Χρησιμοποιώντας αυτήν την ενότητα, μπορούμε να προσδιορίσουμε το επίπεδο κοινωνικής αναγνώρισης των φοιτητών, της συνεισφοράς, της δημιουργίας εκπαιδευτικού υλικού και την κοινωνική συμμετοχή στην ομάδα.

Σύμφωνα με τα παραπάνω, επιλέγονται δεκαεπτά χαρακτηριστικά για να αντιπροσωπεύουν το προφίλ του φοιτητή, βάσει του οποίου εκτελείται ο σχηματισμός ομάδας. Η σύνθεση ομάδων στις οποίες τα μέλη έχουν μια ποικιλία από αυτά τα χαρακτηριστικά, μπορεί να οδηγήσει σε αποτελεσματική συνεργασία και καλύτερα μαθησιακά αποτελέσματα τόσο σε ομαδικό όσο και σε ατομικό επίπεδο.

4.2.2 Συνάρτηση καταλληλόλητας

Η συνάρτηση καταλληλόλητας εκχωρεί μια τιμή σε κάθε χρωμόσωμα, η οποία καθορίζει την καταλληλόλητα της ομαδοποίησης κάθε ομάδας φοιτητών. Ο σκοπός αυτής της συνάρτησης είναι να μεγιστοποιήσει τη συμπληρωματικότητα μεταξύ των μελών κάθε ομάδας και να ελαχιστοποιήσει τη συνολική διακύμανση μεταξύ των ομάδων. Η συνάρτηση καταλληλόλητας για το χρωμόσωμα δίνεται με τον υπολογισμό της καταλληλόλητας κάθε ομάδας.

Σε αυτή την εργασία, η καταλληλόλητα μιας ομάδας βασίζεται στις εμφανίσεις κάθε χαρακτηριστικού μεταξύ των μελών της ομάδας [20]. Μια χαμηλή τιμή καταλληλόλητας σε μια ομάδα υποδηλώνει ότι οι επαναλήψεις των τιμών των ίδιων χαρακτηριστικών είναι οι ελάχιστες. Η καταλληλόλητα της ιδανικής ομάδας είναι ίση με μηδέν, που σημαίνει ότι κανένα από τα χαρακτηριστικά δεν επαναλαμβάνεται μεταξύ των μελών της ομάδας και έτσι επιτυγχάνεται η βέλτιστη ετερογένεια. Ωστόσο, αυτή η ιδανική τιμή δεν μπορεί να επιτευχθεί εάν τα χαρακτηριστικά έχουν λιγότερες διακριτές τιμές από το μέγεθος της ομάδας.

Το πρώτο βήμα για τον υπολογισμό της καταλληλόλητας κάθε ομάδας είναι να καθοριστεί η ιδανική επανάληψη για κάθε χαρακτηριστικό, δηλαδή ο ελάχιστος αριθμός επαναλαμβανόμενων εναλλακτικών χαρακτηριστικών. Επομένως, ένας μετρητής c_{ij} ορίζεται για να αντιπροσωπεύει τις εναλλακτικές εμφανίσεις του j του χαρακτηριστικού i σε μια ομάδα. Αυτός ο μετρητής υψώνεται στη δύναμη του τρία προκειμένου να τονιστεί η διαφορά στην καταλληλόλητα όταν επαναλαμβάνεται ένα χαρακτηριστικό.

$$IdealRepitition(i) = \begin{cases} m, & s \geq m \\ \sum_{j=1}^{s} (c_{ij})^3, & s < m \text{ and } \sum_{j=1}^{s} c_{ij} = m \end{cases} \quad (4.3)$$

όπου s: αριθμός εναλλακτικών τιμών του χαρακτηριστικού i, c_{ij}: επαναλήψεις του εναλλακτικού j του χαρακτηριστικού i, και m: μέγεθος ομάδας.

Εάν ένα χαρακτηριστικό έχει περισσότερες ή ίσες εναλλακτικές τιμές από το μέγεθος της ομάδας, τότε η ιδανική επανάληψη είναι ίση με το μέγεθος της ομάδας, διαφορετικά εφαρμόζεται κάποια «ποινή». Για παράδειγμα, θεωρείστε το μέγεθος της ομάδας να είναι 4 φοιτητές. Έτσι, εάν ένα χαρακτηριστικό έχει περισσότερες από 4 εναλλακτικές τιμές, σημαίνει ότι μια ομάδα μπορεί να δομηθεί με φοιτητές που παρουσιάζουν διαφορετικές τιμές χαρακτηριστικών, και τότε η ιδανική επανάληψη για αυτό το χαρακτηριστικό είναι 4 (1^3+1^3+1^3+1^3). Εάν ένα χαρακτηριστικό έχει 3 εναλλακτικές, τότε στην καλύτερη περίπτωση που επαναλαμβάνεται μία εναλλακτική τιμή η ιδανική τιμή επανάληψης είναι 10 (1^3+1^3+2^3).

Μετά τον υπολογισμό της ιδανικής επανάληψης για όλα τα χαρακτηριστικά, η καταλληλόλητα μιας συγκεκριμένης ομάδας προκύπτει από την απόσταση μεταξύ των τιμών επανάληψης χαρακτηριστικών και των ιδανικών τιμών επανάληψης χαρακτηριστικών. Συγκεκριμένα, για κάθε χαρακτηριστικό i, η επανάληψη κάθε εναλλακτικής τιμής που εμφανίζεται στην ομάδα υπολογίζεται ως εξής:

$$Repitition(i) = \sum_{j=1}^{s} (c_{ij})^3, \quad s \leq m \quad (4.4)$$

όπου s: αριθμός εναλλακτικών τιμών του χαρακτηριστικού i στην ομάδα, cij: επαναλήψεις του εναλλακτικού j του χαρακτηριστικού i στην ομάδα και m: μέγεθος ομάδας. Κατά συνέπεια, η καταλληλόλητα της ομάδας διαμορφώνεται ως εξής:

$$GroupFitness(g_j) = \sum_{i=1}^{k} (Repitition(i) - IdealRepitition(i)) \quad (4.5)$$

όπου g_j: ομάδα j του χρωμοσώματος, k: ο αριθμός των χαρακτηριστικών.

Όσο χαμηλότερη είναι η τιμή αυτής της συνάρτησης, τόσο πιο ετερογενής ομάδα έχει σχηματιστεί. Η βέλτιστη τιμή ορίζεται στο μηδέν. Ως εκ τούτου, για να δημιουργηθούν οι καλύτερες ετερογενείς ομάδες, η τιμή καταλληλόλητας ομάδας θα πρέπει να ελαχιστοποιηθεί.

4.3 Ο προτεινόμενος γενετικός αλγόριθμος ομαδοποίησης

Ο κύριος στόχος του προτεινόμενου αλγορίθμου είναι να βελτιώσει την ποιότητα του σχηματισμού ομάδας, άρα και την αποτελεσματικότητα της συνεργατικής μάθησης. Προς

αυτή την κατεύθυνση, δοκιμάζεται ένα σύνολο διαμορφώσεων και εισάγονται τροποποιήσεις γενετικών τελεστών. Η ροή του γενετικού αλγορίθμου που χρησιμοποιείται για το σχηματισμό ομάδων φοιτητών είναι:

Βήμα 1. Προσδιορισμός των χαρακτηριστικών των φοιτητών. Το πρώτο βήμα είναι να προσδιοριστούν τα χαρακτηριστικά των φοιτητών με βάση τα οποία πραγματοποιήθηκε ο σχηματισμός της ομάδας. Αυτό είναι ζωτικής σημασίας για τη δόμηση κατάλληλων ομάδων που προάγουν την αποτελεσματική και αποδοτική συνεργασία και για την επίτευξη καλύτερων μαθησιακών αποτελεσμάτων. Τα χαρακτηριστικά που χρησιμοποιούνται σε αυτή την εργασία περιγράφονται στην ενότητα 4.2.1.

Βήμα 2. Αριθμητική κατηγοριοποίηση των χαρακτηριστικών των φοιτητών. Εκτελείται μια διαδικασία κατηγοριοποίησης αριθμητικών χαρακτηριστικών, αφού η συνάρτηση καταλληλόλητας εξετάζει τα επαναλαμβανόμενα χαρακτηριστικά που εμφανίζονται σε μια ομάδα και όχι την απόσταση μεταξύ των τιμών. Ως εκ τούτου, οι βαθμοί των φοιτητών, στην ακαδημαϊκή διάσταση, κατανέμονται στις κατηγορίες: πολύ χαμηλός, χαμηλός, καλός, πολύ καλός και άριστος.

Βήμα 3. Καθορισμός των γενετικών ρυθμίσεων. Πριν από την εκτέλεση του γενετικού αλγόριθμου, θα πρέπει να οριστούν το μέγεθος της ομάδας, το μέγεθος του πληθυσμού, οι γενιές, το μέγεθος τουρνουά για τη διαδικασία επιλογής, το ποσοστό ελιτισμού και η πιθανότητα διασταύρωσης και μετάλλαξης.

Βήμα 4. Κωδικοποίηση του χρωμοσώματος. Σε αυτό το βήμα, το χρωμόσωμα αναπαρίσταται σε μια προκαθορισμένη δομή δεδομένων προκειμένου να επιτραπεί η εφαρμογή γενετικών τελεστών. Σε αυτή τη μελέτη, χρησιμοποιείται μια δομή πίνακα όπως περιγράφεται στην επόμενη υποενότητα.

Βήμα 5. Αρχικοποίηση πληθυσμού. Ο γενετικός αλγόριθμος ξεκινά με τη δημιουργία ενός αρχικού πληθυσμού που αποτελείται από ορισμένες εφικτές κωδικοποιημένες λύσεις (χρωμοσώματα). Αυτός ο πληθυσμός δημιουργείται τυχαία για να διασφαλιστεί η ποικιλομορφία του.

Βήμα 6. Αξιολόγηση της καταλληλόλητας. Μια συνάρτηση καταλληλόλητας χρησιμοποιείται για την αξιολόγηση των χρωμοσωμάτων του πληθυσμού με βάση τα χαρακτηριστικά των φοιτητών. Όσο χαμηλότερη τιμή καταλληλόλητας έχει ένα χρωμόσωμα, τόσο καλύτερη λύση αντιπροσωπεύει.

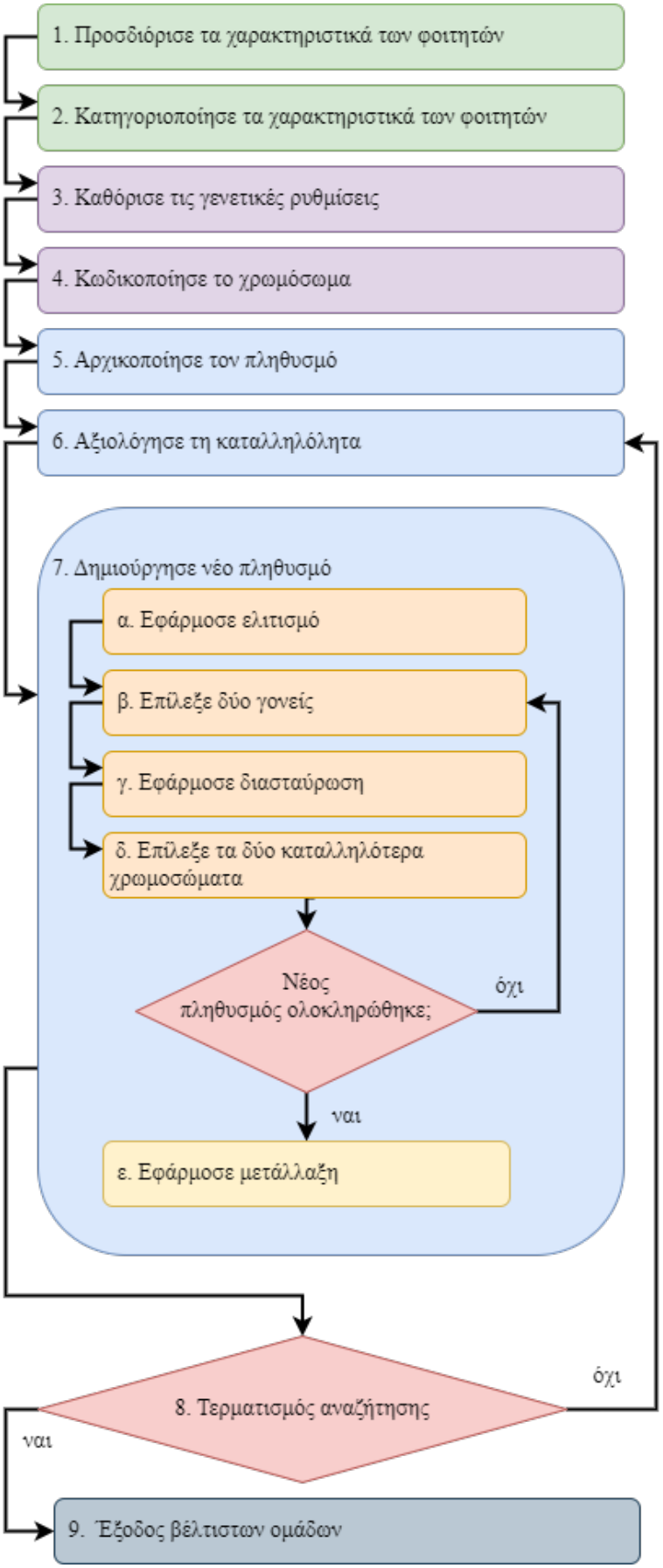
Βήμα 7. Δημιουργία νέου πληθυσμού. Αυτή η διαδικασία είναι η καρδιά του γενετικού αλγορίθμου, όπου δημιουργούνται νέες και καλύτερες λύσεις. Οι γενετικοί τελεστές που εφαρμόζονται σε αυτό το βήμα είναι: α) ελιτισμός, όπου ένα ποσοστό από τα καλύτερα

χρωμοσώματα περιλαμβάνεται στον νέο πληθυσμό, β) επιλογή, όπου επιλέγονται δύο γονείς για διασταύρωση, γ) διασταύρωση, όπου με βάση μια πιθανότητα πραγματοποιείται συνδυασμός των γονιδίων των γονέων με σκοπό την παραγωγή καλύτερων απογόνων και δ) μετάλλαξη, όπου με βάση μια πιθανότητα μεταλλάσσονται τμήματα του χρωμοσώματος του νέου πληθυσμού.

Βήμα 8. Τερματισμός αναζήτησης. Μετά από αρκετές γενιές, ο αλγόριθμος τερματίζει και συγκλίνει στο πιο κατάλληλο χρωμόσωμα, το οποίο αντιπροσωπεύει τη βέλτιστη λύση.

Βήμα 9. Έξοδος βέλτιστων ομάδων. Το σύστημα δημιουργεί τις ομάδες των φοιτητών με βάση τα αποτελέσματα γενετικού αλγορίθμου και ειδοποιεί τους φοιτητές να ξεκινήσουν ομαδική εργασία για να παραδώσουν το έργο που τους έχει ανατεθεί.

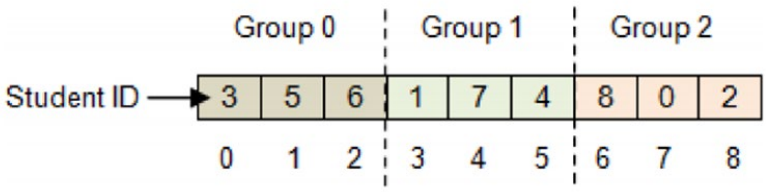
Παρακάτω παρουσιάζεται ο αλγόριθμος σε διάγραμμα ροής (Εικόνα 4.1):


1. Προσδιόρισε τα χαρακτηριστικά των φοιτητών
2. Κατηγοριοποίησε τα χαρακτηριστικά των φοιτητών
3. Καθόρισε τις γενετικές ρυθμίσεις
4. Κωδικοποίησε το χρωμόσωμα
5. Αρχικοποίησε τον πληθυσμό
6. Αξιολόγησε τη καταλληλόλητα
7. Δημιούργησε νέο πληθυσμό
α. Εφάρμοσε ελιτισμό
β. Επίλεξε δύο γονείς
γ. Εφάρμοσε διασταύρωση
δ. Επίλεξε τα δύο καταλληλότερα χρωμοσώματα
Νέος πληθυσμός ολοκληρώθηκε;
όχι
ναι
ε. Εφάρμοσε μετάλλαξη
8. Τερματισμός αναζήτησης
όχι
ναι
9. Έξοδος βέλτιστων ομάδων

Εικόνα 4.1

4.3.1 Κωδικοποίηση

Για να εφαρμοστεί ο γενετικός αλγόριθμος, είναι απαραίτητο να κωδικοποιηθούν οι μεταβλητές του προβλήματος ομαδοποίησης στο χρωμόσωμα. Κάθε φοιτητής πρέπει να ανήκει σε μια ομάδα μόνο. Γι' αυτό το λόγο χρησιμοποιείται για την αναπαράσταση του χρωμοσώματος κωδικοποίηση μεταλλαγής (Permutation encoding). Δεδομένου ότι υπάρχουν n φοιτητές για να σχηματίσουν ομάδες, το χρωμόσωμα κωδικοποιείται ως δομή δεδομένων πίνακα με μήκος ίσο με n. Σε κάθε γονίδιο (στοιχείο πίνακα) εκχωρείται ένας αριθμός που αντιστοιχεί στο φοιτητικό id. Η θέση του γονιδίου στον πίνακα δίνει την ομάδα στην οποία ανήκει ο φοιτητής. δηλ., έστω το p συμβολίζει τη θέση και το m το μέγεθος της ομάδας, τότε ο φοιτητής στη θέση p αντιστοιχίζεται στην ομάδα p div m. Η Εικόνα 4.2 αντιπροσωπεύει το σχήμα κωδικοποίησης που χρησιμοποιείται με ένα παράδειγμα ομαδοποίησης 9 φοιτητών σε ομάδες των 3.



Εικόνα 4.2

4.3.2 Αρχικοποίηση

Ο αλγόριθμος ξεκινά με τη δημιουργία ενός αρχικού πληθυσμού που αποτελείται από έναν ορισμένο αριθμό εφικτών κωδικοποιημένων λύσεων. Αυτός ο αριθμός λύσεων αναφέρεται στο μέγεθος του πληθυσμού και παραμένει σταθερός κατά τη διαδικασία του γενετικού αλγορίθμου. Όπως αναφέρθηκε παραπάνω, ο αρχικός πληθυσμός παράγεται χρησιμοποιώντας μια τυχαία μέθοδο προκειμένου να διασφαλιστεί η ποικιλομορφία του πληθυσμού και να βελτιωθεί η σύγκλιση προς την καλύτερη λύση.

4.3.3 Δημιουργία νέου πληθυσμού

Αυτή είναι η κύρια διαδικασία ενός γενετικού αλγορίθμου. Μέσω αυτής της διαδικασίας, ένας νέος πληθυσμός δημιουργείται χρησιμοποιώντας γενετικούς τελεστές, με στόχο την παραγωγή του πιο κατάλληλου χρωμοσώματος και συνεπώς τη σύγκλιση προς τη βέλτιστη λύση. Η υιοθέτηση των κατάλληλων γενετικών τελεστών είναι ζωτικής σημασίας για την απόδοση του γενετικού αλγορίθμου και την ποιότητα και την ακρίβεια των αποτελεσμάτων του [27], [28].

Στον προτεινόμενο αλγόριθμο, ο νέος πληθυσμός χτίζεται χρησιμοποιώντας τη στρατηγική του ελιτισμού, την επιλογή γονέων μέσω μεθόδου τουρνουά, τη χρήση δύο τελεστών διασταύρωσης και τη μετάλλαξη. Συγκεκριμένα, πρώτον, τα καλύτερα χρωμοσώματα του πληθυσμού εισάγονται απευθείας στο νέο. Στη συνέχεια, επιλέγονται δύο γονείς με τη μέθοδο του τουρνουά. Έπειτα γίνεται η εφαρμογή διασταύρωσης βάσει της πιθανότητας που έχει οριστεί, όπου εκτελούνται δύο τελεστές διασταύρωσης στους επιλεγμένους γονείς και τα δύο καλύτερα χρωμοσώματα μεταξύ γονέων και τεσσάρων απογόνων που παράγονται, εισάγονται στον νέο πληθυσμό. Διαφορετικά, οι γονείς αντιγράφονται σε νέο πληθυσμό. Αυτή η εργασία συνεχίζεται μέχρι να δημιουργηθεί νέος πληθυσμός. Οι δύο τελεστές διασταύρωσης που χρησιμοποιούνται είναι μια τροποποίηση διασταύρωσης 2 σημείων και μια εισαγόμενη προσέγγιση, που ονομάζεται διασταύρωση ενός σημείου ανά ομάδα. Τέλος, η μετάλλαξη με αντιμετάθεση (swap) εφαρμόζεται στο νέο πληθυσμό σύμφωνα με σχετική πιθανότητα. Ο Πίνακας 4.2 παρουσιάζει τον αλγόριθμο της δημιουργίας νέου πληθυσμού σε ψευδοκώδικα.

ΨΕΥΔΟΚΩΔΙΚΑΣ: ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ ΠΛΥΘΗΣΜΟΥ

```
ΓΙΑ ΚΑΘΕ ΓΕΝΕΑ
        "ΔΙΑΔΙΚΑΣΙΑ ΕΛΙΤΙΣΜΟΥ"
        ΒΡΕΣ ΤΑ ΚΑΛΥΤΕΡΑ (ΠΟΣΟΣΤΟ_ΕΛΙΤΙΣΜΟΥ * ΜΕΓΕΘΟΣ_ΠΛΥΘΗΣΜΟΥ)
        ΧΡΩΜΟΣΩΜΑΤΑ
        ΑΝΤΕΓΡΑΨΕ ΤΑ ΚΑΛΥΤΕΡΑ ΧΡΩΜΟΣΩΜΑΤΑ ΣΤΟ ΝΕΟ_ΠΛΥΘΗΣΜΟ

        "ΔΙΑΔΙΚΑΣΙΑ ΕΠΙΛΟΓΗΣ ΓΩΝΕΩΝ ΚΑΙ ΔΙΑΣΤΑΥΡΩΣΗΣ"
        ΕΠΑΝΕΛΑΒΕ
                ΕΠΕΛΕΞΕ ΔΥΟ ΓΟΝΕΙΣ ΒΑΣΗ ΤΗΣ ΔΙΑΔΙΚΑΣΙΑΣ ΤΟΥΡΝΟΥΑ
                ΑΝ ΜΠΟΡΕΙ ΝΑ ΕΦΑΡΜΟΣΤΕΙ ΔΙΑΣΤΑΥΡΩΣΗ ΒΑΣΗ ΤΗΝ
                ΠΙΘΑΝΟΤΗΤΑ_ΔΙΑΣΤΑΥΡΩΣΗΣ ΤΟΤΕ
                        ΕΦΑΡΜΟΣΕ ΤΟΥΣ ΤΕΛΕΣΤΕΣ ΔΙΑΣΤΑΥΡΩΣΗΣ
                        ΕΦΑΡΜΟΣΕ ΤΟΥΡΝΟΥΑ ΜΕΤΑΞΥ ΓΟΝΕΩΝ ΚΑΙ ΠΑΙΔΙΩΝ
                        ΑΝ ΔΕΝ ΥΠΑΡΧΟΥΝ ΟΙ ΔΥΟ ΝΙΚΗΤΕΣ ΤΟΥ ΤΟΥΡΝΟΥΑ ΣΤΟ
                        ΝΕΟ_ΠΛΥΘΗΣΜΟ ΤΟΤΕ
                                ΑΝΤΕΓΡΑΨΕ ΤΟΥΣ ΔΥΟ ΝΙΚΗΤΕΣ ΤΟΥ ΤΟΥΡΝΟΥΑ ΣΤΟ
                                ΝΕΟ_ΠΛΥΘΗΣΜΟ
                        ΤΕΛΟΣ_ΑΝ
                ΑΛΛΙΩΣ
                        ΑΝ ΟΙ ΓΟΝΕΙΣ ΔΕΝ ΑΝΗΚΟΥΝ ΣΤΟ ΝΕΟ_ΠΛΥΘΗΣΜΟ ΤΟΤΕ
                                ΑΝΤΕΓΡΑΨΕ ΤΟΥΣ ΓΟΝΕΙΣ ΣΤΟ ΝΕΟ_ΠΛΥΘΗΣΜΟ
                        ΤΕΛΟΣ_ΑΝ
                ΤΕΛΟΣ_ΑΝ
        ΜΕΧΡΙΣ_ΟΤΟΥ ΤΟ ΜΕΓΕΘΟΣ ΣΤΟ ΝΕΟ_ΠΛΥΘΗΣΜΟ ΝΑ ΓΙΝΕΙ ΙΣΟ ΜΕ ΤΟ ΜΕΓΕΘΟΣ
        ΠΛΥΘΗΣΜΟΥ ΠΟΥ ΕΧΕΙ ΟΡΙΣΤΕΙ
```

```
"ΔΙΑΔΙΚΑΣΙΑ ΜΕΤΑΛΛΑΞΗΣ"
ΓΙΑ ΚΑΘΕ ΕΝΑ ΧΡΩΜΟΣΩΜΑ
        ΑΝ ΜΠΟΡΕΙ ΝΑ ΕΦΟΡΜΟΣΤΕΙ ΜΕΤΑΛΛΑΞΗ ΒΑΣΗ ΤΗΝ
        ΠΙΘΑΝΟΤΗΤΑΣ_ΜΕΤΑΛΛΑΞΗΣ ΤΟΤΕ
                ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΔΥΟ ΟΜΑΔΕΣ ΤΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ
                ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΔΥΟ ΓΟΝΙΔΙΑ ΑΥΤΩΝ ΤΩΝ ΟΜΑΔΩΝ
                ΑΝΤΑΛΑΞΕ ΤΑ ΓΟΝΙΔΙΑ
                ΑΝ ΤΟ ΝΕΟ ΧΡΩΜΟΣΩΜΑ ΕΧΕΙ ΜΕΓΑΛΥΤΕΡΗ ΤΙΜΗ  ΣΥΝΑΡΤΗΣΗΣ
                ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΤΟΤΕ
                        ΑΝΤΙΚΑΤΑΣΤΗΣΕ ΤΟ ΠΑΛΑΙΟ ΜΕ ΤΟ ΝΕΟ ΧΡΩΜΟΣΩΜΑ
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
```

Πίνακας 4.2

1) Ελιτισμός: Για να διατηρηθούν τα πιο κατάλληλα χρωμοσώματα σε κάθε γενιά, χρησιμοποιείται μια επιλογή ελιτισμού. Έτσι, ένα μικρό ποσοστό των καλύτερων χρωμοσωμάτων αντιγράφεται στην επόμενη γενιά. Αυτή η στρατηγική εγγυάται ότι το καλύτερο χρωμόσωμα ενός πληθυσμού θα είναι καλύτερο ή τουλάχιστον ίσο με το καλύτερο χρωμόσωμα του προηγούμενου πληθυσμού. Επιπλέον, δεν οδηγεί τον αλγόριθμο σε πρόωρη σύγκλιση καθώς το ποσοστό ελιτισμού επιλέγεται να είναι πολύ μικρό. Το ποσοστό ελιτισμού αυτής της προσέγγισης ορίζεται στο 0,1.

2) Επιλογή: Ο χειριστής επιλογής αποτελείται από την επιλογή των καταλληλότερων χρωμοσωμάτων που θα εισαχθούν σε μια δεξαμενή ζευγαρώματος, έτσι ώστε αυτά ή ένα μέρος τους να αντιγραφούν στην επόμενη γενιά. Αυτή η διαδικασία βασίζεται στην αρχή του καταλληλότερου χρωμοσώματος που έχει επιβιώσει και στην ελπίδα ότι οι απόγονοι των πιο ικανών χρωμοσωμάτων μπορεί να έχουν καλύτερη καταλληλόλητα. Η επιρροή της επιλογής είναι ο βαθμός στον οποίο προτιμώνται τα καλύτερα άτομα: όσο υψηλότερη είναι η επιρροή της επιλογής, τόσο περισσότερο προτιμώνται τα καλύτερα άτομα και προκύπτει υψηλότερο ποσοστό σύγκλισης.

Η επιλογή πραγματοποιείται χρησιμοποιώντας διαφορετικές μεθόδους, όπως τροχό ρουλέτας, τουρνουά και στοχαστική καθολική δειγματοληψία. Στην εργασία αυτή χρησιμοποιείται η επιλογή τουρνουά μεταξύ των εναλλακτικών στρατηγικών. Σχετικά με αυτή τη μέθοδο, τα k χρωμοσώματα επιλέγονται τυχαία από τον πληθυσμό για να ανταγωνιστούν μεταξύ τους. Το χρωμόσωμα με την καλύτερη καταλληλόλητα είναι ο νικητής και ορίζεται ως γονέας. Αυτή η εργασία επαναλαμβάνεται μέχρι να επιλεγούν δύο γονείς. Όλα τα k χρωμοσώματα επιστρέφουν στον πληθυσμό και έχουν δικαίωμα συμμετοχής ξανά σε άλλα τουρνουά.

Η επιλογή τουρνουά είναι μια ιδανική στρατηγική καθώς μπορεί να προσαρμόσει την επιρροή της επιλογής, αλλάζοντας το πλήθος των χρωμοσωμάτων που έχουν επιλεχθεί για το τουρνουά και να διατηρήσει την ποικιλομορφία του πληθυσμού, καθώς δίνει ίσες ευκαιρίες σε όλα τα χρωμοσώματα να αγωνιστούν [28]. Το μέγεθος του τουρνουά μπορεί να οριστεί ως περίπου το 20% του πληθυσμού σε κάθε γενιά ή η πλησιέστερη δύναμη του 2. Σε αυτόν τον αλγόριθμο, επιλέχθηκε η δεύτερη προσέγγιση, επομένως το μέγεθος του τουρνουά είναι 10 τυχαία χρωμοσώματα (Πίνακας 4.3).

ΨΕΥΔΟΚΩΔΙΚΑΣ: ΕΠΙΛΟΓΗ ΓΟΝΕΩΝ ΜΕ ΤΟΥΡΝΟΥΑ

```
ΟΣΟ ΠΛΗΘΟΣ_ΓΟΝΕΩΝ <> 2
        ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΧΡΩΜΟΣΩΜΑΤΑ ΣΥΜΦΩΝΑ ΜΕ ΤΟ ΜΕΓΕΘΟΣ_ΤΟΥΡΝΟΥΑ
        ΑΝ ΤΟ ΧΡΩΜΟΣΩΜΑ ΜΕ ΤΗ ΜΕΓΑΛΥΤΕΡΗ ΤΙΜΗ ΣΥΝΑΡΤΗΣΗΣ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ
        ΔΕΝ ΕΧΕΙ ΗΔΗ ΕΠΙΛΕΓΕΙ ΩΣ ΓΟΝΕΑΣ ΤΟΤΕ
                ΕΠΕΛΕΞΕ ΤΟ ΩΣ ΓΟΝΕΑ
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Πίνακας 4.3

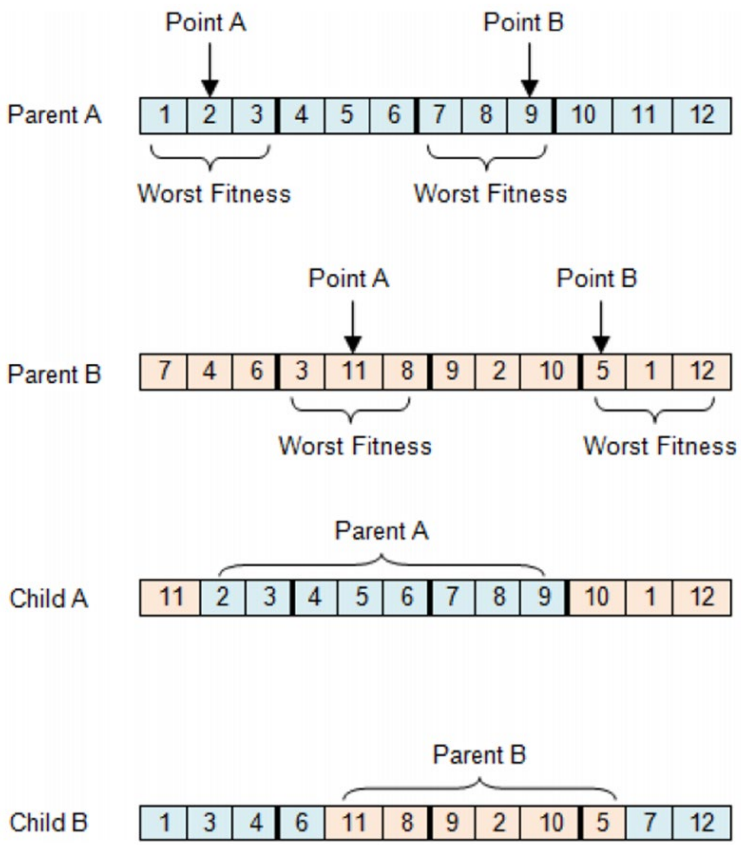
3) Διασταύρωση: Η διασταύρωση είναι η διαδικασία συνδυασμού μερών γονέων στον τρέχοντα πληθυσμό προκειμένου να παραχθούν απόγονοι για τον επόμενο πληθυσμό. Σκοπός αυτού του βήματος είναι ο εμπλουτισμός του νέου πληθυσμού με καλύτερα χρωμοσώματα μέσω της ανταλλαγής γενετικών πληροφοριών. Ο τελεστής διασταύρωσης εφαρμόζεται στους γονείς με βάση μια πιθανότητα διασταύρωσης p_c. Εάν οι γονείς δεν επιλεγούν για διασταύρωση, διατηρούνται χωρίς τροποποίηση στον επόμενο πληθυσμό.

Μερικοί από τους πιο γνωστούς τελεστές διασταύρωσης είναι η διασταύρωση ενός σημείου, η διασταύρωση δύο σημείων, η μερικώς χαρτογραφημένη διασταύρωση, ο κύκλος κ.α. Οι περισσότερες από τις προσεγγίσεις εφαρμόζουν μία μέθοδο διασταύρωσης. Σε αυτή την εργασία, εισάγονται δύο νέες προσεγγίσεις διασταύρωσης που εκμεταλλεύονται τα πλεονεκτήματα γνωστών χειριστών και τα επεκτείνουν με πρόθεση να δημιουργήσουν καλύτερους απογόνους. Η πρώτη προσέγγιση είναι μια τροποποίηση της διασταύρωσης δύο (2) σημείων όπου τα σημεία δεν ορίζονται τυχαία, και η άλλη είναι μια νέα προσέγγιση, που ονομάζεται διασταύρωση ενός (1) σημείου ανά ομάδα, βασισμένη στη φιλοσοφία της διασταύρωσης πολλαπλών σημείων. Μετά την εφαρμογή των τελεστών διασταύρωσης, τα δύο χρωμοσώματα με την καλύτερη καταλληλόλητα μεταξύ των δύο γονέων και των τεσσάρων απογόνων που παράγονται εισάγονται στην επόμενη γενιά.

α) Τροποποίηση διασταύρωσης δύο (2) σημείων: Σε διασταύρωση δύο (2) σημείων, επιλέγονται τυχαία δύο σημεία διασταύρωσης και τα περιεχόμενα μεταξύ αυτών των σημείων

ανταλλάσσονται μεταξύ δύο γονέων. Στην προτεινόμενη τροποποίηση, τα δύο σημεία διασταύρωσης δεν επιλέγονται τυχαία λαμβάνοντας υπόψη ολόκληρο το χρωμόσωμα, αλλά πρώτα επιλέγονται οι δύο ομάδες με τη χειρότερη καταλληλόλητα και προκύπτουν τα σημεία από αυτές τις ομάδες, με τυχαία επιλογή εντός αυτών των ομάδων. Ο λόγος για τον οποίο προτείνεται αυτό το είδος επιλογής σημείων είναι επειδή αποτρέπει την απώλεια κατάλληλων ομάδων καθώς διατηρούνται στον επόμενο πληθυσμό και διασφαλίζει τον συνδυασμό των χειρότερων ομάδων με την ελπίδα ότι θα δημιουργηθούν καλύτεροι απόγονοι.

Πιο συγκεκριμένα, μετά την επιλογή των δύο ομάδων με τη χειρότερη καταλληλόλητα και την τυχαία επιλογή ενός σημείου σε κάθε ομάδα, οι απόγονοι αποτελούνται από τα ίδια ακριβώς γονίδια του ενός γονέα που βρίσκονται ανάμεσα στα δύο σημεία και τα υπόλοιπα γονίδια συμπληρώνονται από τα αχρησιμοποίητα γονίδια του άλλου γονέα με τη σειρά που συναντιούνται στο χρωμόσωμα. Η Εικόνα 4.3 απεικονίζει ένα παράδειγμα της προτεινόμενης τροποποίησης διασταύρωσης δύο (2) σημείων, ενώ ο Πίνακας 4.4 δείχνει τη λογική αυτής της διαδικασίας χρησιμοποιώντας ψευδοκώδικα.



Εικόνα 4.3

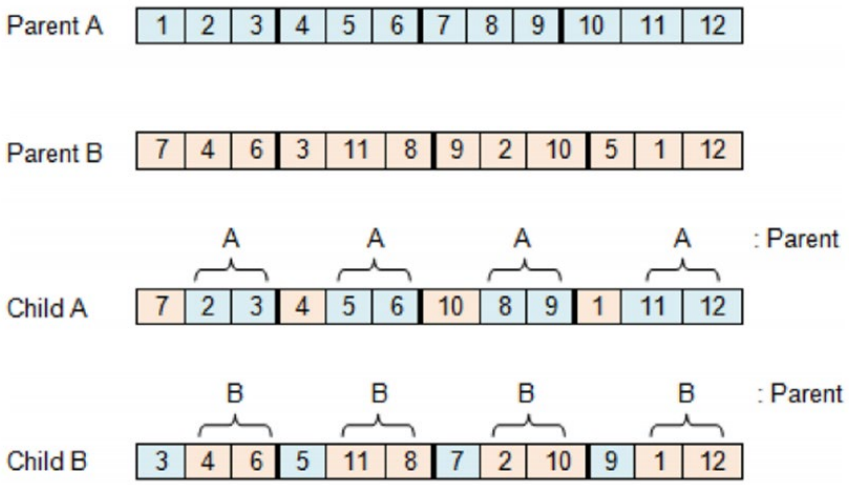
ΨΕΥΔΟΚΩΔΙΚΑΣ: ΕΦΑΡΜΟΓΗ ΤΡΟΠΟΠΟΙΗΜΕΝΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ ΔΥΟ ΣΗΜΕΙΩΝ

```
ΓΙΑ ΚΑΘΕ ΓΟΝΕΑ
    ΒΡΕΣ ΤΙΣ ΔΥΟ ΟΜΑΔΕΣ ΜΕ ΤΗ ΧΕΙΡΟΤΕΡΗ (ΜΕΓΑΛΥΤΕΡΗ) ΤΙΜΗ ΣΥΝΑΡΤΗΣΗΣ
    ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ
    ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΔΥΟ ΔΕΙΚΤΕΣ ΓΙΑ ΤΗΝ ΚΑΘΕ ΟΜΑΔΑ
    ΑΝ ΠΡΩΤΟΣ_ΔΕΙΚΤΗΣ > ΔΕΥΤΕΡΟ_ΔΕΙΚΤΗ ΤΟΤΕ
        ΑΝΤΙΜΕΤΑΘΕΣΕ ΤΟΥΣ ΔΕΙΚΤΕΣ
    ΤΕΛΟΣ_ΑΝ
    ΣΤΟΝ ΠΡΩΤΟ ΑΠΟΓΟΝΟ ΑΝΤΕΓΡΑΨΕ ΤΑ ΓΟΝΙΔΙΑ ΠΟΥ ΒΡΙΣΚΟΝΤΑΙ ΜΕΤΑΞΥ ΤΩΝ
    ΕΠΙΛΕΓΜΕΝΩΝ ΔΕΙΚΤΩΝ ΤΟΥ ΠΡΩΤΟΥ ΓΟΝΕΑ
    ΣΤΟΝ ΔΕΥΤΕΡΟ ΑΠΟΓΟΝΟ  ΑΝΤΕΓΡΑΨΕ ΤΑ ΓΟΝΙΔΙΑ ΠΟΥ ΒΡΙΣΚΟΝΤΑΙ ΜΕΤΑΞΥ
    ΤΩΝ ΕΠΙΛΕΓΜΕΝΩΝ ΔΕΙΚΤΩΝ ΤΟΥ ΔΕΥΤΕΡΟΥ ΓΟΝΕΑ
    ΑΝΤΕΓΡΑΨΕ ΜΕ ΤΗ ΣΕΙΡΑ ΤΑ ΜΗ ΧΡΗΣΙΜΟΠΟΙΗΜΕΝΑ ΓΟΝΙΔΙΑ ΤΟΥ ΔΕΥΤΕΡΟΥ
    ΓΟΝΕΑ ΣΤΟ ΠΡΩΤΟ ΠΑΙΔΙ (ΑΠΟ ΤΗΝ ΑΡΧΗ ΜΕΧΡΙ ΠΡΩΤΟΣ_ΔΕΙΤΗΣ-1 ΚΑΙ ΑΠΟ ΤΟ
    ΔΕΥΤΕΡΟ_ΔΕΙΚΤΗ ΜΕΧΡΙ ΤΕΛΟΣ)
    ΑΝΤΕΓΡΑΨΕ ΜΕ ΤΗ ΣΕΙΡΑ ΤΑ ΜΗ ΧΡΗΣΙΜΟΠΟΙΗΜΕΝΑ ΓΟΝΙΔΙΑ ΤΟΥ ΠΡΩΤΟΥ
    ΓΟΝΕΑ ΣΤΟ ΔΕΥΤΕΡΟ ΠΑΙΔΙ (ΑΠΟ ΤΗΝ ΑΡΧΗ ΜΕΧΡΙ ΠΡΩΤΟΣ_ΔΕΙΤΗΣ-1 ΚΑΙ ΑΠΟ
    ΤΟ ΔΕΥΤΕΡΟ_ΔΕΙΚΤΗ ΜΕΧΡΙ ΤΕΛΟΣ)
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Πίνακας 4.4

β) Μια νέα προσέγγιση, διασταύρωση ενός (1) σημείου ανά ομάδα: Προκειμένου να αναζητηθεί πιο διεξοδικά ο χώρος λύσεων του προβλήματος και να προστεθούν νέες γενετικές πληροφορίες στον πληθυσμό, αποτρέποντας τον αλγόριθμο να συγκλίνει σε ένα τοπικό ελάχιστο, μια νέα προσέγγιση διασταύρωσης παρουσιάζεται σύμφωνα με τη στρατηγική διασταύρωσης πολλαπλών σημείων. Η προτεινόμενη διασταύρωση ενός (1) σημείου ανά ομάδα ανταλλάσσει τα γονίδια κάθε ομάδας μεταξύ των δύο γονέων με την πρόθεση να παράγει νέους πιο κατάλληλους απογόνους. Ως εκ τούτου, οι ομάδες των απογόνων αποτελούνται από τα μισά γονίδια κάθε γονέα. Η Εικόνα 4.4 απεικονίζει ένα παράδειγμα διασταύρωσης ενός (1) σημείου ανά ομάδα, ενώ ο Πίνακας 4.5 περιγράφει τα βήματα αυτής της διαδικασίας χρησιμοποιώντας ψευδοκώδικα.

Αυτή η προσέγγιση αναπτύσσεται λαμβάνοντας υπόψη την αρχή ότι τα συνδυασμένα πιο κατάλληλα χρωμοσώματα μπορούν να παράγουν ακόμη καλύτερους απογόνους. Ο μόνος περιορισμός μπορεί να είναι ο μεγάλος αριθμός ομάδων που μπορεί να επηρεάσουν την απόδοση του αλγορίθμου αυξάνοντας τον χρόνο εκτέλεσης λόγω της πολυπλοκότητας της προσέγγισης. Ωστόσο, στη συνεργατική μάθηση, ο αριθμός των φοιτητικών ομάδων που απαιτείται να σχηματιστούν δεν είναι απαγορευτικά μεγάλος. Σε περίπτωση εξαιρετικά μεγάλου αριθμού ομάδων ή για άμεσα αποδοτικά αποτελέσματα, αυτός ο τελεστής διασταύρωσης μπορεί να εφαρμοστεί με σχετική πιθανότητα, δηλαδή να μην εφαρμόζεται κάθε φορά που αποφασίζεται να γίνει διασταύρωση.

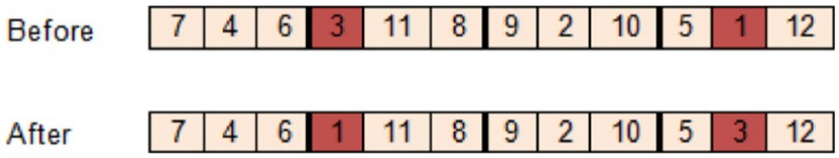


Εικόνα 4.4

ΨΕΥΔΟΚΩΔΙΚΑΣ: ΕΦΑΡΜΟΓΗ ΔΙΑΣΤΑΥΡΩΣΗΣ ΕΝΟΣ ΣΗΜΕΙΟΥ ΑΝΑ ΟΜΑΔΑ
ΟΡΙΣΕ ΤΟΥΣ ΔΕΙΚΤΕΣ ΠΡΩΤΟΥ ΜΕΛΟΥΣ ΚΑΘΕ ΟΜΑΔΑΣ ΓΙΑ ΚΑΘΕ ΔΕΙΚΤΗ ΠΡΩΤΟΥ ΜΕΛΟΥΣ ΟΜΑΔΑΣ ΣΤΟΝ ΠΡΩΤΟ ΑΠΟΓΟΝΟ ΑΝΤΕΓΡΑΨΕ ΑΝΑ ΔΥΟ ΤΑ ΓΟΝΙΔΙΑ ΠΟΥ ΑΠΟΤΕΛΟΥΝ ΤΟ ΔΕΥΤΕΡΟ ΚΑΙ ΤΡΙΤΟ ΜΕΛΟΣ ΚΑΘΕ ΟΜΑΔΑΣ ΤΟΥ ΠΡΩΤΟΥ ΓΟΝΕΑ ΣΤΟΝ ΔΕΥΤΕΡΟ ΑΠΟΓΟΝΟ ΑΝΤΕΓΡΑΨΕ ΑΝΑ ΔΥΟ ΤΑ ΓΟΝΙΔΙΑ ΠΟΥ ΑΠΟΤΕΛΟΥΝ ΤΟ ΔΕΥΤΕΡΟ ΚΑΙ ΤΡΙΤΟ ΜΕΛΟΣ ΚΑΘΕ ΟΜΑΔΑΣ ΤΟΥ ΔΕΥΤΕΡΟΥ ΓΟΝΕΑ ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΑΝΤΕΓΡΑΨΕ ΜΕ ΤΗ ΣΕΙΡΑ ΤΑ ΜΗ ΧΡΗΣΙΜΟΠΟΙΗΜΕΝΑ ΓΟΝΙΔΙΑ ΤΟΥ ΔΕΥΤΕΡΟΥ ΓΟΝΕΑ ΣΤΟ ΠΡΩΤΟ ΠΑΙΔΙ (ΠΡΩΤΟ ΜΕΛΟΣ ΚΑΘΕ ΟΜΑΔΑΣ) ΑΝΤΕΓΡΑΨΕ ΜΕ ΤΗ ΣΕΙΡΑ ΤΑ ΜΗ ΧΡΗΣΙΜΟΠΟΙΗΜΕΝΑ ΓΟΝΙΔΙΑ ΤΟΥ ΠΡΩΤΟΥ ΓΟΝΕΑ ΣΤΟ ΔΕΥΤΕΡΟ ΠΑΙΔΙ (ΠΡΩΤΟ ΜΕΛΟΣ ΚΑΘΕ ΟΜΑΔΑΣ)

Πίνακας 4.5

4) Μετάλλαξη: Η μετάλλαξη είναι η ανταλλαγή δύο τυχαίων γονιδίων και πραγματοποιείται μετά τη δημιουργία του νέου πληθυσμού με πολύ χαμηλή πιθανότητα p_m. Ο ρόλος της μετάλλαξης είναι να εισάγει τυχαίες παραλλαγές, αυξάνοντας τη γενετική ποικιλότητα και έτσι αποτρέποντας τη σύγκλιση σε τοπικά βέλτιστες λύσεις [29]. Σε αυτή τη μελέτη, χρησιμοποιείται μετάλλαξη swap, όπου επιλέγονται τυχαία δύο γονίδια και ανταλλάσσονται οι θέσεις τους. Ένα παράδειγμα swap μετάλλαξης φαίνεται στην Εικόνα 4.5. Η μετάλλαξη swap προτιμήθηκε καθώς είναι μια απλή και αποτελεσματική μέθοδος που δεν δημιουργεί άκυρα χρωμοσώματα όσον αφορά τον αριθμό των ομάδων ή ίδια γονίδια.



Εικόνα 4.5

4.3.4 Τερματισμός αναζήτησης

Το κριτήριο τερματισμού που χρησιμοποιείται σε αυτή τη μελέτη είναι ο αριθμός των γενεών, δηλαδή οι επαναλήψεις παραγωγής νέου πληθυσμού. Έτσι, ο γενετικός αλγόριθμος σταματά όταν έχουν φτάσει καθορισμένες γενιές. Εάν ο αριθμός των γενεών είναι μικρός, η πιθανότητα να βρεθεί η βέλτιστη λύση είναι μικρή. Από την άλλη πλευρά, εάν αυτός ο αριθμός είναι πολύ υψηλός, ο χρόνος εκτέλεσης του αλγορίθμου αυξάνεται.

Κεφάλαιο 5

5. Υλοποίηση δημιουργίας αρχικού πληθυσμού και συνάρτησης καταλληλότητας του γενετικού αλγορίθμου ομαδοποίησης με τη χρήση της γλώσσας Python

Σε αυτό το κεφάλαιο θα γίνει η υλοποίηση των έξι πρώτων βημάτων του γενετικού αλγορίθμου ομαδοποίησης, όπως αναφέρονται στην ενότητα 4.3 του κεφαλαίου 4, με τη χρήση γλώσσας προγραμματισμού Python 3, εκτός από το βήμα 3. Το βήμα αυτό, δηλαδή ο καθορισμός των γενετικών ρυθμίσεων θα αναφερθεί αποκλειστικά στο κεφάλαιο 6, όπου θα σχηματιστεί πλήρως ο γενετικός αλγόριθμος ομαδοποίησης. Παρακάτω γίνεται συνοπτική υπενθύμιση αυτών των βημάτων:

Βήμα 1. Προσδιορισμός των χαρακτηριστικών των φοιτητών.

Βήμα 2. Αριθμητική κατηγοριοποίηση των χαρακτηριστικά των φοιτητών.

Βήμα 4. Κωδικοποίηση του χρωμοσώματος.

Βήμα 5. Αρχικοποίηση πληθυσμού.

Βήμα 6. Αξιολόγηση καταλληλόλητας χρωμοσώματος.

Τα βήματα 1 και 2 θα υλοποιηθούν στην ενότητα 5.1 και στην ενότητα 5.4 με την δημιουργία συνάρτησης (με παράμετρο εισόδου το πλήθος φοιτητών), το βήμα 4 στην ενότητα 5.2, το βήμα 5 στην ενότητα 5.3 και στην ενότητα 5.4 με την δημιουργία συνάρτησης (με παράμετρο εισόδου το πλήθος φοιτητών, και το πλήθος των χρωμοσωμάτων) και τέλος το βήμα 6 υλοποιείται σταδιακά στις ενότητες 5.5, με τη δημιουργία χρήσιμων βοηθητικών συναρτήσεων, και 5.6 με τη δημιουργία της συνάρτησης καταλληλόλητας (βασισμένης στις συναρτήσεις της ενότητας 5.5).

Μπορούμε να τρέξουμε τον κώδικα που θα παρουσιαστεί στην ενότητα αυτή, χρησιμοποιώντας τον διερμηνευτή της Python 3 στο διαδίκτυο στην ιστοσελίδα https://www.programiz.com/python-programming/online-compiler/. Μπορούμε επίσης να κατεβάσουμε κάποια έκδοση της Python 3 από την ιστοσελίδα https://www.python.org/downloads/, στον υπολογιστή μας και να τρέξουμε τον κώδικα τοπικά.

5.1 Υλοποίηση χαρακτηριστικών φοιτητών

Στην υλοποίησή μας θεωρούμε ότι έχουμε 48 φοιτητές τριτοβάθμιας εκπαίδευσης (n = 48). Στα πλαίσια της εκπαίδευσης, οι φοιτητές όφειλαν να εργαστούν σε ομάδες για να παραδώσουν μία εργασία. Έτσι, τοποθετήθηκαν σε ομάδες των τριών ατόμων (m = 3) με σκοπό να σχηματιστούν ετερογενείς ομάδες (z = 16) και λαμβάνοντας υπόψη τα 17

χαρακτηριστικά των φοιτητών, που προέκυψαν από ακαδημαϊκή, γνωστική και κοινωνική άποψη όπως περιγράφεται στην Ενότητα 4.2.1. Στον πίνακα 5.1 παρουσιάζονται αυτά τα 17 χαρακτηριστικά και η αριθμητική κατηγοριοποίησή τους.

Ο λόγος για τον οποίο επιλέχθηκε αυτό το μέγεθος ομάδας (m = 3) είναι ότι σε μικρότερες ομάδες δεν εξασφαλίζεται η απαιτούμενη διαφορετικότητα, ενώ σε μεγαλύτερες είναι δύσκολο να διασφαλιστεί ότι όλα τα μέλη εργάζονται και μαθαίνουν εξίσου. Επιπλέον, η ετερογενής ομαδοποίηση προτιμάται από άλλους τύπους, π.χ. ομοιογενής, καθώς η ύπαρξη μελών με διαφορετικά ακαδημαϊκά και γνωστικά χαρακτηριστικά επιτρέπει στο μέλος που έχει καλύτερη γνώση για μια έννοια να την εξηγήσει σε άλλους που μπορεί να έχουν λανθασμένη αντίληψη, ενώ περιλαμβάνοντας μέλη με διαφορετικά κοινωνικά χαρακτηριστικά επιτρέπει στους φοιτητές με χαμηλές κοινωνικές δεξιότητες να ενθαρρύνονται από άλλους που είναι πιο κοινωνικοί, διευκολύνοντας τη συζήτηση εντός της ομάδας.

ΜΑΘΗΣΙΑΚΗ ΔΙΑΣΤΑΣΗ	ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ	ΒΑΘΜΟΣ (ΜΕ ΑΡΙΘΜΗΤΙΚΗ ΤΙΜΗ) ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
ΑΚΑΔΗΜΑΪΚΗ	0. ΔΗΜΙΟΥΡΓΙΚΗ ΔΕΞΙΟΤΗΤΑ 1. ΔΕΞΙΟΤΗΤΑ ΑΞΙΟΛΟΓΗΣΗΣ 2. ΑΝΑΛΥΤΙΚΗ ΔΕΞΙΟΤΗΤΑ 3. ΔΕΞΙΟΤΗΤΑ ΕΦΑΡΜΟΓΗΣ 4. ΔΕΞΙΟΤΗΤΑ ΚΑΤΑΝΟΗΣΗΣ 5. ΔΕΞΙΟΤΗΤΑ ΜΝΗΜΗΣ	1: ΠΟΛΥ ΧΑΜΗΛΗ 2: ΧΑΜΗΛΗ 3: ΚΑΛΗ 4: ΠΟΛΥ ΚΑΛΗ 5: ΕΞΑΙΡΕΤΙΚΗ (ΑΡΙΣΤΗ)
	6. ΓΝΩΣΤΙΚΟ ΕΠΙΠΕΔΟ	1: ΠΡΩΤΟ ΕΠΙΠΕΔΟ 2: ΔΕΥΤΕΡΟ ΕΠΙΠΕΔΟ 3: ΤΡΙΤΟ ΕΠΙΠΕΔΟ
ΓΝΩΣΤΙΚΗ	7. ΣΤΥΛ ΜΑΘΗΣΗΣ	1: ΑΚΤΙΒΙΣΤΗΣ 2: ΣΤΟΧΑΣΤΗΣ 3: ΘΕΩΡΗΤΙΚΟΣ 4: ΠΡΑΓΜΑΤΙΣΤΗΣ
	8. ΒΑΘΜΟΣ ΛΑΝΘΑΣΜΕΝΗΣ ΑΝΤΙΛΗΨΗΣ	1: ΑΠΡΟΣΕΚΤΟΣ 2: ΠΙΘΑΝΩΣ ΑΠΡΟΣΕΚΤΟΣ 3: ΠΙΘΑΝΩΣ ΑΔΙΑΒΑΣΤΟΣ 4: ΑΔΙΑΒΑΣΤΟΣ
	9. ΔΙΑΓΡΑΜΜΑ ΠΕΡΙΠΤΩΣΗΣ ΧΡΗΣΗΣ 10. ΔΙΑΓΡΑΜΜΑ ΤΑΞΗΣ 11. ΔΙΑΓΡΑΜΜΑ ΚΑΤΑΣΤΑΣΗΣ 12. ΔΙΑΓΡΑΜΜΑ ΑΚΟΛΟΥΘΙΑΣ	1: ΛΑΝΘΑΣΜΕΝΗ ΑΝΤΙΛΗΨΗ ΣΕ ΚΑΠΟΙΑ ΥΠΟΕΝΟΤΗΤΑ 2: ΛΑΝΘΑΣΜΕΝΗ ΑΝΤΙΛΗΨΗ ΣΕ ΚΑΜΙΑ ΥΠΟΕΝΟΤΗΤΑ
ΚΟΙΝΩΝΙΚΗ	13. ΕΠΙΠΕΔΟ ΑΝΑΓΝΩΡΙΣΗΣ 14. ΕΠΙΠΕΔΟ ΣΥΝΕΙΣΦΟΡΑΣ 15. ΕΠΙΠΕΔΟ ΔΗΜΙΟΥΡΓΙΑΣ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ 16. ΚΟΙΝΩΝΙΚΗ ΣΥΜΜΕΤΟΧΗ	1: ΠΡΩΤΟ ΕΠΙΠΕΔΟ 2: ΔΕΥΤΕΡΟ ΕΠΙΠΕΔΟ 3: ΤΡΙΤΟ ΕΠΙΠΕΔΟ

Πίνακας 5.1

Μπορούν να οριστούν τυχαία αυτά τα 17 χαρακτηριστικά για 48 φοιτητές χρησιμοποιώντας την συνάρτηση "randint()" της βιβλιοθήκης "random" που εισάγεται με την

εντολή "import random" πριν από οποιαδήποτε άλλη εντολή. Για την προσωρινή αποθήκευσή τους μπορούν να χρησιμοποιηθούν λίστες και πλειάδες. Στον πίνακα 5.1 τα χαρακτηριστικά απαριθμούνται από το μηδέν (0) για να υπάρχει άμεση πρόσβαση μέσω λίστας ή πλειάδας, δεδομένου ότι η προσπέλαση τους ξεκινάει από τον δείκτη μηδέν (0), που βρίσκεται αποθηκευμένο το πρώτο στοιχείο. Παρακάτω παρουσιάζεται ο κώδικας υλοποίησης σε Python.

ΚΩΔΙΚΑΣ PYTHON: ΔΗΜΙΟΥΡΓΙΑ 48 ΦΟΙΤΗΤΩΝ ΜΕ 17 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

```
import random
# ΠΛΕΙΑΔΑ ΠΟΥ ΑΠΟΘΗΚΕΥΕΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΝΟΣ ΦΟΙΤΗΤΗ
characteristics = ()
# ΠΛΕΙΑΔΑ ΠΟΥ ΑΠΟΘΗΚΕΥΕΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΟΛΩΝ ΤΩΝ ΦΟΙΤΗΤΩΝ
tuples_characteristics = ()
# ΛΙΣΤΑ ΠΟΥ ΑΠΟΘΗΚΕΥΕΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΝΟΣ ΦΟΙΤΗΤΗ
l_characteristics = []
# ΛΙΣΤΑ ΠΟΥ ΑΠΟΘΗΚΕΥΕΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΟΛΩΝ ΤΩΝ ΦΟΙΤΗΤΩΝ
listes_characteristics = []
# ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΙΑ 48 ΦΟΙΤΗΤΕΣ
for j in range(48):
    # 0-5: ΑΚΑΔΗΜΑΪΚΕΣ ΔΕΞΙΟΤΗΤΕΣ
    for i in range(6):
        academic_characteristic = random.randint(1,5)
        l_characteristics.append(academic_characteristic)
    # 6: ΓΝΩΣΙΑΚΟ ΕΠΙΠΕΔΟ
    cognitive_bagde = random.randint(1,3)
    l_characteristics.append(cognitive_bagde)
    # 7: ΜΑΘΗΣΙΑΚΟΣ ΤΥΠΟΣ
    learning_style = random.randint(1,4)
    l_characteristics.append(learning_style)
    # 8: ΒΑΘΜΟΣ  ΛΑΝΘΑΣΜΕΝΗΣ ΑΝΤΙΛΗΨΗΣ
    misconception_degree = random.randint(1,4)
    l_characteristics.append(misconception_degree)
    # 9-12: ΛΑΝΘΑΣΜΕΝΗ ΑΝΤΙΛΗΨΗ ΣΕ ΚΑΠΟΙΑ ΥΠΟΕΝΟΤΗΤΑ ΤΟΥ ΑΝΤΙΣΤΟΙΧΟΥ ΔΙΑΓΡΑΜΜΑΤΟΣ
    for i in range(4):
        diagramm_misconception = random.randint(1,2)
        l_characteristics.append(diagramm_misconception)
    # 13-16: ΚΟΙΝΩΝΙΚΕΣ ΔΕΞΙΟΤΗΤΕΣ
    for i in range(4):
        social_skill = random.randint(1,3)
        l_characteristics.append(social_skill)
    characteristics = tuple(l_characteristics)
    # ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΦΟΙΤΗΤΗ ΣΤΗ ΛΙΣΤΑ ΠΟΥ ΘΑ ΠΕΡΙΕΧΕΙ ΟΛΟΥΣ ΤΟΥΣ ΦΟΙΤΗΤΕΣ
    listes_characteristics.append(characteristics)
    # ΚΑΘΑΡΙΖΕΙ ΓΙΑ ΝΑ ΑΠΟΘΗΚΕΥΤΟΥΝ  ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΕΠΟΜΕΝΟΥ ΦΟΙΤΗΤΗ
    l_characteristics = []
# ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΦΟΙΤΗΤΗ ΣΤΗ ΠΛΕΙΑΔΑ ΠΟΥ ΘΑ ΠΕΡΙΕΧΕΙ ΟΛΟΥΣ
```

```
ΤΟΥ ΦΟΙΤΗΤΕΣ
tuples_characteristics = tuple(listes_characteristics)
# ΕΜΦΑΝΙΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΑΝΑ ΦΟΙΤΗΤΗ
ap=0
for t in tuples_characteristics:
   print(ap, t, sep="\t", end="\n")
   ap=ap+1
```

Πίνακας 5.2

Για να δοθούν τιμές στα χαρακτηριστικά χρησιμοποιούνται οι ακόλουθες εντολές.

```
academic_characteristic = random.randint(1,5)          (1)
cognitive_bagde = random.randint(1,3)                  (2)
learning_style = random.randint(1,4)                   (3)
misconception_degree = random.randint(1,4)             (4)
diagramm_misconception = random.randint(1,2)           (5)
social_skill = random.randint(1,3)                     (6)
```

Σύμφωνα με τον πίνακα 5.1 τα χαρακτηριστικά που αφορούν την Ακαδημαϊκή διάσταση χωρίζονται σε δύο κατηγορίες. Η πρώτη περιλαμβάνει τα χαρακτηριστικά 0-5 που μπορούν να πάρουν τυχαίες ακέραιες τιμές από ένα ως πέντε και γι' αυτό χρησιμοποιείται η εντολή (1). Το χαρακτηριστικό 6 όμως μπορεί να πάρει τυχαίες ακέραιες τιμές από ένα ως τρία, αφού υπάρχουν μόνο τρία γνωστικά επίπεδα και γι' αυτό χρησιμοποιείται η εντολή (2). Αντίστοιχα για το μαθησιακό τύπο, τον βαθμό λανθασμένης αντίληψης, τον βαθμό λανθασμένης αντίληψης σε κάποια υποενότητα του διαγράμματος και τις κοινωνικές δεξιότητες, χρησιμοποιούνται οι εντολές (3), (4), (5), (6).

Για κάθε φοιτητή χρησιμοποιείται η λίστα "l_characteristics" για να προστίθεται ένα προς ένα τα χαρακτηριστικά με τη χρήση της συνάρτησης "append()", ενώ κάθε λίστα (l_characteristics) με τα χαρακτηριστικά του φοιτητή μετατρέπεται σε πλειάδα με την εντολή "characteristics = tuple(l_characteristics)", για να εξασφαλιστεί ότι τα χαρακτηριστικά του φοιτητή δεν θα μεταβληθούν, και προστίθεται στη λίστα "listes_characteristics", με την εντολή "listes_characteristics.append(characteristics)", ώστε να περιλαμβάνει τα χαρακτηριστικά όλων των φοιτητών. Τέλος η λίστα που θα περιέχει τα χαρακτηριστικά όλων των φοιτητών (listes_characteristics) θα μετατραπεί και αυτή σε πλειάδα (tuples_characteristics) με την εντολή "tuples_characteristics = tuple(listes_characteristics)", έτσι ώστε να εξασφαλιστεί ότι όλα τα δεδομένα δεν μπορούν να μεταβληθούν. Με την τελευταία επαναληπτική δομή "for" του παραπάνω κώδικα γίνεται προσπέλαση στον κάθε

ένα φοιτητή και εμφανίζονται οι τιμές χαρακτηριστικών του. Στην εικόνα 5.1 παρουσιάζεται το αποτέλεσμα μιας εκτέλεσης του παραπάνω κώδικα στο programiz.

Shell

```
0  (5, 2, 5, 1, 5, 3, 1, 3, 4, 2, 1, 2, 2, 3, 3, 2, 1)
1  (3, 5, 4, 4, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2)
2  (4, 4, 4, 1, 2, 4, 1, 2, 1, 1, 1, 1, 2, 3, 2, 2, 1)
3  (2, 5, 5, 5, 1, 2, 2, 3, 3, 2, 1, 1, 1, 1, 2, 2, 1)
4  (5, 4, 5, 2, 5, 3, 3, 4, 2, 2, 2, 2, 2, 2, 2, 3, 3)
5  (1, 3, 2, 3, 4, 4, 3, 1, 3, 2, 2, 2, 2, 2, 3, 2, 1)
6  (4, 4, 1, 2, 3, 4, 3, 2, 3, 2, 1, 2, 2, 2, 1, 3, 1)
7  (5, 1, 3, 2, 5, 1, 3, 3, 4, 1, 1, 1, 2, 1, 3, 3, 1)
8  (2, 3, 2, 2, 1, 1, 2, 3, 4, 1, 1, 2, 1, 2, 3, 3, 3)
9  (1, 1, 2, 3, 3, 1, 1, 3, 2, 1, 1, 2, 1, 3, 3, 2, 1)
10 (1, 1, 1, 2, 1, 2, 2, 3, 2, 2, 2, 2, 1, 3, 1, 2, 3)
11 (2, 3, 5, 5, 5, 3, 2, 3, 2, 2, 1, 2, 1, 3, 2, 3, 1)
12 (1, 3, 1, 3, 5, 1, 2, 1, 3, 1, 2, 1, 1, 1, 3, 1, 1)
13 (1, 5, 4, 2, 2, 5, 2, 2, 2, 1, 2, 2, 1, 1, 3, 1, 1)
14 (5, 5, 5, 2, 5, 3, 3, 3, 3, 1, 2, 1, 1, 1, 2, 3, 3)
15 (4, 5, 2, 1, 4, 4, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 3)
16 (2, 2, 4, 2, 1, 4, 2, 4, 3, 2, 1, 1, 1, 1, 1, 1, 3)
17 (5, 1, 1, 2, 3, 1, 3, 3, 2, 2, 2, 2, 2, 2, 3, 3, 1)
18 (3, 1, 1, 5, 4, 1, 2, 3, 3, 2, 1, 2, 2, 2, 1, 2, 2)
19 (3, 1, 1, 2, 5, 1, 1, 3, 1, 2, 1, 2, 2, 2, 1, 3, 1)
20 (3, 5, 5, 2, 5, 4, 3, 4, 1, 1, 2, 2, 2, 2, 2, 1, 3)
21 (5, 4, 3, 1, 3, 1, 1, 2, 2, 2, 1, 2, 2, 3, 2, 1, 1)
22 (3, 3, 2, 2, 1, 2, 1, 4, 4, 2, 1, 2, 2, 3, 1, 2, 3)
23 (2, 3, 4, 4, 4, 3, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2)
24 (3, 5, 5, 3, 2, 1, 3, 1, 4, 2, 1, 2, 2, 3, 1, 1, 3)
25 (2, 5, 5, 1, 2, 5, 1, 3, 4, 2, 1, 2, 2, 2, 3, 1, 1)
26 (3, 5, 1, 2, 1, 4, 2, 3, 1, 1, 1, 2, 1, 2, 2, 3, 1)
27 (4, 4, 4, 3, 5, 2, 3, 1, 3, 2, 1, 2, 2, 2, 1, 1, 1)
28 (4, 1, 4, 3, 2, 1, 2, 1, 4, 2, 2, 1, 2, 2, 2, 3, 1)
29 (3, 2, 1, 5, 3, 3, 3, 4, 4, 2, 2, 1, 2, 1, 1, 1, 2)
30 (4, 2, 4, 5, 5, 4, 1, 1, 4, 2, 2, 1, 1, 1, 1, 3, 3)
31 (1, 2, 1, 1, 1, 2, 1, 4, 2, 2, 2, 1, 2, 3, 3, 2, 1)
32 (4, 5, 5, 4, 5, 3, 3, 2, 4, 2, 1, 2, 1, 1, 1, 2, 2)
33 (1, 2, 1, 1, 5, 4, 2, 2, 1, 1, 2, 2, 1, 3, 2, 2, 1)
34 (4, 2, 2, 3, 5, 1, 1, 1, 1, 1, 2, 2, 2, 3, 2, 1, 3)
35 (1, 5, 5, 4, 4, 4, 2, 4, 3, 1, 2, 1, 2, 1, 1, 2, 1)
36 (1, 4, 3, 4, 5, 5, 2, 2, 2, 1, 2, 2, 1, 1, 3, 1, 3)
37 (2, 1, 5, 3, 3, 1, 2, 4, 4, 1, 2, 1, 1, 3, 3, 1, 2)
38 (2, 3, 4, 4, 5, 3, 3, 2, 2, 2, 1, 1, 1, 2, 1, 2, 3)
39 (1, 5, 3, 1, 5, 2, 2, 3, 3, 1, 2, 2, 2, 1, 1, 2, 2)
40 (4, 2, 4, 3, 2, 3, 1, 2, 1, 1, 1, 2, 2, 2, 3, 2, 1)
41 (3, 2, 5, 4, 5, 2, 1, 4, 3, 2, 1, 2, 1, 3, 1, 3, 1)
42 (3, 5, 3, 3, 1, 1, 2, 3, 4, 1, 2, 2, 1, 1, 3, 1, 3)
43 (3, 2, 3, 3, 3, 4, 3, 1, 4, 2, 1, 2, 1, 3, 1, 1, 1)
44 (2, 2, 1, 5, 2, 1, 1, 4, 4, 1, 1, 1, 2, 2, 1, 1, 2)
45 (3, 2, 3, 3, 5, 2, 2, 2, 3, 2, 1, 2, 2, 3, 1, 2, 1)
46 (1, 1, 3, 2, 5, 1, 1, 4, 4, 2, 2, 2, 1, 2, 3, 2, 2)
47 (1, 3, 3, 1, 1, 3, 1, 2, 4, 2, 1, 2, 2, 3, 2, 3, 3)
>
```

Εικόνα 5.1

5.2 Υλοποίηση χρωμοσώματος

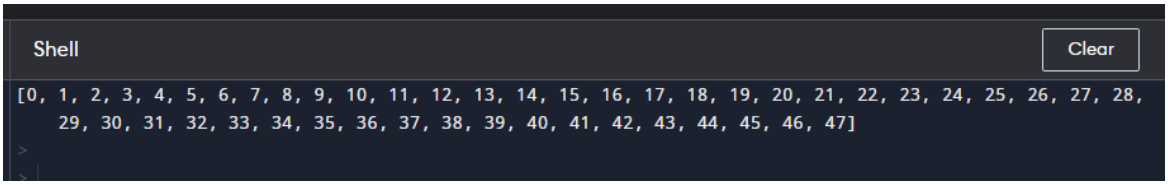
Όπως αναφέρθηκε στην ενότητα 4.3.1, επειδή κάθε ένας φοιτητής πρέπει να ανήκει σε μια μόνο ομάδα, για την αναπαράσταση του χρωμοσώματος θα χρησιμοποιηθεί η κωδικοποίηση μεταλλαγής (permutation encoding) και ως δομή αναπαράστασης θα χρησιμοποιηθεί λίστα της γλώσσας προγραμματισμού python. Αρχικά δεν χρησιμοποιείται πλειάδα, για να είναι δυνατόν, με βάση το αρχικό χρωμόσωμα που θα δημιουργηθεί, να δημιουργηθούν και τα υπόλοιπα, όπως θα δούμε στην επόμενη ενότητα. Ωστόσο βέβαια δεν χρειάζεται να χρησιμοποιηθεί καθόλου πλειάδα, δεδομένου ότι θα εξασφαλιστεί από τον κώδικα ότι θα συμπεριληφθούν όλοι οι φοιτητές σε κάθε χρωμόσωμα και θα εμφανίζεται ο κάθε φοιτητής μόνο μια φορά. Έτσι για να γίνει η αναπαράσταση του (αρχικού) χρωμοσώματος με την αρχική διάταξη (σειρά) των φοιτητών χρησιμοποιείται ο παρακάτω κώδικας.

ΚΩΔΙΚΑΣ PYTHON: ΥΛΟΠΟΙΗΣΗ ΑΡΧΙΚΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ

```
initial_chromosome = []
for i in range(48):
   initial_chromosome.append(i)
print(initial_chromosome)
```

Πίνακας 5.3

και το αποτέλεσμα στο programiz θα είναι αυτό που φαίνεται στην Εικόνα 5.2:



Εικόνα 5.2

Η αρίθμηση των φοιτητών ξεκινάει με id = 0, για να υπάρχει η δυνατότητα άμεσης πρόσβασης στα χαρακτηριστικά του που θα έχουν αποθηκευτεί στην πλειάδα tuples_characteristics.

5.3 Υλοποίηση αρχικοποίησης πληθυσμού

Για να γίνει η αρχικοποίηση του πληθυσμού θα δημιουργηθεί πρώτα το αρχικό χρωμόσωμα (initial_chromosome) όπως ακριβώς γίνεται στην ενότητα 5.2 και έπειτα τυχαία, με την χρήση της συνάρτησης "shuffle()" της βιβλιοθήκης "random", θα δημιουργηθούν και

τα υπόλοιπα χρωμοσώματα που θα αποτελέσουν τον αρχικό πληθυσμό. Ο παρακάτω κώδικας υλοποιεί την αρχικοποίηση του πληθυσμού, δημιουργώντας εκατό χρωμοσώματα στο σύνολο, πάνω στον οποίο θα εφαρμοστούν οι τελεστές του γενετικού αλγορίθμου (ελιτισμός, διασταύρωση και μετάλλαξη) για να δημιουργηθεί η επόμενη γενιά.

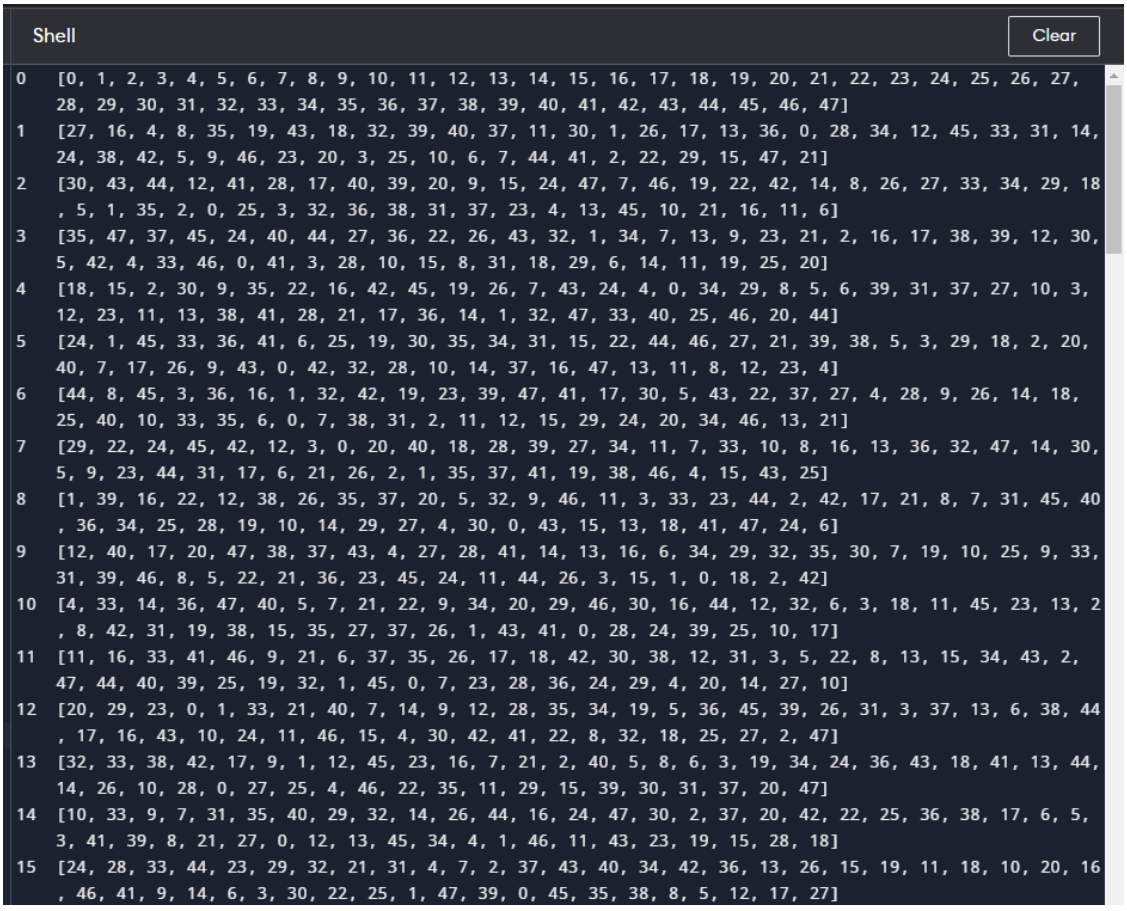
ΚΩΔΙΚΑΣ PYTHON: ΔΗΜΙΟΥΡΓΙΑ 100 ΤΥΧΑΙΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΓΙΑ ΤΗΝ ΑΡΧΙΚΟΠΟΙΗΣΗ ΠΛΥΘΗΣΜΟΥ

```
import random
initial_chromosome = []
l_chromosome = []
# ΑΝΑΓΝΩΡΙΣΤΙΚΟ ΦΟΙΤΗΤΗ ΑΠΟ 0 ΕΩΣ 47
for i in range(48):
    # ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΙΚΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ [0,1,…,47]
    initial_chromosome.append(i)
# ΚΡΑΤΑΕΙ ΑΝΤΙΓΡΑΦΟ ΤΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ
l_chromosome = initial_chromosome.copy()
# ΛΙΣΤΑ ΠΟΥ ΑΠΟΘΗΚΕΥΕΙ ΤΑ ΧΡΩΜΟΣΩΜΑΤΑ
listes_chromosomes = []
listes_chromosomes.append(l_chromosome)
# ΔΗΜΙΟΥΡΓΙΑ 99 ΧΡΩΜΟΣΩΜΑΤΟΝ
for i in range(99):
    # ΑΝΑΚΑΤΕΥΕΙ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΧΡΩΜΟΣΩΜΑ ΚΑΙ ΔΗΜΙΟΥΡΓΕΙ ΚΑΙΝΟΥΡΓΙΟ
    random.shuffle(initial_chromosome)
    # ΑΝΤΙΓΡΑΦΗ ΧΡΩΜΟΣΩΜΑΤΟΣ
    l_chromosome = initial_chromosome.copy()
    # ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΛΙΣΤΑ ΧΡΩΜΟΣΩΜΑΤΩΝ
    listes_chromosomes.append(l_chromosome)
# ΕΜΦΑΝΙΖΕΙ ΕΝΑ ΠΡΟΣ ΕΝΑ ΤΑ ΧΡΩΜΟΣΩΜΑΤΑ
ap=0
for chromosome in listes_chromosomes:
    print(ap, chromosome, sep="\t", end="\n")
    ap=ap+1
```

Πίνακας 5.4

Στον παραπάνω κώδικα χρησιμοποιούνται τρεις λίστες (initial_chromosome, l_chromosome, listes_chromosomes). Η λίστα "initial_chromosome" χρησιμοποιείται για να δημιουργηθεί, όπως έχει ήδη αναφερθεί το πρώτο χρωμόσωμα που περιέχει τους φοιτητές με τη σειρά (id=0, id=1,…, id=47). Επίσης χρησιμοποιείται για να δημιουργηθούν και τα υπόλοιπα 99 χρωμοσώματα με την εντολή "random.shuffle(initial_chromosome)" που ανακατεύει τα γονίδια του (αρχικού ή του προηγούμενου χρωμοσώματος) δημιουργώντας καινούργιο. Η λίστα "l_chromosome" χρησιμοποιείται για να αποθηκεύεται προσωρινά το κάθε ένα χρωμόσωμα, πριν ξαναχρησιμοποιηθεί συνάρτηση "shuffle()", με την εντολή "l_chromosome = initial_chromosome.copy()" και αποθηκεύεται στην τρίτη λίστα

(listes_chromosomes) με την εντολή "listes_chromosomes.append(l_chromosome)", που στο τέλος θα περιλαμβάνει όλα τα χρωμοσώματα μαζί. Επιπλέον ο κώδικας χρησιμοποιεί μια επαναληπτική δομή "for" για να γίνει η προσπέλαση και η εμφάνιση του καθενός χρωμοσώματος ένα προς ένα. Στην εικόνα 5.3 παρουσιάζονται μερικά χρωμοσώματα που ενδέχεται να δημιουργηθούν από την εκτέλεση του αλγορίθμου αρχικοποίησης πληθυσμού στο programiz.



Εικόνα 5.3

5.4 Υλοποίηση χαρακτηριστικών φοιτητών και αρχικοποίησης πληθυσμού με τη χρήση συναρτήσεων

Στις προηγούμενες ενότητες του κεφαλαίου χρησιμοποιήθηκε αυτόνομος κώδικας για να σχηματιστούν τα χαρακτηριστικά των φοιτητών και να γίνει η αρχικοποίηση του πληθυσμού. Σκοπός αυτής της ενότητας είναι να δημιουργηθούν αντίστοιχες συναρτήσεις οι οποίες θα μπορούν να χρησιμοποιηθούν κατάλληλα στο επόμενο κεφάλαιο για να πάρουμε αποτελέσματα από τη χρήση γενετικού αλγορίθμου που θα τις αξιοποιήσει.

Στον παρακάτω κώδικα ορίζεται μια συνάρτηση (learning_characteristics()) που σχηματίζει τα χαρακτηριστικά των φοιτητών και δέχεται ως παράμετρο εισόδου το πλήθος των φοιτητών (number_students), που θα είναι 48 αν η συνάρτηση κληθεί χωρίς όρισμα.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΦΟΙΤΗΤΩΝ ΜΕ 17 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

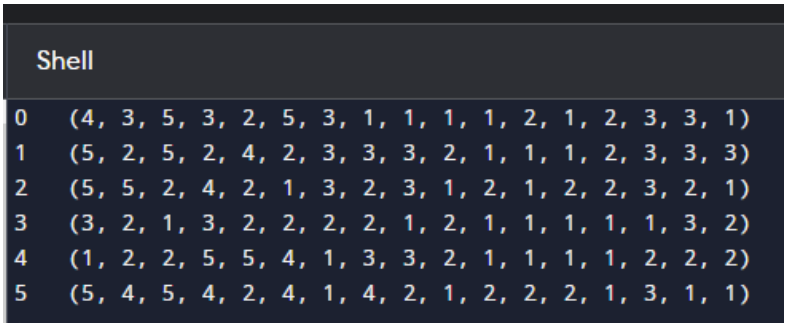
```
import random
def learning_characteristics(number_students=48):
    # ΚΑΘΟΛΙΚΗ ΜΕΤΑΒΛΗΤΗ ΜΕ ΤΟ ΠΛΗΘΟΣ ΤΩΝ ΦΟΙΤΗΤΩΝ
    global n_students
    characteristics = ()
    listes_characteristics = []
    #ΚΑΘΟΛΙΚΗ ΠΛΕΙΑΔΑ ΜΕ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΦΟΙΤΗΤΩΝ
    global tuples_characteristics
    tuples_characteristics = ()
    l_characteristics = []
    # ΜΠΛΟΚ ΠΟΥ ΕΛΕΓΧΕΙ ΑΝ ΤΟ ΠΛΗΘΟΣ ΤΩΝ ΦΟΙΤΗΤΩΝ ΕΙΝΑΙ ΑΚΕΡΑΙΟΣ ΑΡΙΘΜΟΣ
    try: number_students = int(number_students)
    except ValueError: number_students = 48
    # ΜΠΛΟΚ ΠΟΥ ΕΛΕΓΧΕΙ ΑΝ ΣΧΗΜΑΤΙΖΟΝΤΑΙ ΟΜΑΔΕΣ ΤΩΝ ΤΡΙΩΝ ΚΑΙ ΤΟ ΠΛΗΘΟΣ ΔΕΝ
ΕΙΝΑΙ ΠΟΛΥ ΜΕΓΑΛΟ
    if (number_students < 3 or number_students > 48 or number_students % 3 !=0):
number_students = 48
    n_students = number_students
    # ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΙΑ ΠΛΗΘΟΣ ΦΟΙΤΗΤΩΝ number_students
    for j in range(number_students):
        # 0-5: ΑΚΑΔΗΜΑΪΚΕΣ ΔΕΞΙΟΤΗΤΕΣ
        for i in range(6):
            academic_characteristic = random.randint(1,5)
            l_characteristics.append(academic_characteristic)
        # 6: ΓΝΩΣΙΑΚΟ ΕΠΙΠΕΔΟ
        cognitive_bagde = random.randint(1,3)
        l_characteristics.append(cognitive_bagde)
        # 7: ΜΑΘΗΣΙΑΚΟΣ ΤΥΠΟΣ
        learning_style = random.randint(1,4)
        l_characteristics.append(learning_style)
        # 8: ΒΑΘΜΟΣ  ΛΑΝΘΑΣΜΕΝΗΣ ΑΝΤΙΛΗΨΗΣ
        misconception_degree = random.randint(1,4)
        l_characteristics.append(misconception_degree)
        # 9-12: ΛΑΝΘΑΣΜΕΝΗ ΑΝΤΙΛΗΨΗ ΣΕ ΚΑΠΟΙΑ ΥΠΟΕΝΟΤΗΤΑ ΤΟΥ ΑΝΤΙΣΤΟΙΧΟΥ
ΔΙΑΓΡΑΜΜΑΤΟΣ
        for i in range(4):
            diagramm_misconception = random.randint(1,2)
            l_characteristics.append(diagramm_misconception)
        # 13-16: ΚΟΙΝΩΝΙΚΕΣ ΔΕΞΙΟΤΗΤΕΣ
        for i in range(4):
            social_skill = random.randint(1,3)
            l_characteristics.append(social_skill)
        characteristics = tuple(l_characteristics)
        # ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΦΟΙΤΗΤΗ ΣΤΗ ΛΙΣΤΑ ΠΟΥ ΘΑ ΠΕΡΙΕΧΕΙ
```

```
ΟΛΟΥΣ ΤΟΥΣ ΦΟΙΤΗΤΕΣ
      listes_characteristics.append(characteristics)
      # ΚΑΘΑΡΙΖΕΙ ΓΙΑ ΝΑ ΑΠΟΘΗΚΕΥΤΟΥΝ  ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΕΠΟΜΕΝΟΥ ΦΟΙΤΗΤΗ
      l_characteristics = []
   # ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΦΟΙΤΗΤΗ ΣΤΗ ΠΛΕΙΑΔΑ ΠΟΥ ΘΑ ΠΕΡΙΕΧΕΙ
ΟΛΟΥΣ ΤΟΥ ΦΟΙΤΗΤΕΣ
   tuples_characteristics = tuple(listes_characteristics)
   # ΕΜΦΑΝΙΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΑΝΑ ΦΟΙΤΗΤΗ
   ap=0
   for t in tuples_characteristics:
      print(ap, t, sep="\t", end="\n")
      ap=ap+1
```

Πίνακας 5.5

Ο παραπάνω κώδικας δεν διαφέρει πολύ από εκείνον της ενότητας 5.1. Η κύρια διαφορά του είναι ότι μπορεί να δημιουργήσει χαρακτηριστικά και για λιγότερους από 48 φοιτητές αρκεί να σχηματίζουν ομάδες των τριών. Σε περίπτωση που ως όρισμα δοθεί πλήθος φοιτητών που δεν σχηματίζει ούτε μια ομάδα ή δοθεί πλήθος μεγαλύτερο από 48 ή το πλήθος δεν διαιρείται με το τρία, ώστε όλες να είναι ομάδες τριών φοιτητών, τότε σύμφωνα με την εντολή "if (number_students < 3 or number_students > 48 or number_students % 3 !=0): number_students = 48" το πλήθος φοιτητών ορίζεται στους 48. Το ίδιο συμβαίνει και αν δοθεί ως όρισμα κάποια τιμή που δεν αποτελεί ακέραιος, βάση του μπλοκ "try: number_students = int(number_students) except ValueError: number_students = 48", όπου ο χρήστης μπορεί να δώσει τον αριθμό των φοιτητών (ακέραιος αριθμός) και σαν χαρακτήρα ή συμβολοσειρά μέσα μονά ή διπλά αυτάκια. Επιπλέον, η δομή επανάληψης τρέχει τόσες επαναλήψεις, όσες μας δίνει η μεταβλητή number_students για το πλήθος φοιτητών. Τέλος ορίζονται δύο καθολικές μεταβλητές (global), η n_students που αποθηκεύει το πλήθος των φοιτητών της τοπικής παραμέτρου number_students και η tuples_characteristics που εμπεριέχει τα χαρακτηριστικά των φοιτητών, που χρησιμοποιούνται για τον υπολογισμό συνάρτησης καταλληλόλητας ενός συγκεκριμένου χρωμοσώματος όπως θα δούμε στις επόμενες ενότητες 5.5 και 5.6.

Η κλήση της συνάρτησης learning_characteristics(6) ή learning_characteristics('6') ή learning_characteristics("6") θα μας δώσει τα 17 χαρακτηριστικά μάθησης για 6 φοιτητές. Παρακάτω παρουσιάζεται ένα αποτέλεσμα κλήσης της στο programiz.



Εικόνα 5.4

Αντίστοιχα ορίζεται συνάρτηση "initial_population()" με δύο παραμέτρους (number_students=48, number_chromosomes=100) για την αρχικοποίηση πληθυσμού χρωμοσωμάτων, που θα αποτελέσουν τη βάση για να δημιουργηθούν οι επόμενες γενεές, μέσω του γενετικού αλγορίθμου, οι οποίες θα μας φέρνουν όλο και πιο κοντά στη βέλτιστη λύση. Αν κληθεί η συνάρτηση χωρίς ορίσματα τότε θα δημιουργήσει αρχικό πληθυσμό με εκατό τυχαία χρωμοσώματα (εκτός από το αρχικό).

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΑΡΧΙΚΟΥ ΠΛΥΘΗΣΜΟΥ ΜΕ ΤΥΧΑΙΑ ΧΡΩΜΟΣΩΜΑΤΑ

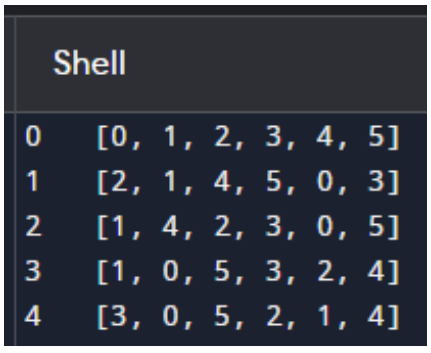
```
import random
def initial_population(number_students=48, number_chromosomes=100):
    initial_chromosome = []
    l_chromosome = []
    # ΚΑΘΟΛΙΚΗ ΛΙΣΤΑ ΧΡΩΜΟΣΩΜΑΤΩΝ
    global listes_chromosomes
    listes_chromosomes = []
    # ΕΛΕΓΧΟΙ ΠΛΗΘΟΥΣ ΦΟΙΤΗΤΩΝ
    try: number_students = int(number_students)
    except ValueError: number_students = 48
    if (number_students < 3 or number_students > 48 or number_students % 3 != 0):
number_students = 48
    # ΕΛΕΓΧΟΙ ΠΛΗΘΟΥΣ ΧΡΩΜΟΣΩΜΑΤΩΝ
    try: number_chromosomes = int(number_chromosomes)
    except ValueError: number_chromosomes = 100
    if (number_chromosomes <= 0 or number_chromosomes > 150):
number_chromosomes = 100
    # ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΙΚΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ
    for i in range(number_students):
        initial_chromosome.append(i)
    l_chromosome = initial_chromosome.copy()
    listes_chromosomes.append(l_chromosome)
    # ΔΗΜΙΟΥΡΓΙΑ number_chromosomes-1 ΑΚΟΜΗ ΧΡΩΜΟΣΩΜΑΤΩΝ
    for i in range(number_chromosomes-1):
        random.shuffle(initial_chromosome)
        l_chromosome = initial_chromosome.copy()
        listes_chromosomes.append(l_chromosome)
```

```
# ΕΜΦΑΝΙΖΕΙ ΤΑ ΧΡΩΜΟΣΩΜΑΤΑ ΕΝΑ ΠΡΟΣ ΕΝΑ
ap=0
for chromosome in listes_chromosomes:
   print(ap, chromosome, sep="\t", end="\n")
   ap=ap+1
```

Πίνακας 5.6

Ο παραπάνω κώδικας διαφέρει λίγο από εκείνον της ενότητας 5.3. Η βασική διαφορά του είναι ότι μπορεί να δημιουργήσει από 1 έως 150 χρωμοσώματα για έως και 48 φοιτητές αρκεί να σχηματίζουν ομάδες των τριών. Οι έλεγχοι που πραγματοποιούνται όσον αφορά το πλήθος των φοιτητών είναι ακριβώς ίδιοι με της συνάρτησης χαρακτηριστικών τους "learning_characteristics()", ενώ χρησιμοποιούνται αντίστοιχοι έλεγχοι και για το πλήθος των χρωμοσωμάτων. Δηλαδή, αν είναι αρνητικό ή μηδέν ή μεγαλύτερο από 150 ορίζεται στα 100 από την εντολή "if (number_chromosomes <= 0 or number_chromosomes > 150): number_chromosomes = 100", όπως επίσης και στην περίπτωση που δεν αποτελεί ακέραιο αριθμό κάτι που ελέγχεται στο μπλοκ "try: number_chromosomes = int(number_chromosomes) except ValueError: number_chromosomes = 100", όπου ο χρήστης μπορεί να δώσει τον αριθμό των χρωμοσωμάτων (ακέραιος αριθμός) και σαν χαρακτήρα ή συμβολοσειρά μέσα μονά ή διπλά αυτάκια. Τέλος ορίζεται ως καθολική μεταβλητή (global), η listes_chromosomes που περιλαμβάνει τη λίστα με τα χρωμοσώματα και εκτός από τον αρχικό πληθυσμό θα αξιοποιηθεί από τον γενετικό αλγόριθμο για την δημιουργία των επόμενων γενεών.

Η κλήση της συνάρτησης initial_population(6,5) ή initial_population('6','5') ή initial_population("6","5") θα μου δώσει 5 χρωμοσώματα για 6 φοιτητές. Παρακάτω παρουσιάζεται ένα αποτέλεσμα κλήσης της στο programiz.



Εικόνα 5.5

5.5 Υλοποίηση βοηθητικών συναρτήσεων για το υπολογισμό συνάρτησης καταλληλόλητας

Πρώτα θα υλοποιηθεί η συνάρτηση που υπολογίζει την ιδανική επανάληψη "ideal repetition(j)" ενός χαρακτηριστικού της ομάδας. Επειδή οι ομάδες που σχηματίζονται είναι τριών ατόμων η ιδανική επανάληψη για τα περισσότερα χαρακτηριστικά είναι 3 ($1^3 + 1^3 + 1^3$), αν ιδανικά δεν επαναληφθεί η τιμή του χαρακτηριστικού, κάτι που μπορεί να συμβεί δεδομένου ότι παίρνει τουλάχιστον τρεις διαφορετικές τιμές. Για την περίπτωση όμως λανθασμένης αντίληψης σε κάποιο διάγραμμα UML, τότε η ιδανική επανάληψη είναι 9 ($1^3 + 2^3$) καθώς στην καλύτερη περίπτωση αυτό το χαρακτηριστικό θα επαναληφθεί μια φόρα, δεδομένου ότι περιλαμβάνει μόνο δύο διαφορετικές τιμές (όπως φαίνεται στον πίνακα 5.1 για τα χαρακτηριστικά 9 ως 12). Παρακάτω παρουσιάζεται ο κώδικας της συνάρτησης υλοποίησης ιδανικής επανάληψης idealrepitition(j=0) χαρακτηριστικού j (από 0 ως 16).

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΙΔΑΝΙΚΗΣ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ

```
def idealrepitition(j=0):
   if (j >= 0 and j <= 16):
      id_rep = 0
      # ΜΟΝΟ ΔΥΟ ΔΙΑΦΟΡΕΤΙΚΕΣ ΤΙΜΕΣ ΓΙΑ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΠΟΥ ΑΦΟΡΟΥΝ ΤΑ
ΔΙΑΓΡΑΜΜΑΤΑ
      if (j == 9 or j == 10 or j == 11 or j == 12): id_rep = 9
      else: id_rep = 3
      return id_rep
   else: print("error out of boundaries")
```

Πίνακας 5.7

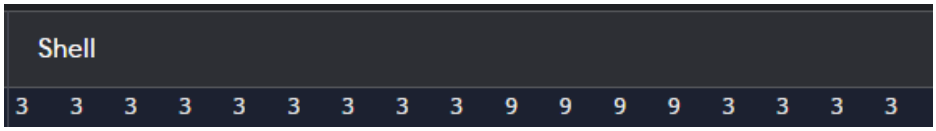
Για να πάρουμε αποτέλεσμα θα πρέπει να δοθεί δείκτης για χαρακτηριστικό που θα υπάρχει (0 ως 16), αλλιώς θα εμφανιστεί μήνυμα λάθους ("error out of boundaries"). Η συνάρτηση ορίζεται να επιστρέφει την τιμή της μεταβλητής id_rep στο όνομά της, κάτι το οποίο είναι ιδιαίτερα χρήσιμο για τον υπολογισμό συνάρτησης καταλληλόλητας, όπως θα δούμε στην επόμενη ενότητα. Συγκεκριμένα το αποτέλεσμα του παρακάτω κώδικα

ΚΩΔΙΚΑΣ PYTHON: ΕΜΦΑΝΙΣΗ ΙΔΑΝΙΚΗΣ ΕΠΑΝΑΛΗΨΗΣ ΤΩΝ 17 ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

```
for j in range(17):
   if(j != 16): print(idealrepitition(j), end="\t")
   else: print(idealrepitition(j), end="\n")
```

Πίνακας 5.8

στο programiz είναι:



Εικόνα 5.6

Η εντολή ελέγχου "if(j != 16): print(idealrepitition(j), end="\t") else: print(idealrepitition(j), end="\n")" βοηθάει να εμφανιστούν τα αποτελέσματα ιδανικής επανάληψης για κάθε ένα από τα 17 χαρακτηριστικά το ένα δίπλα στο άλλο.

Αντίστοιχα ορίζεται και η συνάρτηση επανάληψης χαρακτηριστικού (repitition), χρειάζεται όμως να ξέρουμε σε ποιο group γίνεται η αναφορά και την αντίστοιχη τιμή του χαρακτηριστικού j από την "tuples_characteristics" του χρωμοσώματος που είναι αποθηκευμένο στο "listes_chromosomes". Για να πάρουμε τιμές από τα χαρακτηριστικά των φοιτητών πρέπει να έχει οπωσδήποτε οριστεί και κληθεί η συνάρτηση "learning_characteristics()". Για να πάρουμε το αντίστοιχο χρωμόσωμα πρέπει να έχει οριστεί και κληθεί η συνάρτηση "initial_population()". Για να οριστούν τα group και ο δείκτης του πρώτου μέλους κάθε group ορίζονται οι δύο παρακάτω συναρτήσεις.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΕΙΣ ΔΗΜΙΟΥΡΓΙΑΣ ΟΜΑΔΩΝ ΚΑΙ ΔΕΙΚΤΩΝ ΠΡΩΤΟΥ ΜΕΛΟΥΣ ΚΑΘΕ ΟΜΑΔΑΣ

```
def create_groups():
    global groups
    groups = []
    for i in range (n_students // 3):
        groups.append(i)
    groups = tuple(groups)
    print (len(groups), groups, sep ="\t", end = "\n\n")

def create_indexes():
    global index
    index = []
    for g in groups:
        index.append(3*g)
    index = tuple(index)
    print(index, end = "\n\n")
```

Πίνακας 5.9

Η "create_groups ()" περιλαμβάνει μια καθολική μεταβλητή (global) "groups" (πλειάδα) που περιλαμβάνει τις ομάδες ανάλογα με το πλήθος των φοιτητών που έχει οριστεί στην συνάρτηση "learning_characteristics()" και έχει αποθηκευτεί στην καθολική μεταβλητή "n_students". Δεδομένου ότι οι ομάδες περιλαμβάνουν τρία άτομα, οι ομάδες ορίζονται από 0 μέχρι n_students // 3 (ακέραια διαίρεση με το τρία). Όταν οριστούν οι ομάδες, τότε μπορούν να οριστούν και οι δείκτες για το πρώτο μέλος της κάθε ομάδας με τη χρήσης της συνάρτησης "create_indexes()", όπου αντίστοιχα οι δείκτες είναι 0, 3,…, 3*g με το g να ανήκει στις ομάδες "groups", και οι οποίοι αποθηκεύονται στην καθολική (global) μεταβλητή "index" (πλειάδα). Εφόσον λοιπόν έχουν οριστεί και κληθεί οι προαναφερθείσες συναρτήσεις, τότε μπορεί να οριστεί και η συνάρτηση επανάληψης χαρακτηριστικού (repitition), όπως φαίνεται παρακάτω.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ

```
def repitition(ch, group=0, j=0):
    if (group in groups and j >= 0 and j <= 16):
        rep = 0
        i = index[group]
        k = ch[i]
        k1 = ch[i+1]
        k2 = ch[i+2]
        s1 = tuples_characteristics[k][j] == tuples_characteristics[k1][j]
        s2 = tuples_characteristics[k][j] == tuples_characteristics[k2][j]
        s3 = tuples_characteristics[k1][j] == tuples_characteristics[k2][j]
        # ΑΝ ΔΕΝ ΕΠΑΝΑΛΑΜΒΑΝΕΤΑΙ Η ΤΙΜΗ ΤΟΥ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
        rep = 3
        # ΑΝ ΕΙΝΑΙ ΙΔΙΑ Η ΤΙΜΗ ΤΟΥ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ ΚΑΙ ΓΙΑ ΤΟΥΣ ΤΡΕΙΣ ΦΟΙΤΗΤΕΣ
        if (s1 and s2): rep = 27
        # ΑΝ ΕΠΑΝΑΛΑΜΒΑΝΕΤΑΙ ΜΙΑ ΦΟΡΑ Η ΤΙΜΗ ΤΟΥ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
        elif (s1): rep = 9
        elif (s2): rep = 9
        elif (s3): rep = 9
        return rep
    else: print("error out of boundaries")
```

Πίνακας 5.10

Όπως βλέπουμε στον ορισμό της συνάρτησης χρειάζεται να εισαχθούν ως παράμετροι, το χρωμόσωμα (ch), η ομάδα (group) και το χαρακτηριστικό (j). Ο δείκτης i θα αναφέρεται στο πρώτο μέλος της ομάδας (group) και λαμβάνει αυτήν την τιμή από την εντολή "i = index[group]". Οι δείκτες που ακολουθούν i+1, i+2 θα αναφέρονται στα επόμενα δύο μέλη της ομάδας. Για να γίνει η αναφορά στα χαρακτηριστικά των τριών φοιτητών της ομάδας, ορίζονται οι δείκτες "k = ch[i]", "k1 = ch[i+1]", "k2 = ch[i+2]". Τέλος για να μπορεί να

υπολογιστεί η επανάληψη του χαρακτηριστικού (rep) έχουν οριστεί οι παρακάτω τρεις συνθήκες.

s1 = tuples_characteristics[k][j] == tuples_characteristics[k1][j] (1)

s2 = tuples_characteristics[k][j] == tuples_characteristics[k2][j] (2)

s3 = tuples_characteristics[k1][j] == tuples_characteristics[k2][j] (3)

Αν ισχύουν οι συνθήκες (1) και (2) τότε σίγουρα θα ισχύει και η συνθήκη (3) με το χαρακτηριστικό να είναι ίδιο (επαναλαμβάνεται) και για τους τρεις φοιτητές της ομάδας. Σε αυτή την περίπτωση δίνεται η μεγαλύτερη ποινή και η επανάληψη του χαρακτηριστικού ορίζεται ίση με 3**3 = 27, με την εντολή "rep = 27". Αν ισχύει μόνο μια από τις συνθήκες (1), (2), (3) που σημαίνει ότι το χαρακτηριστικό επαναλαμβάνεται μία μόνο φορά τότε δίνεται ελαφρύτερη ποινή και η επανάληψη του χαρακτηριστικού ορίζεται ίση 9 (1**3 + 2**3) με την εντολή "rep = 9". Τέλος αν δεν επαναλαμβάνεται κανένα χαρακτηριστικό, δεν υπάρχει ποινή και η επανάληψη του χαρακτηριστικού ορίζεται ίση με 3 (1**3 + 1**3 + 1**3) με την εντολή "rep = 3". Η συνάρτηση επιστρέφει την τιμή της μεταβλητής rep στο όνομά της, κάτι το οποίο είναι ιδιαίτερα χρήσιμο για τον υπολογισμό της συνάρτησης καταλληλόλητας, όπως θα δούμε στην επόμενη ενότητα. Με την παρακάτω τριπλή επαναληπτική δομή "for" εμφανίζονται τα αποτελέσματα της συνάρτησης επανάληψης (repitition(c,g,j)) κάθε χαρακτηριστικού (j) για κάθε ομάδα (g) φοιτητών καθενός χρωμοσώματος (c) που υπάρχει στη λίστα χρωμοσωμάτων "listes_chromosomes".

ΚΩΔΙΚΑΣ PYTHON: ΕΜΦΑΝΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΠΑΝΑΛΗΨΗΣ 17 ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΑΝΑ ΧΡΩΜΟΣΩΜΑ ΚΑΙ ΑΝΑ ΟΜΑΔΑ

```
ap_chromosomes=0
for c in listes_chromosomes:
    for g in groups:
        print("chromosome", ap_chromosomes,sep="\t", end="\t")
        print("group", g, sep="\t", end="\t\t")
        for j in range (17):
            if(j != 16): print(repitition(c, g, j), end="\t")
            else: print(repitition(c, g, j), end="\n")
    ap_chromosomes = ap_chromosomes + 1
```

Πίνακας 5.11

Στην Εικόνα 5.7 βλέπουμε εποπτικά στο programiz τις κλήσεις των συναρτήσεων:

learning_characteristics(6)

print("/n") (για να αφήσει κενή γραμμή να το ξεχωρίζει από τα χρωμοσώματα)

initial_population(6,5)

print("/n") (για να αφήσει κενή γραμμή να το ξεχωρίζει από τις ομάδες)

create_groups()

create_indexes()

Στη συνέχεια, καλώντας τον παραπάνω κώδικα με την τριπλή επαναληπτική δομή "for", που περιλαμβάνει την συνάρτηση "repitition(c,g,j)", θα εμφανίστουν στο programiz τα αποτελέσματα της Εικόνας 5.8.

```
0   (2, 2, 4, 3, 2, 3, 2, 1, 1, 2, 2, 2, 2, 2, 1, 3, 3)
1   (4, 3, 5, 1, 3, 3, 1, 4, 4, 2, 1, 1, 1, 1, 3, 1, 2)
2   (3, 1, 3, 3, 2, 2, 3, 4, 3, 1, 2, 2, 2, 1, 2, 3, 1)
3   (3, 2, 4, 4, 1, 3, 3, 1, 3, 2, 1, 1, 1, 1, 3, 2, 1)
4   (5, 5, 3, 5, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1, 1, 1, 3)
5   (1, 2, 2, 1, 1, 2, 1, 3, 4, 1, 2, 2, 2, 3, 2, 3, 3)


0   [0, 1, 2, 3, 4, 5]
1   [2, 4, 0, 1, 3, 5]
2   [3, 0, 4, 1, 2, 5]
3   [5, 1, 2, 0, 4, 3]
4   [2, 1, 3, 5, 4, 0]


2   (0, 1)

(0, 3)
```

Εικόνα 5.7

```
chromosome  0   group   0       3   3   3   9   9   9   3   9   3   9   9   9   9   9   3   9   3
chromosome  0   group   1       3   9   3   3   27  9   9   9   3   9   9   9   9   9   3   3   9
chromosome  1   group   0       3   3   9   9   9   9   3   3   9   9   9   9   27  9   9   9   9
chromosome  1   group   1       3   9   3   9   9   9   9   3   9   9   9   9   9   9   9   3   3
chromosome  2   group   0       3   9   9   3   9   27  3   9   9   9   9   9   9   9   9   3   9
chromosome  2   group   1       3   3   3   9   3   9   9   9   9   9   9   9   9   9   9   9   3
chromosome  3   group   0       3   3   3   9   3   9   9   9   9   9   9   9   9   9   9   9   3
chromosome  3   group   1       3   9   9   3   9   27  3   9   9   9   9   9   9   9   9   3   9
chromosome  4   group   0       9   3   3   3   3   9   9   9   9   9   9   9   9   27  9   3   9
chromosome  4   group   1       3   9   3   3   9   9   9   9   9   9   9   9   27  3   9   9   27
```

Εικόνα 5.8

Κάθε γραμμή αναφέρεται σε μια ομάδα του χρωμοσώματος. Εφόσον κάθε χρωμόσωμα περιλαμβάνει 6 φοιτητες, θα σχηματίζονται σε αυτό δύο ομάδες συνολικά. Άρα ανά δύο οι γραμμές, από τα αποτελέσματα της συνάρτησης "repitition(c,g,j)", αναφέρονται σε ένα χρωμόσωμα. Συνολικά για τα 6 χρωμοσώματά, που έχουν δημιουργηθεί εμφανίζονται 12

γραμμές αποτελεσμάτων που περιέχουν το χρωμόσωμα, την ομάδα και 17 στήλες με την τιμή συνάρτησης επανάληψης "repitition(c,g,j)" κάθε ενός από τα 17 χαρακτηριστικά.

5.6 Υλοποίηση συνάρτησης καταλληλόλητας

Δεδομένου ότι από την ενότητα 5.5 έχουν οριστεί οι συναρτήσεις υπολογισμού επανάληψης χαρακτηριστικού repitition(c,g,j) και ιδανικής επανάληψης χαρακτηριστικού idealrepitition(j), μπορεί να οριστεί και η συνάρτηση καταλληλόλητας ομάδας ως ακολούθως:

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΧΡΩΜΟΣΩΜΑΤΟΣ

```
def fitness_function(c):
    fitness = 0
    for g in groups:
        for j in range(17):
            fitness = fitness + (repitition(c, g, j)-idealrepitition(j))
    if(fitness != 0): return 1/fitness
    else: return 0.17
```

Πίνακας 5.12

Η συνάρτηση καταλληλόλητας παίρνει ως ορίσματα το χρωμόσωμα (c). Επειδή χρειάζεται να υπολογιστεί για όλα τα χαρακτηριστικά, και για όλες τις ομάδες του χρωμοσώματος, δεν χρειάζεται όρισμα που θα δηλώνει το κάθε ένα χαρακτηριστικό ή την ομάδα, αλλά μια διπλή εμφωλευμένη επαναληπτική δομή (for g in groups: for j in range(17): fitness = fitness + (repitition(c,g,j)-idealrepitition(j))), που θα αθροίζει τις τιμές της διαφοράς repitition(c,g,j)-idealrepitition(j), για όλα τα χαρακτηριστικά (j) και τις ομάδες (g) του χρωμοσώματος (c). Επίσης, χρησιμοποιείται η αντιστροφή της τιμής της συνάρτησης καταλληλόλητας κατά την επιστροφή της τιμής της συνάρτησης καταλληλόλητας στο όνομα συνάρτησης (return 1/fitness) και αυτό γιατί ο γενετικός αλγόριθμος που θα υλοποιηθεί στο επόμενο κεφάλαιο, προσπαθεί να μεγιστοποιήσει τη συνάρτηση καταλληλόλητας, και όχι να την ελαχιστοποιήσει, όπως δηλαδή απαιτείται σε προβλήματα με ετερογενής ομαδοποίηση. Αντιστρέφοντάς την λοιπόν, επιτυγχάνεται το επιθυμητό αποτέλεσμα, αφού κατά αυτόν το τρόπο η μεγιστοποίηση της αντίστροφής συνάρτησης καταλληλόλητας οδηγεί στην ελαχιστοποίηση της συνάρτησης καταλληλόλητας. Οι δύο ακόλουθες επαναληπτικές δομές τυπώνουν όλες τις τιμές της συνάρτησης επανάληψης χαρακτηριστικού και της συνάρτησης ιδανικής επανάληψης χαρακτηριστικού:

ΚΩΔΙΚΑΣ PYTHON: ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΔΟΜΕΣ ΠΟΥ ΕΜΦΑΝΙΖΟΥΝ ΤΙΣ ΤΙΜΕΣ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ ΚΑΙ ΙΔΑΝΙΚΗΣ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ

```
ap_chromosomes=0
for c in listes_chromosomes:
      for g in groups:
         print("chromosome", ap_chromosomes, sep="\t", end="\t")
         print("group", g, sep="\t", end="\t\t")
         for j in range (17):
            if(j != 16): print(repitition(c, g, j), end="\t")
            else: print(repitition(c, g, j), end="\n")
      ap_chromosomes = ap_chromosomes + 1
print("/n")
print("idealrepitition", end="\t\t\t\t\t")
for j in range(17):
   if(j != 16): print(idealrepitition(j), end="\t")
   else: print(idealrepitition(j), end="\n")
```

Πίνακας 5.13

Αν αυτές οι επαναληπτικές δομές δώσουν τα ακόλουθα αποτελέσματα (Εικόνα 5.9) στο programiz:

```
chromosome  0   group   0       3   3   9   3   9   3   9   9   9   9   9   27  9   9   9   9   27
chromosome  0   group   1       9   27  9   3   3   27  3   27  3   9   9   9   27  9   9   9   9
chromosome  1   group   0       3   3   3   3   9   3   27  9   3   9   27  9   9   9   9   3   9
chromosome  1   group   1       9   27  3   3   3   9   9   9   3   9   9   9   27  9   9   9   9
chromosome  2   group   0       3   3   3   3   9   3   27  9   3   9   27  9   9   9   9   3   9
chromosome  2   group   1       9   27  3   3   3   9   9   9   3   9   9   9   27  9   9   9   9
chromosome  3   group   0       9   9   9   3   9   3   9   3   9   9   9   27  27  9   27  9   27
chromosome  3   group   1       3   9   9   9   3   9   9   9   3   9   9   9   9   9   3   3   9
chromosome  4   group   0       9   9   9   3   9   3   9   3   9   9   9   27  27  9   27  9   27
chromosome  4   group   1       3   9   9   9   3   9   9   9   3   9   9   9   9   9   3   3   9


idealrepitition             3   3   3   3   3   3   3   3   3   9   9   9   9   3   3   3   3
```

Εικόνα 5.9

τότε η συνάρτηση καταλληλόλητας, με τη χρήση της ακόλουθης επαναληπτικής δομής:

ΚΩΔΙΚΑΣ PYTHON: ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΟΜΗ ΠΟΥ ΕΜΦΑΝΙΖΕΙ ΤΗΝ ΤΙΜΗ ΣΥΝΑΡΤΗΣΗΣ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΓΙΑ ΚΑΘΕ ΧΡΩΜΟΣΩΜΑ

```
print("/n")
ap_chromosomes=0
```

```
for c in listes_chromosomes:
   print("chromosome", ap_chromosomes, sep="\t", end="\t")
   print("{0:.16f}".format(fitness_function(c)), end="\n")
   ap_chromosomes = ap_chromosomes + 1
```

Πίνακας 5.14

θα δώσει τα παρακάτω αποτελέσματα (Εικόνα 5.10) στο programiz:

```
chromosome  0   0.0046296296296296
chromosome  1   0.0061728395061728
chromosome  2   0.0061728395061728
chromosome  3   0.0055555555555556
chromosome  4   0.0055555555555556
```

Εικόνα 5.10

Η δεύτερη στήλη περιλαμβάνει τα αποτελέσματα συνάρτησης καταλληλόλητας κάθε χρωμοσώματος (με ακρίβεια 16 δεκαδικών ψηφίων).

Όλα τα παραπάνω αποτελέσματα προέρχονται από το γεγονός ότι έχουν δημιουργηθεί 6 φοιτητές (με τα 17 χαρακτηριστικά τους) από την κλήση της συνάρτησης "learning_characteristics(6)" (print("/n") για να αφήσει κενή γραμμή να το ξεχωρίζει από τα χρωμοσώματα), 5 χρωμοσώματα από την κλήση της συνάρτησης "initial_population(6,5)" (print("/n") για να αφήσει κενή γραμμή να το ξεχωρίζει από τις ομάδες), οι ομάδες από την κλήση της συνάρτησης "create_groups()" και οι δείκτες πρώτου μέλους κάθε ομάδας από κλήση της συνάρτησης "create_indexes()". Επομένως καλούμε τις προαναφερόμενες συναρτήσεις και εντολές με την κάτωθι σειρά:

learning_characteristics(6)

print("/n") (για να αφήσει κενή γραμμή να το ξεχωρίζει από τα χρωμοσώματα)

initial_population(6,5)

print("/n") (για να αφήσει κενή γραμμή να το ξεχωρίζει από τις ομάδες)

create_groups()

create_indexes()

και τα αντίστοιχα αποτελέσματα τους στο programiz παρουσιάζονται στην Εικόνα 5.11:

```
0   (3, 5, 5, 4, 2, 2, 3, 2, 1, 2, 2, 2, 2, 1, 1, 1, 1)
1   (2, 2, 2, 1, 2, 1, 3, 2, 3, 1, 2, 2, 1, 1, 3, 2, 1)
2   (4, 3, 5, 5, 5, 4, 2, 4, 1, 1, 1, 2, 2, 3, 1, 1, 1)
3   (3, 3, 4, 1, 1, 3, 1, 3, 4, 2, 1, 1, 2, 1, 2, 1, 3)
4   (3, 3, 1, 2, 2, 3, 2, 3, 3, 1, 2, 2, 2, 3, 1, 3, 1)
5   (4, 3, 4, 3, 5, 3, 3, 3, 2, 2, 2, 1, 2, 3, 1, 3, 3)


0   [0, 1, 2, 3, 4, 5]
1   [1, 5, 0, 2, 4, 3]
2   [5, 0, 1, 2, 4, 3]
3   [4, 2, 0, 3, 5, 1]
4   [0, 2, 4, 5, 3, 1]


2   (0, 1)

(0, 3)
```

Εικόνα 5.11

Κεφάλαιο 6

6. Υλοποίηση δημιουργίας νέας γενιάς του γενετικού αλγορίθμου ομαδοποίησης με τη χρήση της γλώσσας Python

Σε αυτό το κεφάλαιο γίνεται η υλοποίηση των συναρτήσεων του 7^{ου} βήματος του γενετικού αλγορίθμου ομαδοποίησης (δημιουργία νέας γενιάς), όπως αναφέρεται στην ενότητα 4.3 του κεφαλαίου 4, με τη χρήση γλώσσας προγραμματισμού Python 3.

Στην ενότητα 6.1 υλοποιείται η επιλογή βέλτιστων χρωμοσωμάτων, τα οποία εισάγονται στη νέα γενιά με την μέθοδο του ελιτισμού, ενώ η επιλογή των δύο γονέων, γίνεται με την μέθοδο του τουρνουά και υλοποιείται στην ενότητα 6.2. Στην ενότητα 6.3 υλοποιείται η διασταύρωση ενός σημείου ανά ομάδα και η τροποποιημένη διασταύρωση δύο σημείων, όπου δημιουργούνται τέσσερις απόγονοι, δύο από κάθε διασταύρωση. Ακολούθως, στην ενότητα 6.4 υλοποιείται εκ νέου τουρνουά μεταξύ των δύο γονέων και των τεσσάρων απογόνων, όπως αυτοί έχουν δημιουργηθεί στην ενότητα 6.3 και επιλέγονται τα δύο καταλληλότερα χρωμοσώματα.

Για την δημιουργία του νέου πληθυσμού, η συνάρτηση, που υλοποιείται στην ενότητα 6.5 επιλέγει επαναληπτικά δύο γονείς με την μέθοδο τουρνουά, που έχει υλοποιηθεί στην ενότητα 6.2 και σε περίπτωση που δεν εφαρμόζεται διασταύρωση, εάν αυτοί οι δύο γονείς δεν ανήκουν στο νέο πληθυσμό, προσθέτονται σε αυτόν. Διαφορετικά εφαρμόζονται στους δύο γονείς οι δύο τελεστές διασταύρωσης, που έχουν υλοποιηθεί στην ενότητα 6.3, και παράγονται τέσσερις απόγονοι. Στη συνέχεια εφαρμόζεται τουρνουά μεταξύ των δύο γονέων και των τεσσάρων απογόνων από την συνάρτηση που έχει υλοποιηθεί στην ενότητα 6.4 και επιλέγονται τα δύο καταλληλότερα χρωμοσώματα, όπως έχει αναφερθεί παραπάνω. Εάν οι δύο νικητές του τουρνουά δεν ανήκουν στο νέο πληθυσμό, εισάγονται σε αυτόν. Η επανάληψη τερματίζει όταν ο πληθυσμός της νέας γενιάς γίνει ίσος με τον πληθυσμό την προηγούμενης γενιάς.

Τέλος στην ενότητα 6.6 υλοποιείται η μέθοδος της μετάλλαξης ανταλλαγής σε κάποιο ή κάποια χρωμοσώματα της νέας γενιάς και στην περίπτωση που τα μεταλλαγμένα χρωμοσώματα έχουν καλύτερη τιμή συνάρτησης καταλληλότητας από αυτά τα χρωμοσώματα από τα οποία έχουν προέλθει, τα αντικαθιστούν.

6.1 Ελιτισμός

Στον πίνακα 6.1 παρουσιάζεται η συνάρτηση υλοποίησης ελιτισμού "elitism()". Στη συνάρτηση αυτή ορίζονται δύο μεταβλητές, η "new_population" και η "keep". Η "new_population" είναι καθολική (global) μεταβλητή και ουσιαστικά είναι μία λίστα στην

οποία αποθηκεύεται προσωρινά η κάθε νέα γενιά. Η "keep" περιέχει το πλήθος των καταλληλότερων χρωμοσωμάτων που θα μεταβούν από την τρέχουσα γενιά στην επόμενη, χωρίς να μεταβληθούν και ορίζεται ως το 10% του πληθυσμού της τρέχουσας γενιάς με την εντολή "if (population >= 10): keep = int(population * 0.1)", εφόσον υπάρχουν τουλάχιστον 10 χρωμοσώματα, με ελάχιστη δηλαδή τιμή το 1.

Ακολούθως γίνεται αξιολόγηση των χρωμοσωμάτων της τρέχουσας γενιάς και αποθηκεύονται οι τιμές της συνάρτησης καταλληλότητας στην λίστα "solution_fitness" με την εντολή "for c in listes_chromosomes: solution_fitness.append(fitness_function(c))". Στη συνέχεια με την επαναληπτική δομή "for i in range(keep):" υπολογίζεται κάθε ένα από τα "keep" καταλληλότερα χρωμοσώματα με την εντολή "if s == max(solution_fitness):" και αποθηκεύονται στην λίστα "new_population" με την εντολή "new_population .append(listes_chromosomes[count])". Με την εντολή "solution_fitness[count] = -1" εξασφαλίζεται από τον κώδικα ότι δεν θα επιλεγεί ξανά το ίδιο χρωμόσωμα, αφού ορίζεται η τιμή της συνάρτησης καταλληλότητας του χρωμοσώματος -1 που είναι αρνητική, δηλαδή σίγουρα μικρότερη από τις τιμές συνάρτησης καταλληλότητας των υπολοίπων χρωμοσωμάτων που δεν έχουν επιλεγεί. Τέλος, παρουσιάζονται τα αποτελέσματα της εφαρμογής συνάρτησης του ελιτισμού με την επαναληπτική δομή "for ap, c in enumerate(new_population): print("chromosome ", ap, c, fitness_function(c), sep = "\t", end ="\n\n")".

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΕΦΑΡΜΟΓΗΣ ΕΛΙΤΙΣΜΟΥ

```
def elitism():
  # ΟΡΙΣΜΟΣ ΚΑΘΟΛΙΚΗΣ ΜΕΤΑΒΛΗΤΗΣ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΝΕΑΣ ΓΕΝΙΑΣ
  global new_population
  new_population = []
  # ΠΛΗΘΟΣ ΚΑΛΥΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΠΟΥ ΘΑ ΜΕΤΑΒΟΥΝ ΣΤΗΝ ΕΠΟΜΕΝΗ ΓΕΝΙΑ
  keep = 1
  solution_fitness = []
  population = len(listes_chromosomes)
  if (population >= 10): keep = int(population * 0.1)
  # ΑΞΙΟΛΟΓΗΣΗ ΧΡΩΜΟΣΩΜΑΤΩΝ ΤΡΕΧΟΝΤΟΣ ΠΛΥΘΗΣΜΟΥ
  for c in listes_chromosomes:
    solution_fitness.append(fitness_function(c))
  for ap, f in enumerate(solution_fitness):
    print("chromosome ", ap, listes_chromosomes[ap], f, sep="\t",end="\n")
  # ΠΡΟΣΘΗΚΗ ΚΑΛΥΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΣΤΗΝ ΕΠΟΜΕΝΗ ΓΕΝΙΑ
  for i in range(keep):
    for count, s in enumerate(solution_fitness):
      if s == max(solution_fitness):
        new_population.append(listes_chromosomes[count])
```

```
            solution_fitness[count] = -1
            break
    # ΕΜΦΑΝΙΣΗ ΚΑΛΥΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΠΟΥ ΕΧΟΥΝ ΠΡΟΣΤΕΘΕΙ ΣΤΗΝ ΕΠΟΜΕΝΗ
ΓΕΝΙΑ
    print("\n")
    print("elitism chromosomes", end="\n")
    for ap, c in enumerate(new_population):
        print("chromosome ", ap, c, fitness_function(c), sep = "\t", end ="\n\n")
```

Πίνακας 6.1

Όπως φαίνεται από τον πίνακα 6.2, για να εφαρμοστεί ο ελιτισμός, ορίζονται πρώτα 6 φοιτητές με τα χαρακτηριστικά τους (συνολικά 17, βλ. κεφ. 4 & 5), από την συνάρτηση "learning_characteristics(6)" και ο αρχικός πληθυσμός 5 χρωμοσωμάτων με την συνάρτηση "initial_population(6,5)". Επιπλέον ορίζονται οι ομάδες και οι δείκτες πρώτου μέλους κάθε ομάδας με τις συναρτήσεις "create_groups()" και "create_indexes()". Αυτές οι συναρτήσεις έχουν υλοποιηθεί στο προηγούμενο κεφάλαιο.

ΚΩΔΙΚΑΣ PYTHON: ΕΜΦΑΝΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΛΙΤΙΣΜΟΥ

```
learning_characteristics(6)
print("\n")
initial_population(6,5)
print("\n")
create_groups()
create_indexes()
elitism()
```

Πίνακας 6.2

Στην εικόνα 6.1 παρουσιάζονται τα αποτελέσματα εκτέλεσης του παραπάνω κώδικα.

```
Shell
0   (4, 1, 4, 2, 2, 4, 1, 3, 1, 2, 1, 2, 1, 2, 2, 1, 3)
1   (4, 1, 2, 5, 2, 5, 3, 1, 4, 1, 1, 2, 1, 3, 2, 2, 2)
2   (3, 1, 4, 3, 1, 3, 1, 1, 2, 1, 2, 1, 1, 2, 2, 3, 1)
3   (1, 2, 1, 4, 5, 3, 1, 1, 1, 1, 2, 1, 2, 1, 1, 2, 2)
4   (5, 4, 5, 4, 5, 5, 1, 4, 1, 2, 1, 2, 2, 2, 2, 3, 3)
5   (3, 2, 3, 3, 1, 2, 2, 3, 4, 1, 2, 1, 1, 2, 1, 3, 3)


0   [0, 1, 2, 3, 4, 5]
1   [2, 4, 3, 1, 0, 5]
2   [2, 5, 3, 1, 4, 0]
3   [3, 0, 2, 4, 1, 5]
4   [0, 1, 5, 3, 2, 4]


2   (0, 1)

(0, 3)

chromosome  0   [0, 1, 2, 3, 4, 5]  0.00641025641025641
chromosome  1   [2, 4, 3, 1, 0, 5]  0.007246376811594203
chromosome  2   [2, 5, 3, 1, 4, 0]  0.0045045045045045045
chromosome  3   [3, 0, 2, 4, 1, 5]  0.00980392156862745
chromosome  4   [0, 1, 5, 3, 2, 4]  0.007246376811594203


elitism chromosomes
chromosome  0   [3, 0, 2, 4, 1, 5]  0.00980392156862745
```

Εικόνα 6.1

6.2 Επιλογή γονέων με τουρνουά

Στον πίνακα 6.3 παρουσιάζεται η συνάρτηση επιλογής γονέων με τουρνουά "def tournament(k = 10):", η οποία δέχεται ως όρισμα το πλήθος k τυχαίων χρωμοσωμάτων. Τα χρωμοσώματα αυτά θα επιλεγούν τυχαία για να συμμετάσχουν στο τουρνουά. Ο νικητής του τουρνουά μπορεί να αποτελέσει έναν από τους δύο γονείς. Εάν δεν δοθεί όρισμα στην

συνάρτηση, ορίζεται ως πλήθος τυχαίων χρωμοσωμάτων ο αριθμός 10, ο οποίος είναι και ο μέγιστος όπως φαίνεται από την εντολή "if (k <= 1) or (k > 10): k = 10". Το ίδιο ισχύει και στην περίπτωση που ο χρήστης δώσει στην μεταβλητή k τιμή που δεν είναι ακέραιος, σύμφωνα με την εντολή "try: k = int(k) except ValueError: k = 10".

Η τυχαία επιλογή των χρωμοσωμάτων γίνεται με τυχαία επιλογή δεικτών της λίστας των χρωμοσωμάτων "listes_chromosomes", με την βοήθεια της επαναληπτικής δομής "for i in range (k): dt = random.randint(0, population-1)". Για να μην επιλεγεί ο ίδιος δείκτης και συνεπώς το ίδιο χρωμόσωμα, χρησιμοποιείται η εντολή "if previous_dt: while (dt in previous_dt): dt = random.randint(0, population-1)". Στην λίστα "previous_dt" αποθηκεύονται οι δείκτες των χρωμοσωμάτων που έχουν επιλεγεί με την εντολή "previous_dt.append(dt)". Στην λίστα "tournament_chromosomes" αποθηκεύονται τα χρωμοσώματα που θα συμμετάσχουν στο τουρνουά με την εντολή "tournament _chromosomes.append(listes_chromosomes[dt])", ενώ στην λίστα "tournament_fitness" αποθηκεύονται οι αντίστοιχες τιμές συνάρτησης καταλληλότητας των επιλεγμένων χρωμοσωμάτων με την εντολή "tournament_fitness.append(solution_fitness[dt])".

Για την εύρεση του νικητή του τουρνουά χρησιμοποιείται εκ νέου η επαναληπτική δομή "for i in range (k): if tournament_fitness[i] == max(tournament_fitness):". Το χρωμόσωμα που αποτελεί το νικητή του τουρνουά αποθηκεύεται στην μεταβλητή "winner", η οποία ουσιαστικά είναι και αυτή μία λίστα, με την εντολή "winner = tournament_chromosomes[i]" και εμφανίζεται με την εντολή "print("winner of tournament is chromosome: ", i, winner, tournament_fitness[i], sep = "\t", end="\n\n")". Η επαναληπτική δομή for σπάει με την εντολή break αμέσως μόλις βρεθεί το πρώτο χρωμόσωμα με την μεγαλύτερη τιμή συνάρτησης καταλληλότητας. Τέλος, το χρωμόσωμα νικητής "winner" επιστρέφεται στο όνομα της συνάρτησης με την εντολή "return winner", διότι δεν πρέπει να επιλεγεί το ίδιο χρωμόσωμα ως γονέας, κάτι που θα φανεί στην συνέχεια με την κλίση της συνάρτησης παρακάτω (πίνακας 6.4).

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΕΠΙΛΟΓΗΣ ΓΟΝΕΑ ΜΕ ΤΟΥΡΝΟΥΑ

```
def tournament(k = 10):
   winner = []
   tournament_chromosomes = []
   solution_fitness = []
   tournament_fitness = []
   population = len(listes_chromosomes)
   # k ΤΟ ΠΛΗΘΟΣ ΤΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΠΟΥ ΘΑ ΕΠΙΛΕΧΘΟΥΝ ΤΥΧΑΙΑ ΓΙΑ ΤΟ
ΤΟΥΡΝΟΥΑ - ΜΕΧΡΙ 10
   try: k = int(k)
   except ValueError: k = 10
```

```
    if (k <= 1) or (k > 10): k = 10
    for c in listes_chromosomes:
        solution_fitness.append(fitness_function(c))
    previous_dt = []
    # ΤΥΧΑΙΑ ΕΠΙΛΟΓΗ ΧΡΩΜΟΣΟΜΑΤΩΝ ΓΙΑ ΤΟΥΡΝΟΥΑ
    for i in range (k):
        dt = random.randint(0, population-1)
        if previous_dt:
            while (dt in previous_dt): dt = random.randint(0, population-1)
        tournament_chromosomes.append(listes_chromosomes[dt])
        tournament_fitness.append(solution_fitness[dt])
        previous_dt.append(dt)
    # ΕΥΡΕΣΗ ΚΑΙ ΕΜΦΑΝΙΣΗ ΝΙΚΗΤΗ ΤΟΥΡΝΟΥΑ
    for ap, c in enumerate(tournament_chromosomes):
        print(ap, c, tournament_fitness[ap], sep = "\t", end ="\n")
    for i in range (k):
        if  tournament_fitness[i] == max(tournament_fitness):
            winner = tournament_chromosomes[i]
            print("winner of tournament is chromosome: ", i, winner, tournament_fitness[i],
sep = "\t", end="\n\n")
            break
    return winner
```

Πίνακας 6.3

Όπως και παραπάνω, δηλαδή στον πίνακα 6.2, ορίζονται πάλι 6 φοιτητές με τα χαρακτηριστικά τους από την συνάρτηση "learning_characteristics(6)" και ο αρχικός πληθυσμός 5 χρωμοσωμάτων με την συνάρτηση "initial_population(6,5)". Επιπλέον ορίζονται οι ομάδες και οι δείκτες πρώτου μέλους κάθε ομάδας με τις συναρτήσεις "create_groups()" και "create_indexes()". Στη συνέχεια του πίνακα 6.4 καλούμε την συνάρτηση "tournament(2)", καταρχήν για να επιλεγεί ο πρώτος νικητής του τουρνουά, που θα αποτελέσει τον πρώτο γονέα στην μεταβλητή winner1. Ακολούθως, με την επαναληπτική δομή "while True: winner2 = tournament(2) if winner1 != winner2: break", επιλέγεται ο δεύτερος νικητής, εφόσον είναι διαφορετικός από τον πρώτο, για να αποτελέσει τον δεύτερο γονέα.

ΚΩΔΙΚΑΣ PYTHON: ΕΠΙΛΟΓΗ ΓΟΝΕΩΝ ΜΕ ΤΟΥΡΝΟΥΑ

```
learning_characteristics(6)
print("\n")
initial_population(6,5)
print("\n")
create_groups()
create_indexes()
```

```
winner1 = tournament(2)
while True:
   winner2 = tournament(2)
   if winner1 != winner2: break
print("parent1", winner1, sep = "\t", end ="\n")
print("parent2", winner2, sep = "\t", end ="\n")
print("\n")
```

Πίνακας 6.4

Παρακάτω εμφανίζονται τα αποτελέσματα της εκτέλεσης του κώδικα του πίνακα 6.4 (Εικόνα 6.2).

```
Shell
0   (3, 2, 1, 4, 5, 4, 2, 1, 2, 1, 2, 2, 1, 2, 3, 2, 3)
1   (1, 1, 1, 4, 2, 1, 3, 4, 1, 1, 2, 2, 1, 1, 3, 1, 2)
2   (3, 2, 5, 4, 1, 2, 1, 2, 3, 2, 2, 1, 2, 2, 1, 1, 2)
3   (1, 5, 1, 1, 1, 2, 2, 1, 4, 1, 2, 1, 2, 2, 1, 2, 2)
4   (4, 2, 3, 4, 1, 1, 2, 3, 4, 2, 1, 1, 1, 3, 3, 1, 1)
5   (1, 3, 5, 5, 3, 5, 2, 3, 2, 1, 1, 2, 1, 3, 1, 2, 1)


0   [0, 1, 2, 3, 4, 5]
1   [0, 3, 4, 2, 5, 1]
2   [2, 4, 3, 0, 5, 1]
3   [2, 5, 3, 4, 0, 1]
4   [1, 0, 4, 2, 5, 3]


2   (0, 1)

(0, 3)

0   [2, 4, 3, 0, 5, 1]  0.005208333333333333
1   [0, 3, 4, 2, 5, 1]  0.008771929824561403
winner of tournament is chromosome:     1   [0, 3, 4, 2, 5, 1]  0.008771929824561403

0   [2, 4, 3, 0, 5, 1]  0.005208333333333333
1   [0, 1, 2, 3, 4, 5]  0.00641025641025641
winner of tournament is chromosome:     1   [0, 1, 2, 3, 4, 5]  0.00641025641025641

parent1 [0, 3, 4, 2, 5, 1]
parent2 [0, 1, 2, 3, 4, 5]
```

Εικόνα 6.2

6.3 Διασταύρωση (crossover)

6.3.1 Διασταύρωση ενός σημείου ανά ομάδα

Για την διασταύρωση ενός σημείου ανά ομάδα, υλοποιήθηκε η συνάρτηση "crossover_one_point_per_group", η οποία παίρνει σαν ορίσματα τους δύο γονείς "parent1" και "parent2". Επιπλέον ορίζονται τρεις λίστες "offspring1", "offspring2" και "offsprings". Στην πρώτη λίστα "offspring1" αποθηκεύεται ο πρώτος απόγονος που θα προκύψει από την διαδικασία της διασταύρωσης ενός σημείου ανά ομάδα. Αντίστοιχα στην λίστα "offspring2" αποθηκεύεται ο δεύτερος απόγονος και στην λίστα "offsprings" αποθηκεύονται και οι δύο απόγονοι, με σκοπό να επιστραφούν στο όνομα της συνάρτησης για να χρησιμοποιηθούν παρακάτω στο τουρνουά μεταξύ γονέων και παιδιών ("return offsprings").

Για την διαδικασία αυτή της διασταύρωσης, χρησιμοποιείται επιπλέον και η καθολική μεταβλητή "index", στην οποία αποθηκεύεται η πλειάδα των τιμών που αντιστοιχούν στο πρώτο μέλος κάθε ομάδας και η οποία δημιουργείται από την κλίση της συνάρτησης "create_indexes()".

Καταρχήν, με την επαναληπτική δομή "for i in index:" προσπελαύνουμε το πρώτο μέλος της κάθε ομάδας του πρώτου γονέα και αντιγράφουμε το δεύτερο και τρίτο μέλος αυτού στα αντίστοιχα μέλη της αντίστοιχης ομάδας του πρώτου απογόνου με τις εντολής "offspring1.append(parent1[i+1])" και "offspring1.append(parent1[i+2])". Ομοίως κάνουμε το ίδιο με τον δεύτερο γονέα στον δεύτερο απόγονο με τις εντολές "offspring2.append(parent2[i+1])" και "offspring2.append(parent2[i+2])".

Ακολούθως συμπληρώνονται τα πρώτα μέλη κάθε ομάδας του πρώτου απογόνου από τον δεύτερο γονέα παίρνοντας με την σειρά που συναντώνται, γονίδια που δεν υπάρχουν στον πρώτο απόγονο ήδη. Η επαναληπτική δομή "for c in parent2:" προσπελαύνει με την σειρά τα γονίδια του δεύτερου γονέα. Εάν το γονίδιο δεν υπάρχει στον πρώτο απόγονο, σύμφωνα με την εντολή ελέγχου "if c not in offspring1:", τότε τοποθετείται ως πρώτο μέλος ομάδας στον απόγονο αυτό με την εντολή "offspring2.insert(ind, c)". Το "ind" αποτελεί τον δείκτη πρώτου μέλους ομάδας του πρώτου απογόνου. Όταν συμπληρωθεί το πρώτο μέλος μίας ομάδας, ο δείκτης αυτός αλλάζει στην τιμή που αναφέρεται στο πρώτο μέλος της επόμενης ομάδας με τις εντολές "i = i + 1 if i < len(index): ind = index[i]".

Η ίδια διαδικασία επαναλαμβάνεται από τον πρώτο γονέα στον δεύτερο απόγονο. Με τις εντολές "print("offspring1 ", offspring1, end="\n")" και "print("offspring2 ", offspring2, end="\n\n")" εκτυπώνονται οι απόγονοι που έχουν σχηματιστεί από αυτή την διαδικασία της διασταύρωσης ενός σημείου ανά ομάδα. Τα παραπάνω φαίνονται στον πίνακα 6.5 που παρατίθεται στην συνέχεια.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΔΙΑΣΤΑΥΡΩΣΗΣ ΕΝΟΣ ΣΗΜΕΙΟΥ ΑΝΑ ΟΜΑΔΑ

```
def crossover_one_point_per_group(parent1, parent2):
   offspring1 = []
   offspring2 = []
   offsprings = []
   # ΑΝΤΕΓΡΑΨΕ ΤΑ ΓΟΝΙΔΙΑ ΔΕΥΤΕΡΟΥ ΚΑΙ ΤΡΙΤΟΥ ΜΕΛΟΥΣ ΚΑΘΕ ΓΟΝΕΑ ΣΤΟ
ΑΝΤΙΣΤΟΙΧΟ ΠΑΙΔΙ
   for i in index:
      offspring1.append(parent1[i+1])
      offspring1.append(parent1[i+2])
      offspring2.append(parent2[i+1])
      offspring2.append(parent2[i+2])
   # ΑΝΤΕΓΡΑΨΕ ΣΤΟ ΠΡΩΤΟ ΠΑΙΔΙ ΩΣ ΠΡΩΤΑ ΜΕΛΗ ΟΜΑΔΩΝ ΓΟΝΙΔΙΑ ΠΟΥ ΔΕΝ ΕΧΟΥΝ
ΤΟΠΟΘΕΤΗΘΕΙ ΜΕ ΤΗ ΣΕΙΡΑ ΠΟΥ ΤΑ ΣΥΝΑΝΤΑΣ ΣΤΟ ΔΕΥΤΕΡΟ ΓΟΝΕΑ
   i = 0
   ind = index[i]
   for c in parent2:
      if c not in offspring1:
         offspring1.insert(ind, c)
         i = i + 1
         if i < len(index): ind = index[i]
      # ΑΝΤΕΓΡΑΨΕ ΣΤΟ ΔΕΥΤΕΡΟ ΠΑΙΔΙ ΩΣ ΠΡΩΤΑ ΜΕΛΗ ΟΜΑΔΩΝ ΓΟΝΙΔΙΑ ΠΟΥ ΔΕΝ
ΕΧΟΥΝ ΤΟΠΟΘΕΤΗΘΕΙ ΜΕ ΤΗ ΣΕΙΡΑ ΠΟΥ ΤΑ ΣΥΝΑΝΤΑΣ ΣΤΟ ΠΡΩΤΟ ΓΟΝΕΑ
   i = 0
   ind = index[i]
   for c in parent1:
      if c not in offspring2:
         offspring2.insert(ind, c)
         i = i + 1
         if i < len(index): ind = index[i]
   print("offspring1 ", offspring1, end="\n")
   print("offspring2 ", offspring2, end="\n\n")
   offsprings.append(offspring1)
   offsprings.append(offspring2)
   return offsprings
```

Πίνακας 6.5

Εκτελώντας με τον ίδιο τρόπο και την ίδια σειρά τις εντολές κατά την πραγματοποίηση του τουρνουά, καλούμε στην συνέχεια τη συνάρτηση διασταύρωση ενός σημείου ανά ομάδα (πίνακας 6.6) για να εμφανιστούν τα αντίστοιχα αποτελέσματα στην εικόνα 6.3.

ΚΩΔΙΚΑΣ PYTHON: ΑΠΟΤΕΛΕΣΜΑ ΕΦΟΡΜΟΓΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ ΕΝΟΣ ΣΗΜΕΙΟΥ ΑΝΑ ΟΜΑΔΑ

```
learning_characteristics(6)
print("\n")
initial_population(6,5)
print("\n")
create_groups()
create_indexes()
winner1 = tournament(2)
while True:
   winner2 = tournament(2)
   if winner1 != winner2: break
print("parent1", winner1, sep = "\t", end = "\n")
print("parent2", winner2, sep = "\t", end = "\n\n")
crossover_one_point_per_group(winner1, winner2)
print("\n")
```

Πίνακας 6.6

```
0   (2, 3, 2, 5, 2, 3, 1, 4, 4, 2, 1, 2, 1, 1, 1, 1, 1)
1   (3, 1, 4, 1, 5, 4, 2, 1, 2, 2, 1, 1, 2, 2, 3, 3, 2)
2   (5, 2, 2, 2, 3, 3, 1, 3, 3, 2, 1, 2, 1, 1, 2, 3, 3)
3   (1, 5, 2, 1, 2, 5, 2, 4, 2, 2, 1, 2, 2, 1, 2, 3, 2)
4   (3, 2, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 3, 2, 3, 3)
5   (4, 3, 5, 2, 5, 1, 1, 4, 2, 1, 2, 2, 1, 2, 2, 1, 2)


0   [0, 1, 2, 3, 4, 5]
1   [5, 2, 1, 3, 0, 4]
2   [5, 3, 1, 4, 0, 2]
3   [1, 0, 3, 4, 5, 2]
4   [0, 4, 2, 1, 3, 5]


2   (0, 1)

(0, 3)

0   [5, 3, 1, 4, 0, 2]  0.005555555555555556
1   [0, 4, 2, 1, 3, 5]  0.005555555555555556
winner of tournament is chromosome:     0   [5, 3, 1, 4, 0, 2]  0.005555555555555556

0   [1, 0, 3, 4, 5, 2]  0.005376344086021506
1   [0, 1, 2, 3, 4, 5]  0.006944444444444444
winner of tournament is chromosome:     1   [0, 1, 2, 3, 4, 5]  0.006944444444444444

parent1 [5, 3, 1, 4, 0, 2]
parent2 [0, 1, 2, 3, 4, 5]

offspring1  [4, 3, 1, 5, 0, 2]
offspring2  [3, 1, 2, 0, 4, 5]
```

Εικόνα 6.3

6.3.2 Τροποποιημένη διασταύρωση δύο σημείων

Όσον αφορά την αναπαραγωγή απογόνων από δύο γονείς με την μέθοδο της τροποποιημένης διασταύρωσης δύο σημείων, ο πίνακας 6.7 περιλαμβάνει έναν εκτεταμένο κώδικα για την υλοποίησή της.

Μετά από την δήλωση των βασικών μεταβλητών "offspring1", "offspring2" και "offsprings", όπως είχε γίνει και στον προηγούμενο κώδικα (βλέπε πίνακα 6.6), αρχικοποιούνται δύο τυχαίοι δείκτες "gparent1A" και "gparent1B" στην τιμή 0 (μηδέν), για τον πρώτο γονέα, που ο καθένας θα αναφέρεται σε κάθε μία από τις δύο ομάδες με την χειρότερη τιμή συνάρτησης καταλληλότητα. Αυτές οι δύο ομάδες αποθηκεύονται στη λίστα-πλειάδα "parent1_worst_groups = []". Το ίδιο γίνεται και για τον δεύτερο γονέα με τους δείκτες "gparent2A" και "gparent2B" και την λίστα-πλειάδα " parent2_worst_groups = []" αντίστοιχα. Η μεταβλητή "group_fitness1" αποθηκεύει τις τιμές της συνάρτησης καταλληλότητας κάθε ομάδας του πρώτου γονέα, ενώ το ίδιο συμβαίνει για τον δεύτερο γονέα στην μεταβλητή "group_fitness2".

Συνεχίζοντας στον κώδικα, για τις g ομάδες, όπως αυτές έχουν προκύψει από την συνάρτησης "create_groups()" κατά τα γνωστά, χρησιμοποιείται μία επαναληπτική δομή "for j in range(17):" για κάθε ένα από τα 17 χαρακτηριστικά των τριών μελών της ομάδας του πρώτου γονέα, υπολογίζοντας την τιμή της συνάρτησης καταλληλότητας της ομάδας g με την εντολή "fitness = fitness + (repitition(parent1, g, j) - idealrepitition(j))". Οι συναρτήσεις "repitition()" και "idealrepitition()", οι οποίες έχουν αναφερθεί στο κεφάλαιο 4, αναφέρονται στον υπολογισμό επανάληψης και ιδανικής επανάληψης χαρακτηριστικού αντίστοιχα. Στη συνέχεια, με την εντολή "print()" για κάθε γονέα, εκτυπώνεται η λίστα "group_fitness1" για τον πρώτο γονέα και "group_fitness2" για τον δεύτερο γονέα.

Επειδή πρέπει να βρεθούν δύο ομάδες με την χειρότερη τιμή συνάρτησης καταλληλότητας, χρησιμοποιείται μία δομή επανάληψης "for g, gf in enumerate(group_fitness1):" και μία συνθήκη ελέγχου "if gf == max(group_fitness1):" δύο φορές. Επίσης ορίζεται ο δείκτης "gparent1A = index[g]", ως δείκτης πρώτου μέλους της πρώτης χειρότερης ομάδας g, προτού επιλεγεί ως τυχαίος δείκτης με την εντολή "gparent1A = random.randint(gparent1A, gparent1A+2)" (βλέπε παρακάτω στον ίδιο κώδικα). Το ίδιο γίνεται και με τον δεύτερο δείκτη "gparent1B = index[g]" με την αντίστοιχη εντολή "gparent1B = random.randint(gparent1B, gparent1B+2)". Επιπλέον, μέσα στην πρώτη προσπέλαση ορίζεται η group_fitness1[g] = -1 για να μην επιλεχθεί ξανά η ομάδα g με την χειρότερη τιμή συνάρτησης καταλληλότητας, στην δεύτερη προσπέλαση. Όλα τα παραπάνω επαναλαμβάνονται και για τον δεύτερο γονέα.

Στη συνέχεια, αφού οριστούν τυχαία οι δείκτες "gparent1A" και "gparent1B" του πρώτου γονέα, γίνεται έλεγχος εάν θα χρειαστεί αντιμετάθεση στην περίπτωση που ο πρώτος δείκτης είναι μεγαλύτερος από τον δεύτερο, με την εντολή "if gparent1A > gparent1B: gparent1A, gparent1B = gparent1B, gparent1A". Το ίδιο γίνεται φυσικά και για του δείκτες του δεύτερου γονέα.

Για την αντιγραφή των γονιδίων μεταξύ των τυχαίων δεικτών του πρώτου γονέα στο πρώτο παιδί, χρησιμοποιείται η εντολή "for i in range (gparent1A, gparent1B+1): offspring1.append(parent1[i])". Αντίστοιχα για την αντιγραφή των γονιδίων μεταξύ των τυχαίων δεικτών του δεύτερου γονέα στο δεύτερο παιδί, χρησιμοποιείται η εντολή "for i in range (gparent2A, gparent2B+1): offspring2.append(parent2[i])". Επειδή πρέπει να συμπληρωθούν γονίδια στο πρώτο παιδί, πριν τον δείκτη "gparent1A" και μετά τον δείκτη "gparent1B", χρησιμοποιούνται δύο επαναληπτικές δομές "for i in range (gparent1A):" και "for i in range (gparent1B+1, len(parent1)):", όπου προσπελαύνονται τα γονίδια του δεύτερου γονέα, σύμφωνα με την εντολή "for c in parent2:" και όποιο γονίδιο δεν υπάρχει ήδη στο πρώτο παιδί, προστίθεται σε αυτό, σύμφωνα με την εντολή " if c not in offspring1: offspring1.insert(i, c)". Η εντολή "break" χρησιμοποιείται όταν συμπληρωθεί ένα γονίδιο στην κατάλληλη θέση, για να τερματίσει η επαναληπτική δομή "for c in parent2:" και να αλλάξει ο δείκτης "i", ώστε να αναφερθεί στην επόμενη θέση στην οποία πρέπει να συμπληρωθεί γονίδιο. Αντίστοιχα τα ίδια γίνονται για το δεύτερο παιδί από τον πρώτο γονέα.

Τέλος εκτυπώνονται οι δύο νέοι απόγονοι, που δημιουργούνται με αυτή την μέθοδο της διασταύρωσης, με τις εντολές "print("offspring3", offspring1, end="\n")" και "print ("offspring4", offspring2, end="\n\n")".

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΤΡΟΠΟΠΟΙΗΜΕΝΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ ΔΥΟ ΣΗΜΕΙΩΝ

```
def crossover_two_point(parent1, parent2):
    offspring1 = []
    offspring2 = []
    offsprings = []
    gparent1A = 0
    gparent1B = 0
    gparent2A = 0
    gparent2B = 0
    group_fitness1 = []
    parent1_worst_groups = []
    parent2_worst_groups = []
    # ΑΞΙΟΛΟΓΗΣΗ ΤΩΝ ΟΜΑΔΩΝ ΚΑΘΕ ΓΟΝΕΑ
    fitness = 0
    for g in groups:
        for j in range(17):
```

```
            fitness = fitness + (repitition(parent1, g, j) - idealrepitition(j))
        group_fitness1.append(fitness)
        fitness = 0
    group_fitness2 = []
    fitness = 0
    for g in groups:
        for j in range(17):
            fitness = fitness + (repitition(parent2, g, j) - idealrepitition(j))
        group_fitness2.append(fitness)
        fitness = 0
    print("parent1 fitness_per_group = ", group_fitness1, end="\n")
    print("parent2 fitness_per_group = ", group_fitness2, end="\n\n")
    # ΕΥΡΕΣΗ ΤΩΝ ΔΥΟ ΟΜΑΔΩΝ ΚΑΘΕ ΓΟΝΕΑ ΜΕ ΤΗ ΧΕΙΡΟΤΕΡΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑ
    for g, gf in enumerate(group_fitness1):
        if gf == max(group_fitness1):
            parent1_worst_groups.append(g)
            gparent1A = index[g]
            group_fitness1[g] = -1
            break
    for g, gf in enumerate(group_fitness1):
        if gf == max(group_fitness1):
            parent1_worst_groups.append(g)
            gparent1B = index[g]
            break
    for g, gf in enumerate(group_fitness2):
        if gf == max(group_fitness2):
            parent2_worst_groups.append(g)
            gparent2A = index[g]
            group_fitness2[g] = -1
            break
    for g, gf in enumerate(group_fitness2):
        if gf == max(group_fitness2):
            parent2_worst_groups.append(g)
            gparent2B = index[g]
            break
    parent1_worst_groups = tuple(parent1_worst_groups)
    parent2_worst_groups = tuple(parent2_worst_groups)
    print("parent1 worst fitness groups = ", parent1_worst_groups, end="\n")
    print("parent2 worst fitness groups = ", parent2_worst_groups, end="\n\n")
    # ΟΡΙΣΜΟΣ ΤΥΧΑΙΩΝ ΔΕΙΚΤΩΝ ΜΕΤΑΞΥ ΤΩΝ ΔΥΟ ΛΙΓΟΤΕΡΟ ΚΑΤΑΛΛΗΛΩΝ ΟΜΑΔΩΝ
ΚΑΘΕ ΓΟΝΕΑ
    gparent1A = random.randint(gparent1A, gparent1A+2)
    gparent1B = random.randint(gparent1B, gparent1B+2)
    gparent2A = random.randint(gparent2A, gparent2A+2)
    gparent2B = random.randint(gparent2B, gparent2B+2)
    if gparent1A > gparent1B: gparent1A, gparent1B = gparent1B, gparent1A
    if gparent2A > gparent2B: gparent2A, gparent2B = gparent2B, gparent2A
    print("gparent1A = ", gparent1A, ", gparent1B = ", gparent1B, end="\n")
```

```
    print("gparent2A = ", gparent2A, " gparent2B = ", gparent2B, end="\n\n")
    # ΑΝΤΕΓΡΑΨΕ ΤΑ ΓΟΝΙΔΙΑ ΜΕΤΑΞΥ ΤΩΝ ΔΕΙΚΤΩΝ ΤΟΥ ΚΑΘΕ ΓΟΝΕΑ ΣΤΑ ΑΝΤΙΣΤΟΙΧΑ
ΠΑΙΔΙΑ
    for i in range (gparent1A, gparent1B+1):
        offspring1.append(parent1[i])
    for i in range (gparent2A, gparent2B+1):
        offspring2.append(parent2[i])
    # ΑΝΤΕΓΡΑΨΕ ΣΤΟ ΠΡΩΤΟ ΠΑΙΔΙ ΓΟΝΙΔΙΑ ΠΟΥ ΔΕΝ ΕΧΟΥΝ ΤΟΠΟΘΕΤΗΘΕΙ ΜΕ ΤΗ ΣΕΙΡΑ
ΠΟΥ ΤΑ ΣΥΝΑΝΤΑΣ ΣΤΟ ΔΕΥΤΕΡΟ ΓΟΝΕΑ
    for i in range (gparent1A): # ΑΠΟ ΤΗΝ ΑΡΧΗ ΜΕΧΡΙ ΠΡΙΝ ΤΟΝ ΠΡΩΤΟ ΔΕΙΚΤΗ ΚΑΙ
        for c in parent2:
            if c not in offspring1:
                offspring1.insert(i, c)
                break
    for i in range (gparent1B+1, len(parent1)): # ΜΕΤΑ ΤΟΝ ΔΕΥΤΕΡΟ ΔΕΙΚΤΗ ΕΩΣ ΤΟ ΤΕΛΟΣ
        for c in parent2:
            if c not in offspring1:
                offspring1.insert(i, c)
                break
        # ΑΝΤΕΓΡΑΨΕ ΣΤΟ ΔΕΥΤΕΡΟ ΠΑΙΔΙ ΓΟΝΙΔΙΑ ΠΟΥ ΔΕΝ ΕΧΟΥΝ ΤΟΠΟΘΕΤΗΘΕΙ ΜΕ ΤΗ
ΣΕΙΡΑ ΠΟΥ ΤΑ ΣΥΝΑΝΤΑΣ ΣΤΟ ΔΕΥΤΕΡΟ ΓΟΝΕΑ
    for i in range (gparent2A): # ΑΠΟ ΤΗΝ ΑΡΧΗ ΜΕΧΡΙ ΠΡΙΝ ΤΟΝ ΠΡΩΤΟ ΔΕΙΚΤΗ ΚΑΙ
        for c in parent1:
            if c not in offspring2:
                offspring2.insert(i, c)
                break
    for i in range (gparent2B+1, len(parent1)): # ΜΕΤΑ ΤΟΝ ΔΕΥΤΕΡΟ ΔΕΙΚΤΗ ΕΩΣ ΤΟ ΤΕΛΟΣ
        for c in parent1:
            if c not in offspring2:
                offspring2.insert(i, c)
                break
    print("offspring3 ", offspring1, end="\n")
    print("offspring4 ", offspring2, end="\n\n")
    offsprings.append(offspring1)
    offsprings.append(offspring2)
    return offsprings
```

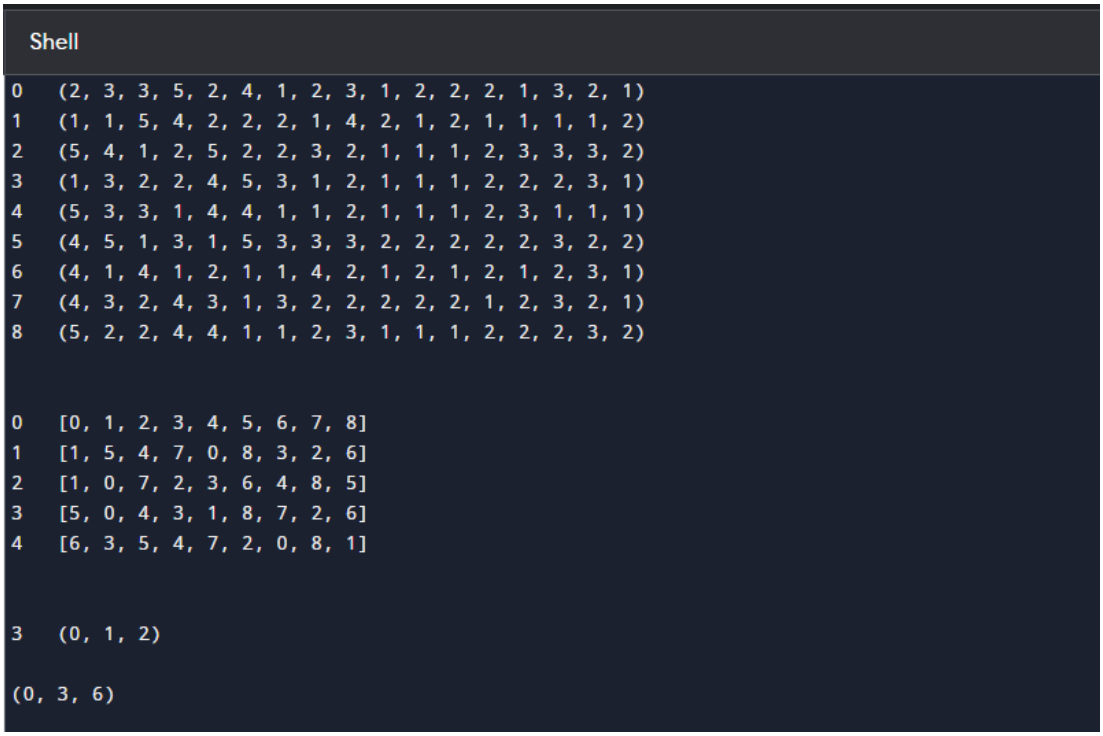
Πίνακας 6.7

Για να πάρουμε εκτυπωμένα τα αποτελέσματα που θέλουμε από την συνάρτηση τροποποιημένης διασταύρωσης δύο σημείων του πίνακα 6.7 (εικόνες 6.4 και 6.5), καλούμε διαδοχικά τις συναρτήσεις και τις εντολές του πίνακα 6.8.

ΚΩΔΙΚΑΣ PYTHON: ΑΠΟΤΕΛΕΣΜΑ ΕΦΑΡΜΟΓΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ ΔΥΟ ΣΗΜΕΙΩΝ

```
learning_characteristics(9)
print("\n")
initial_population(9,5)
print("\n")
create_groups()
create_indexes()
winner1 = tournament(2)
while True:
   winner2 = tournament(2)
   if winner1 != winner2: break
print("parent1", winner1, sep = "\t", end ="\n")
print("parent2", winner2, sep = "\t", end ="\n\n")
crossover_two_point(winner1, winner2)
print("\n")
```

Πίνακας 6.8



Εικόνα 6.4

```
0   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839
1   [1, 5, 4, 7, 0, 8, 3, 2, 6] 0.0043859649122807015
winner of tournament is chromosome:     0   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839

0   [5, 0, 4, 3, 1, 8, 7, 2, 6] 0.005208333333333333
1   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839
winner of tournament is chromosome:     1   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839

0   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839
1   [5, 0, 4, 3, 1, 8, 7, 2, 6] 0.005208333333333333
winner of tournament is chromosome:     0   [6, 3, 5, 4, 7, 2, 0, 8, 1] 0.006172839506172839

0   [1, 5, 4, 7, 0, 8, 3, 2, 6] 0.0043859649122807015
1   [1, 0, 7, 2, 3, 6, 4, 8, 5] 0.004166666666666667
winner of tournament is chromosome:     0   [1, 5, 4, 7, 0, 8, 3, 2, 6] 0.0043859649122807015

parent1 [6, 3, 5, 4, 7, 2, 0, 8, 1]
parent2 [1, 5, 4, 7, 0, 8, 3, 2, 6]

parent1 fitness_per_group =  [66, 54, 42]
parent2 fitness_per_group =  [24, 84, 120]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (2, 1)

gparent1A =  1 , gparent1B =  4
gparent2A =  3 , gparent2B =  8

offspring3  [1, 3, 5, 4, 7, 0, 8, 2, 6]
offspring4  [5, 4, 1, 7, 0, 8, 3, 2, 6]
```

Εικόνα 6.5

6.4 Επιλογή των δύο καταλληλότερων χρωμοσωμάτων μεταξύ γονέων και παιδιών με χρήση τουρνουά

Για την επιλογή των δύο πιο κατάλληλων χρωμοσωμάτων, πραγματοποιείται τουρνουά μεταξύ των δύο γονέων και των τεσσάρων παιδιών, υλοποιώντας την συνάρτηση "parent_children_tournament(parent1, parent2):" του πίνακα 6.9. Προς τον σκοπό αυτό ορίζονται οι μεταβλητές "name1" και "name2", ως συμβολοσειρές που αναφέρονται στα ονόματα των νικητών του τουρνουά, καθώς επίσης και η λίστα "name", η οποία περιλαμβάνει τα ονόματα των χρωμοσωμάτων που συμμετέχουν. Στις μεταβλητές "winner1" και "winner2" αποθηκεύονται τα χρωμοσώματα που αποτελούν τους δύο νικητές του τουρνουά και είναι λίστες.

Η συνάρτηση αυτή καλεί τις δύο συναρτήσεις "offsprings = crossover_one_point _per_group(parent1, parent2)" και "offsprings1 = crossover_two_point(parent1, parent2)" που έχουν υλοποιηθεί στις παραγράφους 6.2 και 6.3 αντίστοιχα. Αυτός είναι και ο λόγος που παίρνει ως ορίσματα μόνο τους δύο γονείς, αφού τα τέσσερα παιδιά δημιουργούνται με αυτόν

τον τρόπο εσωτερικά. Στην μεταβλητή "selected" αποθηκεύονται ως λίστα τα έξι αυτά χρωμοσώματα.

Η αξιολόγησή τους γίνεται με την εντολή "for s in selected: fitness_solution.append(fitness_function(s))". Η ανάδειξη των νικητών μέσω του τουρνουά γίνεται με τον ίδιο τρόπο όπως και στην παράγραφο 6.2, με την διαφορά ότι εδώ αναδεικνύονται δύο νικητές με τον εξής τρόπο. Γίνονται δύο προσπελάσεις των έξι αυτών χρωμοσωμάτων με την εντολή "for i in range(6):" και κάθε φορά επιλέγεται το χρωμόσωμα με την μεγαλύτερη τιμή συνάρτησης καταλληλότητας, ώστε οι νικητές να είναι δύο. Στην πρώτη προσπέλαση η τιμή της συνάρτησης καταλληλότητας γίνεται και εδώ -1 με την "εντολή fitness_solution[i] = -1", για να μην επιλεγεί ξανά το ίδιο χρωμόσωμα ως νικητής στην δεύτερη προσπέλαση. Τέλος εκτυπώνονται και εμφανίζονται οι δύο νικητές του τουρνουά με τις εντολές "print("winner cromosome1 :", name1, winner1, fitness _function(winner1), sep = "\t", end="\n")" για τον πρώτο νικητή και "print("winner cromosome2 : ", name2, winner2, fitness_function(winner2), sep = "\t", end="\n\n")" για τον δεύτερο. Η συνάρτηση επιστρέφει μέσω της μεταβλητής "selected" τους δύο νικητές με την εντολή "return", που θα χρειαστεί στην παραγωγή νέας γενιάς στην επόμενη ενότητα.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΕΠΙΛΟΓΗΣ ΤΩΝ ΔΥΟ ΚΑΤΑΛΛΗΛΟΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΜΕΤΑΞΥ ΓΟΝΕΩΝ ΚΑΙ ΠΑΙΔΙΩΝ ΜΕ ΧΡΗΣΗ ΤΟΥΡΝΟΥΑ

```
def parent_children_tournament(parent1, parent2):
   name1 = ""
   name2 = ""
   name = ["parent1\t", "parent2\t", "offspring1", "offspring2", "offspring3", "offspring4"]
   offsprings = []
   offsprings1 = []
   fitness_solution = []
   winner1 = []
   winner2 = []
   selected = []
   # ΕΦΟΡΜΟΓΗ ΚΑΙ ΤΩΝ ΔΥΟ ΤΕΛΕΣΤΩΝ ΔΙΑΣΤΑΥΡΩΣΗΣ ΓΙΑ ΤΗ ΔΗΜΙΟΥΡΓΙΑ ΤΕΣΣΑΡΩΝ
ΣΥΝΟΛΙΚΑ ΠΑΙΔΙΩΝ
   offsprings = crossover_one_point_per_group(parent1, parent2)
   offsprings1 = crossover_two_point(parent1, parent2)
   # ΑΞΙΟΛΟΓΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΓΟΝΕΩΝ ΚΑΙ ΠΑΙΔΙΩΝ
   selected = [parent1, parent2, offsprings[0], offsprings[1], offsprings1[0], offsprings1[1]]
   for s in selected:
      fitness_solution.append(fitness_function(s))
   for i in range(6):
      print(name[i], selected[i], fitness_solution[i], sep ="\t\t", end="\n")
   print("\n")
   # ΕΥΡΕΣΗ ΚΑΙ ΕΜΦΑΝΙΣΗ ΔΥΟ ΝΙΚΗΤΩΝ ΤΟΥΡΝΟΥΑ
```

```
    for i in range(6):
        if fitness_solution[i] == max(fitness_solution):
            winner1 = selected[i]
            name1 = name[i]
            fitness_solution[i] = -1
            break
    for i in range(6):
        if fitness_solution[i] == max(fitness_solution):
            winner2 = selected[i]
            name2 = name[i]
            break
    print("crossover tournament winners", end="\n")
    print("winner cromosome1 : ", name1, winner1, fitness_function(winner1), sep = "\t",
end="\n")
    print("winner cromosome2 : ", name2, winner2, fitness_function(winner2), sep = "\t",
end="\n\n")
    selected = [winner1, winner2]
    return selected
```

Πίνακας 6.9

Καλώντας στο programiz με την σειρά τις εντολές του πίνακα 6.10 για την υλοποίηση της παραπάνω συνάρτησης, εμφανίζονται στις εικόνες 6.6, 6.7 και 6.8 τα αντίστοιχα αποτελέσματα.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΕΠΙΛΟΓΗΣ ΤΩΝ ΔΥΟ ΚΑΤΑΛΛΗΛΟΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΜΕΤΑΞΥ ΓΟΝΕΩΝ ΚΑΙ ΠΑΙΔΙΩΝ ΜΕ ΧΡΗΣΗ ΤΟΥΡΝΟΥΑ

```
learning_characteristics(9)
initial_population(9,6)
create_groups()
create_indexes()
winner1 = tournament(2)
while True:
    winner2 = tournament(2)
    if winner1 != winner2: break
print("parent1", winner1, sep = "\t", end ="\n")
print("parent2", winner2, sep = "\t", end ="\n\n")
parent_children_tournament(winner1, winner2)
print("\n")
```

Πίνακας 6.10

```
Shell
0   (3, 3, 3, 5, 1, 4, 1, 2, 3, 2, 2, 2, 2, 3, 2, 3, 2)
1   (5, 5, 1, 2, 3, 4, 3, 3, 4, 2, 1, 2, 2, 2, 1, 2, 2)
2   (4, 4, 3, 5, 2, 3, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 3)
3   (1, 4, 2, 5, 1, 5, 3, 3, 3, 2, 2, 1, 2, 1, 1, 1, 3)
4   (2, 4, 1, 5, 1, 4, 2, 2, 3, 1, 2, 1, 2, 1, 2, 1, 3)
5   (5, 2, 3, 2, 5, 4, 3, 3, 3, 2, 2, 2, 2, 1, 3, 2, 2)
6   (5, 3, 1, 5, 5, 4, 1, 1, 2, 1, 1, 2, 2, 2, 3, 3, 3)
7   (4, 3, 2, 2, 3, 4, 2, 3, 3, 2, 2, 2, 2, 3, 2, 2, 1)
8   (5, 2, 3, 2, 5, 3, 3, 1, 3, 1, 1, 2, 1, 1, 2, 1, 1)


0   [0, 1, 2, 3, 4, 5, 6, 7, 8]
1   [0, 4, 6, 3, 2, 8, 5, 7, 1]
2   [1, 0, 5, 4, 7, 2, 6, 3, 8]
3   [0, 8, 7, 6, 5, 3, 4, 2, 1]
4   [4, 8, 5, 3, 7, 0, 6, 1, 2]


3   (0, 1, 2)

(0, 3, 6)
```

Εικόνα 6.6

```
0   [1, 0, 5, 4, 7, 2, 6, 3, 8] 0.003875968992248062
1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629
winner of tournament is chromosome:     1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629

0   [0, 4, 6, 3, 2, 8, 5, 7, 1] 0.0026455026455026454
1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629
winner of tournament is chromosome:     1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.004166666666666667
1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629
winner of tournament is chromosome:     1   [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629

0   [1, 0, 5, 4, 7, 2, 6, 3, 8] 0.003875968992248062
1   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.004166666666666667
winner of tournament is chromosome:     1   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.004166666666666667

parent1 [0, 8, 7, 6, 5, 3, 4, 2, 1]
parent2 [0, 1, 2, 3, 4, 5, 6, 7, 8]

offspring1  [0, 8, 7, 4, 5, 3, 6, 2, 1]
offspring2  [0, 1, 2, 6, 4, 5, 3, 7, 8]

parent1 fitness_per_group =  [102, 78, 36]
parent2 fitness_per_group =  [36, 132, 72]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  2 , gparent1B =  5
gparent2A =  5 , gparent2B =  6

offspring3  [0, 1, 7, 6, 5, 3, 2, 4, 8]
offspring4  [0, 8, 7, 3, 4, 5, 6, 2, 1]
```

Εικόνα 6.7

```
parent1         [0, 8, 7, 6, 5, 3, 4, 2, 1]     0.004629629629629629
parent2         [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.004166666666666667
offspring1      [0, 8, 7, 4, 5, 3, 6, 2, 1]     0.0032679738562091504
offspring2      [0, 1, 2, 6, 4, 5, 3, 7, 8]     0.005050505050505051
offspring3      [0, 1, 7, 6, 5, 3, 2, 4, 8]     0.0032679738562091504
offspring4      [0, 8, 7, 3, 4, 5, 6, 2, 1]     0.0032679738562091504


crossover tournament winners
winner cromosome1 :     offspring2  [0, 1, 2, 6, 4, 5, 3, 7, 8] 0.005050505050505051
winner cromosome2 :     parent1     [0, 8, 7, 6, 5, 3, 4, 2, 1] 0.004629629629629629
```

Εικόνα 6.8

6.5 Δημιουργία νέας γενιάς

Προς τον σκοπό αυτό υλοποιείται η συνάρτηση "produce_new_generation" (βλέπε πίνακα 6.11), η οποία παίρνει δύο ορίσματα. Το "k", το οποίο είναι το πλήθος των χρωμοσωμάτων που επιλέγονται τυχαία για να συμμετάσχουν σε τουρνουά, όπου ο νικητής ενδεχομένως να αποτελέσει τον γονέα και το "crossover_rate", το οποίο είναι η πιθανότητα διασταύρωσης.

Αφού οριστούν οι γνωστές από πριν βοηθητικές μεταβλητές "winner1", "winner2" και "winners", στην συνέχεια με την συνθήκη ελέγχου "if isinstance(crossover_rate, float) == False: crossover_rate = 0.75" γίνεται έλεγχος εάν το "crossover_rate" δεν είναι πραγματικός αριθμός, οπότε σε αυτή την περίπτωση ορίζεται η τιμή του στο 0,75. Επίσης ορίζεται μία μεταβλητή "count" στην οποία αποθηκεύεται το πλήθος των επαναλήψεων της εντολής "while len(new_population) < len(listes_chromosomes):", κατά την οποία δημιουργείται η νέα γενιά, μέχρι ο πληθυσμός της να γίνει ίδιος με την τρέχουσα.

Πιο αναλυτικά, μέσα στην επαναληπτική δομή "while" πραγματοποιείται τουρνουά "k" τυχαίων χρωμοσωμάτων με την συνάρτηση "tournament", μέχρι να επιλεγούν δύο γονείς. Στη συνέχεια πραγματοποιείται έλεγχος μέσω της πιθανότητας εφαρμογής διασταύρωσης. Αυτό γίνεται με την βοήθεια της μεταβλητής "probability = random.randint(1,100)", η οποία ορίζεται ως μία τυχαία ακέραια τιμή μεταξύ ένα (1) και εκατό (100).

Ο έλεγχος γίνεται με βάση τί τιμή θα πάρει η boolean μεταβλητή "s1" και εξαρτάται από το "crossover_rate". Εάν είναι αληθής, θα κληθεί η συνάρτηση "parent_children_tournament", η οποία αναφέρθηκε στην παράγραφο 6.4 και η οποία θα εφαρμόσει τις δύο διασταυρώσεις, θα σχηματίσει τέσσερις απόγονους και θα εφαρμόσει το τουρνουά μεταξύ των δύο γονέων και των τεσσάρων απογόνων, επιλέγοντας τους δύο νικητές με την καλύτερη τιμή συνάρτησης καταλληλότητας, κατά τα γνωστά. Με τις επόμενες δύο

συνθήκες "if winners[0] not in new_population and len(new_population) < len(listes_chromosomes): new_population.append(winners[0])" και "if winners[1] not in new_population and len(new_population) < len(listes_chromosomes): new_population.append(winners[1])" ελέγχεται η δυνατότητα προσθήκης των δύο χρωμοσωμάτων που επιλέχτηκαν από την συνάρτηση "parent_children_tournament" στην νέα γενιά (λίστα "new_population"), εφόσον ο πληθυσμός της είναι μικρότερος από αυτόν της τρέχουσας γενιάς και δεν υπάρχουν τα χρωμοσώματα ήδη στη νέα γενιά.

Στην περίπτωση που η μεταβλητή "s1" είναι ψευδής, δηλαδή όταν δεν θα γίνει διασταύρωση, τότε οι γονείς που έχουν ήδη επιλεγεί από την συνάρτηση "tournament" θα εισαχθούν στη νέα γενιά (λίστα "new_population"), εφόσον ο πληθυσμός της είναι μικρότερος από αυτόν της τρέχουσας γενιάς και δεν υπάρχουν τα χρωμοσώματα ήδη στη νέα γενιά, σύμφωνα με τις συνθήκες "if winner1 not in new_population and len(new_population) < len(listes_chromosomes): new_population.append(winner1)" και "if winner2 not in new_population and len(new_population) < len(listes_chromosomes): new_population.append(winner2)". Τέλος με την εντολή "print("New Population : ", new_population, end="\n\n")" εκτυπώνεται ο νέος πληθυσμός που έχει σχηματιστεί μέχρι την τρέχουσα επανάληψη.

ΚΩΔΙΚΑΣ PYTHON: ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΝΕΑΣ ΓΕΝΙΑΣ

```
def produce_new_generation(k=10, crossover_rate=0.75):
  winner1 = []
  winner2 = []
  winners = []
  if isinstance(crossover_rate, float) == False: crossover_rate = 0.75
  count = 0
  while len(new_population) < len(listes_chromosomes):
    count = count + 1
    print("produce new generation loop", count, "\n")
    # ΤΟΥΡΝΟΥΑ ΕΠΙΛΟΓΗΣ ΓΟΝΕΩΝ
    winner1 = tournament(k)
    while True:
      winner2 = tournament(k)
      if winner1 != winner2: break
    print("parent1 ", winner1, end = "\n")
    print("parent2 ", winner2, end = "\n\n")
    # ΕΛΕΓΧΟΣ ΠΙΘΑΝΟΤΗΤΑΣ ΕΦΑΡΜΟΓΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ
    probability = random.randint(1,100)
    s1 = False
    if crossover_rate == 0.6: s1 = probability <= 60
    elif crossover_rate == 0.7: s1 = probability <= 70
    elif crossover_rate == 0.75: s1 = probability <= 75
```

```
        elif crossover_rate == 0.8: s1 = probability <= 80
        elif crossover_rate == 0.9: s1 = probability <= 90
        else: s1 = probability <= 75
        # ΑΝ Ο ΕΛΕΓΧΟΣ ΕΙΝΑΙ ΑΛΗΘΗΣ ΕΦΑΡΜΟΣΕ ΔΙΑΣΤΑΥΡΩΣΗ ΓΟΝΕΩΝ ΚΑΙ ΤΟΥΡΝΟΥΑ
ΜΕΤΑΞΥ ΓΟΝΙΩΝ ΚΑΙ ΠΑΙΔΙΩΝ
        if s1:
            winners = parent_children_tournament(winner1, winner2)
            # ΑΝ ΟΙ ΔΥΟ ΝΙΚΗΤΕΣ ΤΟΥ ΤΟΥΡΝΟΥΑ ΔΕΝ ΥΠΑΡΧΟΥΝ ΣΤΗΝ ΝΕΑ ΓΕΝΙΑ ΚΑΙ Ο
ΠΛΥΘΥΣΜΟΣ ΤΗΣ ΝΕΑΣ ΓΕΝΙΑΣ ΕΙΝΑΙ ΜΙΚΡΟΤΕΡΟΣ ΑΠΟ ΤΟΝ ΠΛΗΘΥΣΜΟ ΤΗΣ ΤΡΕΧΟΥΣΑΣ
            # ΑΝΤΕΓΡΑΨΕ ΤΟΥΣ ΝΙΚΗΤΕΣ ΣΤΗ ΝΕΑ ΓΕΝΙΑ
            if winners[0] not in new_population and len(new_population) <
len(listes_chromosomes): new_population.append(winners[0])
            if winners[1] not in new_population and len(new_population) <
len(listes_chromosomes): new_population.append(winners[1])
        # ΑΛΛΙΩΣ ΔΕΝ ΕΦΑΡΜΟΖΕΤΑΙ ΔΙΑΣΤΑΥΡΩΣΗ
        else:
            # ΚΑΙ ΑΝ ΟΙ ΔΥΟ ΓΟΝΕΙΣ ΔΕΝ ΥΠΑΡΧΟΥΝ ΣΤΗΝ ΝΕΑ ΓΕΝΙΑ ΚΑΙ Ο ΠΛΥΘΥΣΜΟΣ ΤΗΣ
ΝΕΑΣ ΓΕΝΙΑΣ ΕΙΝΑΙ ΜΙΚΡΟΤΕΡΟΣ ΑΠΟ ΤΟΝ ΠΛΗΘΥΣΜΟ ΤΗΣ ΤΡΕΧΟΥΣΑΣ
            # ΑΝΤΕΓΡΑΨΕ ΤΟΥΣ ΔΥΟ ΓΟΝΕΙΣ ΣΤΗ ΝΕΑ ΓΕΝΙΑ
            if winner1 not in new_population and len(new_population) <
len(listes_chromosomes): new_population.append(winner1)
            if winner2 not in new_population and len(new_population) <
len(listes_chromosomes): new_population.append(winner2)
        print("New Population : ", new_population, end="\n\n")
```

Πίνακας 6.11

Για να κληθεί και να δώσει αποτελέσματα η παραπάνω συνάρτηση, είναι απαραίτητη η κλίση των συναρτήσεων που αναφέρθηκαν στις προηγούμενες παραγράφους αυτού του κεφαλαίου, με την σειρά που φαίνεται στον πίνακα 6.12. Όπως χαρακτηριστικά φαίνεται σε αυτόν τον πίνακα, πέραν από τις γνωστές και από άλλες κλίσεις τους συναρτήσεις (βλέπε πίνακες 6.2, 6.4, 6.6, 6.8 και 6.10), ιδιαίτερη αναφορά πρέπει να γίνει στην κλίση της συνάρτησης ελιτισμού "elitism()". Αυτό είναι απαραίτητο προκειμένου να οριστεί η καθολική λίστα "new_population" και να εισαχθούν σε αυτήν πρώτα τα καλύτερα χρωμοσώματα της τρέχουσας γενιάς, που επιλέγονται μέσω της διαδικασίας του ελιτισμού, όπως αναφέρθηκε στην παράγραφο 6.1.

ΚΩΔΙΚΑΣ PYTHON: ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΦΑΡΜΟΓΗΣ ΔΗΜΙΟΥΡΓΙΑΣ ΝΕΑΣ ΓΕΝΙΑΣ

```
learning_characteristics(9)
print("\n")
initial_population(9,5)
print("\n")
create_groups()
```

```
create_indexes()
elitism()
produce_new_generation(2)
print("New Population Created:", end="\n")
for ap, np in enumerate(new_population):
   print(ap, np, fitness_function(np), sep = "\t", end = "\n")
print("\n")
```

Πίνακας 6.12

Ακολουθούν τα αποτελέσματα που εμφανίζονται στο programiz (εικόνες 6.9 έως 6.18).

```
Shell
0   (1, 2, 3, 2, 5, 1, 1, 4, 2, 2, 1, 1, 1, 2, 2, 1, 2)
1   (1, 4, 4, 3, 1, 4, 1, 3, 1, 2, 2, 2, 2, 1, 3, 2, 1)
2   (3, 4, 1, 1, 1, 2, 2, 1, 4, 1, 1, 1, 2, 2, 2, 3, 1)
3   (2, 2, 3, 1, 2, 3, 3, 4, 2, 2, 2, 1, 2, 2, 1, 2, 2)
4   (3, 1, 3, 2, 3, 2, 1, 2, 3, 2, 1, 2, 2, 1, 2, 3, 1)
5   (1, 5, 2, 5, 3, 3, 2, 1, 2, 1, 1, 1, 1, 2, 3, 1, 1)
6   (5, 4, 2, 2, 4, 5, 3, 4, 1, 2, 1, 2, 1, 1, 1, 3, 2)
7   (5, 2, 5, 2, 5, 3, 1, 4, 3, 1, 1, 1, 1, 3, 1, 3, 3)
8   (4, 4, 2, 5, 5, 2, 3, 3, 4, 2, 2, 1, 2, 1, 3, 3, 3)


0   [0, 1, 2, 3, 4, 5, 6, 7, 8]
1   [7, 3, 6, 0, 4, 8, 2, 5, 1]
2   [6, 1, 8, 5, 4, 2, 3, 7, 0]
3   [0, 3, 1, 4, 7, 8, 6, 2, 5]
4   [2, 1, 3, 7, 8, 5, 0, 4, 6]


3   (0, 1, 2)

(0, 3, 6)

chromosome  0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
chromosome  1   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
chromosome  2   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
chromosome  3   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
chromosome  4   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629


elitism chromosomes
chromosome  0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
```

Εικόνα 6.9

```
produce new generation loop 1

0   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839

parent1  [2, 1, 3, 7, 8, 5, 0, 4, 6]
parent2  [0, 1, 2, 3, 4, 5, 6, 7, 8]

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6]]

produce new generation loop 2

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839

0   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

parent1  [0, 1, 2, 3, 4, 5, 6, 7, 8]
parent2  [0, 3, 1, 4, 7, 8, 6, 2, 5]

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6], [0, 3, 1, 4, 7, 8, 6, 2, 5]]
```

Εικόνα 6.10

```
produce new generation loop 3

0   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

0   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

0   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
1   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629
winner of tournament is chromosome:     1   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629

parent1  [0, 3, 1, 4, 7, 8, 6, 2, 5]
parent2  [2, 1, 3, 7, 8, 5, 0, 4, 6]

offspring1  [0, 3, 1, 4, 7, 8, 6, 2, 5]
offspring2  [0, 1, 3, 7, 8, 5, 2, 4, 6]

parent1 fitness_per_group =  [72, 66, 60]
parent2 fitness_per_group =  [54, 60, 102]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (2, 1)

gparent1A =  0 , gparent1B =  5
gparent2A =  5 , gparent2B =  8

offspring3  [0, 3, 1, 4, 7, 8, 2, 5, 6]
offspring4  [3, 1, 7, 8, 2, 5, 0, 4, 6]
```

Εικόνα 6.11

```
Shell                                                                    Clear

parent1         [0, 3, 1, 4, 7, 8, 6, 2, 5]     0.005050505050505051
parent2         [2, 1, 3, 7, 8, 5, 0, 4, 6]     0.004629629629629629
offspring1      [0, 3, 1, 4, 7, 8, 6, 2, 5]     0.005050505050505051
offspring2      [0, 1, 3, 7, 8, 5, 2, 4, 6]     0.004629629629629629
offspring3      [0, 3, 1, 4, 7, 8, 2, 5, 6]     0.005050505050505051
offspring4      [3, 1, 7, 8, 2, 5, 0, 4, 6]     0.0045045045045045045


crossover tournament winners
winner cromosome1 :     parent1     [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner cromosome2 :     offspring1  [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6], [0, 3, 1, 4, 7, 8, 6, 2, 5]]
```

Εικόνα 6.12

```
produce new generation loop 4

0   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

0   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
1   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
winner of tournament is chromosome:     0   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

0   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629

parent1  [0, 3, 1, 4, 7, 8, 6, 2, 5]
parent2  [2, 1, 3, 7, 8, 5, 0, 4, 6]

offspring1  [0, 3, 1, 4, 7, 8, 6, 2, 5]
offspring2  [0, 1, 3, 7, 8, 5, 2, 4, 6]

parent1 fitness_per_group =  [72, 66, 60]
parent2 fitness_per_group =  [54, 60, 102]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (2, 1)

gparent1A =  1 , gparent1B =  3
gparent2A =  5 , gparent2B =  7

offspring3  [2, 3, 1, 4, 7, 8, 5, 0, 6]
offspring4  [3, 1, 7, 8, 6, 5, 0, 4, 2]
```

Εικόνα 6.13

```
Shell                                                                    Clear

parent1         [0, 3, 1, 4, 7, 8, 6, 2, 5]     0.005050505050505051
parent2         [2, 1, 3, 7, 8, 5, 0, 4, 6]     0.004629629629629629
offspring1      [0, 3, 1, 4, 7, 8, 6, 2, 5]     0.005050505050505051
offspring2      [0, 1, 3, 7, 8, 5, 2, 4, 6]     0.004629629629629629
offspring3      [2, 3, 1, 4, 7, 8, 5, 0, 6]     0.004901960784313725
offspring4      [3, 1, 7, 8, 6, 5, 0, 4, 2]     0.005376344086021506


crossover tournament winners
winner cromosome1 :     offspring4  [3, 1, 7, 8, 6, 5, 0, 4, 2] 0.005376344086021506
winner cromosome2 :     parent1     [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6], [0, 3, 1, 4, 7, 8, 6, 2, 5],
    [3, 1, 7, 8, 6, 5, 0, 4, 2]]
```

Εικόνα 6.14

Shell

```
produce new generation loop 5

0   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
1   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
winner of tournament is chromosome:     0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839

parent1  [7, 3, 6, 0, 4, 8, 2, 5, 1]
parent2  [0, 1, 2, 3, 4, 5, 6, 7, 8]

offspring1  [0, 3, 6, 2, 4, 8, 7, 5, 1]
offspring2  [3, 1, 2, 6, 4, 5, 0, 7, 8]

parent1 fitness_per_group =  [90, 66, 66]
parent2 fitness_per_group =  [42, 36, 84]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (2, 0)

gparent1A =  0 , gparent1B =  5
gparent2A =  1 , gparent2B =  8

offspring3  [7, 3, 6, 0, 4, 8, 1, 2, 5]
offspring4  [0, 1, 2, 3, 4, 5, 6, 7, 8]

parent1         [7, 3, 6, 0, 4, 8, 2, 5, 1]     0.0045045045045045045
parent2         [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.006172839506172839
offspring1      [0, 3, 6, 2, 4, 8, 7, 5, 1]     0.004166666666666667
offspring2      [3, 1, 2, 6, 4, 5, 0, 7, 8]     0.005376344086021506
offspring3      [7, 3, 6, 0, 4, 8, 1, 2, 5]     0.0045045045045045045
offspring4      [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.006172839506172839
```

Εικόνα 6.15

```
crossover tournament winners
winner cromosome1 :     parent2     [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
winner cromosome2 :     offspring4  [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6], [0, 3, 1, 4, 7, 8, 6, 2, 5],
    [3, 1, 7, 8, 6, 5, 0, 4, 2]]
```

Εικόνα 6.16

Shell

```
produce new generation loop 6

0   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner of tournament is chromosome:     1   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051

0   [6, 1, 8, 5, 4, 2, 3, 7, 0] 0.003205128205128205
1   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
winner of tournament is chromosome:     1   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045

parent1  [0, 3, 1, 4, 7, 8, 6, 2, 5]
parent2  [7, 3, 6, 0, 4, 8, 2, 5, 1]

offspring1  [6, 3, 1, 0, 7, 8, 4, 2, 5]
offspring2  [0, 3, 6, 7, 4, 8, 2, 5, 1]

parent1 fitness_per_group =  [72, 66, 60]
parent2 fitness_per_group =  [90, 66, 66]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (0, 1)

gparent1A =  1 , gparent1B =  4
gparent2A =  0 , gparent2B =  4

offspring3  [6, 3, 1, 4, 7, 0, 8, 2, 5]
offspring4  [7, 3, 6, 0, 4, 1, 8, 2, 5]

parent1         [0, 3, 1, 4, 7, 8, 6, 2, 5]     0.005050505050505051
parent2         [7, 3, 6, 0, 4, 8, 2, 5, 1]     0.0045045045045045045
offspring1      [6, 3, 1, 0, 7, 8, 4, 2, 5]     0.004273504273504274
offspring2      [0, 3, 6, 7, 4, 8, 2, 5, 1]     0.004166666666666667
offspring3      [6, 3, 1, 4, 7, 0, 8, 2, 5]     0.003875968992248062
offspring4      [7, 3, 6, 0, 4, 1, 8, 2, 5]     0.003968253968253968
```

Εικόνα 6.17

```
crossover tournament winners
winner cromosome1 :     parent1     [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
winner cromosome2 :     parent2     [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045

New Population :  [[0, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 3, 7, 8, 5, 0, 4, 6], [0, 3, 1, 4, 7, 8, 6, 2, 5],
    [3, 1, 7, 8, 6, 5, 0, 4, 2], [7, 3, 6, 0, 4, 8, 2, 5, 1]]

New Population Created:
0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.006172839506172839
1   [2, 1, 3, 7, 8, 5, 0, 4, 6] 0.004629629629629629
2   [0, 3, 1, 4, 7, 8, 6, 2, 5] 0.005050505050505051
3   [3, 1, 7, 8, 6, 5, 0, 4, 2] 0.005376344086021506
4   [7, 3, 6, 0, 4, 8, 2, 5, 1] 0.0045045045045045045
```

Εικόνα 6.18

6.6 Εφαρμογή μετάλλαξης

Στο σημείο αυτό, αξιοποιούνται όλες οι παραπάνω συναρτήσεις, για να δημιουργηθεί μία νέα γενιά, κάτι ιδιαίτερα σημαντικό για την εξέλιξη της διαδικασίας του γενετικού αλγορίθμου. Το τελευταίο στάδιο, που πραγματοποιείται σε αυτήν την παράγραφο είναι η μετάλλαξη. Στον κώδικα του πίνακα 6.13 υλοποιείται η συνάρτηση "mutation" εφαρμογής της μετάλλαξης, η οποία παίρνει ως όρισμα την πιθανότητα μετάλλαξης ("mutation_rate") και η τιμή της ορίζεται στο 0.05, για την περίπτωση που ο χρήστης δεν βάλει κάποια τιμή στο όρισμα. Παρ' όλ' αυτά, προβλέπεται εξαρχής κατάλληλος έλεγχος, με την εντολή "if isinstance(mutation_rate, float) == False: mutation_rate = 0.05", ώστε εάν δεν δοθεί πραγματικός αριθμός ως όρισμα, ορίζεται και πάλι η τιμή της πιθανότητας μετάλλαξης στο 0,05, δηλαδή στο 5%.

Στη συνέχεια, αφού οριστεί και αρχικοποιηθεί ένας μετρητής για το πλήθος των μεταλλάξεων που θα υλοποιηθούν ("count_mutation = 0"), γίνεται χρήση μίας επαναληπτικής δομής "for count, ch in enumerate(new_population):". Για κάθε χρωμόσωμα της νέας γενιάς (λίστας "new_population") ελέγχεται η συνθήκη "s1" και εάν ισχύει, πραγματοποιείται μετάλλαξη. Δηλαδή επιλέγονται τυχαία στο τρέχον χρωμόσωμα δύο ομάδες "group1 = random.choice(groups)" και "group2 = random.choice(groups)", μέχρι να οριστούν δύο διαφορετικές ομάδες, με την εντολή "while group1 == group2: group2 = random .choice(groups)". Έπειτα επιλέγεται τυχαία ένα γονίδιο από την κάθε ομάδα ("gene1 = random.randint(index[group1], index[group1]+2)" και "gene2 = random.randint(index [group2], index[group2]+2)") και πραγματοποιείται η μετάλλαξη ανταλλαγής.

Πιο συγκεκριμένα, στην λίστα "mutated_ch" που έχει γίνει αντιγραφή του χρωμοσώματος "ch", με την εντολή "mutated_ch = ch.copy()", ορίζεται και αποθηκεύεται το μεταλλαγμένο χρωμόσωμα, με την εντολή "mutated_ch[gene1], mutated_ch[gene2] = mutated_ch[gene2], mutated_ch[gene1]".

Ακολουθεί ο έλεγχος "if fitness_function(mutated_ch) > fitness_function(ch):", όπου ελέγχεται εάν η τιμή της συνάρτησης καταλληλότητας του μεταλλαγμένου χρωμοσώματος είναι μεγαλύτερη από αυτή του χρωμοσώματος πριν υποστεί μετάλλαξη. Στην περίπτωση που η συνθήκη είναι αληθής, τότε αφαιρείται από τη νέα γενιά το χρωμόσωμα "ch" με την εντολή "new_population.pop(count)" και στην θέση του προστίθεται το μεταλλαγμένο χρωμόσωμα "mutated_ch", με την εντολή "new_population.insert(count, mutated_ch)". Τέλος εμφανίζεται το αντίστοιχο μήνυμα, με την εντολή "print("mutated chromosome", mutated_ch, "replaced chromosome", ch, sep="\t", end = "\n\n")" και εφαρμόζεται ο έλεγχος "if count_mutation == 0: print("There was no mutation", end="\n\n")", στην περίπτωση που δεν γίνει καμία μετάλλαξη.

ΚΩΔΙΚΑΣ PYTHON: ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΦΑΡΜΟΓΗΣ ΜΕΤΑΛΛΑΞΗΣ

```
def mutation(mutation_rate=0.05):
   if isinstance(mutation_rate, float) == False: mutation_rate = 0.05
   count_mutation = 0
   print("Results of Mutation", end ="\n")
   # ΓΙΑ ΚΑΘΕ ΕΝΑ ΧΡΩΜΟΣΩΜΑ ΤΗΣ ΝΕΑΣ ΓΕΝΙΑΣ
   for count, ch in enumerate(new_population):
      # ΕΛΕΓΧΟΣ ΠΙΘΑΝΟΤΗΤΑΣ ΕΦΑΡΜΟΓΗΣ ΜΕΤΑΛΛΑΞΗΣ
      s1 = False
      probability = random.randint(1,100)
      if (mutation_rate >= 0.01 and mutation_rate < 0.02): s1 = probability == 1
      elif (mutation_rate >= 0.02 and mutation_rate < 0.03): s1 = probability <= 2
      elif (mutation_rate >= 0.03 and mutation_rate < 0.04): s1 = probability <= 3
      elif (mutation_rate >= 0.04 and mutation_rate < 0.05): s1 = probability <= 4
      elif (mutation_rate >= 0.05 and mutation_rate < 0.06): s1 = probability <= 5
      elif (mutation_rate >= 0.06 and mutation_rate < 0.07): s1 = probability <= 6
      elif (mutation_rate == 0.07): s1 = propability <= 7
      else: s1 = probability <= 5
      # ΑΝ ΙΣΧΥΕΙ Η ΣΥΝΘΗΚΗ ΕΦΑΡΜΟΣΕ ΜΕΤΑΛΛΑΞΗ
      if s1:
         count_mutation = count_mutation + 1
         mutated_ch = []
         mutated_ch = ch.copy()
         group1 = group2 = gene1 = gene2 = -1
         # ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΔΥΟ ΟΜΑΔΕΣ ΤΟΥ ΧΡΩΜΟΣΩΜΑΤΟΣ
         group1 = random.choice(groups)
         group2 = random.choice(groups)
         while group1 == group2: group2 = random.choice(groups)
         # ΕΠΕΛΕΞΕ ΤΥΧΑΙΑ ΕΝΑ ΓΟΝΙΔΙΟ ΑΠΟ ΚΑΘΕ ΓΟΝΕΑ
         gene1 = random.randint(index[group1], index[group1]+2)
         gene2 = random.randint(index[group2], index[group2]+2)
         # ΚΑΙ ΑΝΤΑΛΛΑΞΕ ΤΑ
         mutated_ch[gene1], mutated_ch[gene2] = mutated_ch[gene2],
mutated_ch[gene1]
         print("chromosome\t\t", ch, fitness_function(ch), sep="\t\t", end="\n")
         print("mutated chromosome", mutated_ch, fitness_function(mutated_ch),
sep="\t\t", end="\n\n")
         # ΑΝ ΤΟ ΜΕΤΑΛΛΑΓΜΕΝΟ ΧΡΩΜΟΣΩΜΑ ΕΙΝΑΙ ΚΑΤΑΛΛΗΛΟΤΕΡΟ
         if fitness_function(mutated_ch) > fitness_function(ch):
            new_population.pop(count) # ΑΦΑΙΡΕΣΕ ΤΟ ΧΡΩΜΟΣΩΜΑ ΑΠΟ ΤΗ ΝΕΑ ΓΕΝΙΑ
            new_population.insert(count, mutated_ch) # ΚΑΙ ΤΟΠΟΘΕΤΗΣΕ ΤΟ
ΜΕΤΑΛΛΑΓΜΕΝΟ
            print("mutated chromosome", mutated_ch, "replaced chromosome", ch,
sep="\t", end = "\n\n")
   if count_mutation == 0: print("There was no mutation", end="\n\n")
```

Πίνακας 6.13

Όπως και παραπάνω, εκτελείται στο programiz η αλληλουχία των εντολών του κώδικα του πίνακα 6.14, με τα αποτελέσματα που φαίνονται στις εικόνες 6.19-6.25.

ΚΩΔΙΚΑΣ PYTHON: ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΦΑΡΜΟΓΗΣ ΜΕΤΑΛΛΑΞΗΣ

```
learning_characteristics(9)
print("\n")
initial_population(9,5)
print("\n")
create_groups()
create_indexes()
elitism()
produce_new_generation(2)
mutation()
print("New Population Created:", end="\n")
for ap, np in enumerate(new_population):
   print(ap, np, fitness_function(np), sep = "\t", end = "\n")
print("\n")
```

Πίνακας 6.14

Shell

```
0   (3, 4, 3, 5, 3, 2, 3, 4, 1, 1, 1, 1, 2, 1, 2, 2, 2)
1   (5, 4, 2, 2, 4, 2, 3, 4, 1, 2, 2, 2, 2, 1, 3, 3, 2)
2   (2, 3, 4, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1)
3   (1, 2, 3, 3, 2, 1, 2, 1, 4, 1, 1, 2, 1, 2, 3, 3, 2)
4   (4, 5, 4, 4, 5, 5, 3, 3, 4, 1, 1, 1, 2, 3, 3, 3, 2)
5   (1, 5, 3, 3, 5, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 3, 2)
6   (5, 4, 3, 2, 1, 1, 2, 3, 4, 2, 1, 1, 1, 1, 2, 2, 3)
7   (1, 1, 5, 5, 2, 5, 1, 3, 3, 2, 2, 1, 1, 2, 3, 1, 1)
8   (1, 5, 3, 2, 4, 4, 3, 1, 4, 2, 2, 2, 2, 2, 3, 2, 2)


0   [0, 1, 2, 3, 4, 5, 6, 7, 8]
1   [1, 5, 4, 0, 7, 8, 2, 3, 6]
2   [8, 4, 2, 7, 1, 6, 5, 0, 3]
3   [8, 6, 2, 1, 5, 7, 4, 3, 0]
4   [1, 4, 3, 2, 6, 7, 8, 0, 5]


3   (0, 1, 2)

(0, 3, 6)

chromosome  0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.003703703703703704
chromosome  1   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
chromosome  2   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
chromosome  3   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
chromosome  4   [1, 4, 3, 2, 6, 7, 8, 0, 5] 0.0040650406504065045


elitism chromosomes
chromosome  0   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
```

Εικόνα 6.19

```
produce new generation loop 1

0   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
1   [1, 4, 3, 2, 6, 7, 8, 0, 5] 0.0040650406504065045
winner of tournament is chromosome:     0   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762

0   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
1   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
winner of tournament is chromosome:     0   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762

parent1  [1, 5, 4, 0, 7, 8, 2, 3, 6]
parent2  [8, 4, 2, 7, 1, 6, 5, 0, 3]

offspring1  [2, 5, 4, 1, 7, 8, 0, 3, 6]
offspring2  [5, 4, 2, 7, 1, 6, 8, 0, 3]

parent1 fitness_per_group =  [96, 48, 66]
parent2 fitness_per_group =  [60, 54, 96]

parent1 worst fitness groups =  (0, 2)
parent2 worst fitness groups =  (2, 0)

gparent1A =  0 , gparent1B =  7
gparent2A =  0 , gparent2B =  8

offspring3  [1, 5, 4, 0, 7, 8, 2, 3, 6]
offspring4  [8, 4, 2, 7, 1, 6, 5, 0, 3]
```

Εικόνα 6.20

```
parent1         [1, 5, 4, 0, 7, 8, 2, 3, 6]     0.004761904761904762
parent2         [8, 4, 2, 7, 1, 6, 5, 0, 3]     0.004761904761904762
offspring1      [2, 5, 4, 1, 7, 8, 0, 3, 6]     0.0035460992907801418
offspring2      [5, 4, 2, 7, 1, 6, 8, 0, 3]     0.004166666666666667
offspring3      [1, 5, 4, 0, 7, 8, 2, 3, 6]     0.004761904761904762
offspring4      [8, 4, 2, 7, 1, 6, 5, 0, 3]     0.004761904761904762


crossover tournament winners
winner cromosome1 :     parent1     [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
winner cromosome2 :     parent2     [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762

New Population :  [[1, 5, 4, 0, 7, 8, 2, 3, 6], [8, 4, 2, 7, 1, 6, 5, 0, 3]]
```

Εικόνα 6.21

```
produce new generation loop 2

0   [1, 4, 3, 2, 6, 7, 8, 0, 5] 0.0040650406504065045
1   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
winner of tournament is chromosome:     1   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762

0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
1   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.003703703703703704
winner of tournament is chromosome:     0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.003703703703703704
1   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
winner of tournament is chromosome:     1   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762

parent1  [8, 6, 2, 1, 5, 7, 4, 3, 0]
parent2  [8, 4, 2, 7, 1, 6, 5, 0, 3]

offspring1  [8, 6, 2, 4, 5, 7, 1, 3, 0]
offspring2  [8, 4, 2, 5, 1, 6, 7, 0, 3]

parent1 fitness_per_group =  [66, 54, 90]
parent2 fitness_per_group =  [60, 54, 96]

parent1 worst fitness groups =  (2, 0)
parent2 worst fitness groups =  (2, 0)

gparent1A =  2 , gparent1B =  7
gparent2A =  0 , gparent2B =  6

offspring3  [8, 6, 2, 1, 5, 7, 4, 3, 0]
offspring4  [8, 4, 2, 7, 1, 6, 5, 3, 0]
```

Εικόνα 6.22

```
parent1         [8, 6, 2, 1, 5, 7, 4, 3, 0]     0.004761904761904762
parent2         [8, 4, 2, 7, 1, 6, 5, 0, 3]     0.004761904761904762
offspring1      [8, 6, 2, 4, 5, 7, 1, 3, 0]     0.004629629629629629
offspring2      [8, 4, 2, 5, 1, 6, 7, 0, 3]     0.005376344086021506
offspring3      [8, 6, 2, 1, 5, 7, 4, 3, 0]     0.004761904761904762
offspring4      [8, 4, 2, 7, 1, 6, 5, 3, 0]     0.004761904761904762


crossover tournament winners
winner cromosome1 :     offspring2  [8, 4, 2, 5, 1, 6, 7, 0, 3] 0.005376344086021506
winner cromosome2 :     parent1     [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762

New Population :  [[1, 5, 4, 0, 7, 8, 2, 3, 6], [8, 4, 2, 7, 1, 6, 5, 0, 3], [8, 4, 2, 5, 1, 6, 7, 0, 3],
    [8, 6, 2, 1, 5, 7, 4, 3, 0]]

produce new generation loop 3

0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
1   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
winner of tournament is chromosome:     0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762

0   [1, 4, 3, 2, 6, 7, 8, 0, 5] 0.0040650406504065045
1   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
winner of tournament is chromosome:     1   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762

parent1  [8, 6, 2, 1, 5, 7, 4, 3, 0]
parent2  [1, 5, 4, 0, 7, 8, 2, 3, 6]

New Population :  [[1, 5, 4, 0, 7, 8, 2, 3, 6], [8, 4, 2, 7, 1, 6, 5, 0, 3], [8, 4, 2, 5, 1, 6, 7, 0, 3],
    [8, 6, 2, 1, 5, 7, 4, 3, 0]]
```

Εικόνα 5.63

```
produce new generation loop 4

0   [0, 1, 2, 3, 4, 5, 6, 7, 8] 0.003703703703703704
1   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762
winner of tournament is chromosome:     1   [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762

0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
1   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
winner of tournament is chromosome:     0   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762

parent1  [1, 5, 4, 0, 7, 8, 2, 3, 6]
parent2  [8, 6, 2, 1, 5, 7, 4, 3, 0]

offspring1  [2, 5, 4, 1, 7, 8, 0, 3, 6]
offspring2  [1, 6, 2, 4, 5, 7, 8, 3, 0]

parent1 fitness_per_group =  [96, 48, 66]
parent2 fitness_per_group =  [66, 54, 90]

parent1 worst fitness groups =  (0, 2)
parent2 worst fitness groups =  (2, 0)

gparent1A =  2 , gparent1B =  7
gparent2A =  2 , gparent2B =  7

offspring3  [6, 1, 4, 0, 7, 8, 2, 3, 5]
offspring4  [0, 8, 2, 1, 5, 7, 4, 3, 6]
```

Εικόνα 6.24

```
parent1         [1, 5, 4, 0, 7, 8, 2, 3, 6]     0.004761904761904762
parent2         [8, 6, 2, 1, 5, 7, 4, 3, 0]     0.004761904761904762
offspring1      [2, 5, 4, 1, 7, 8, 0, 3, 6]     0.0035460992907801418
offspring2      [1, 6, 2, 4, 5, 7, 8, 3, 0]     0.0043859649122807015
offspring3      [6, 1, 4, 0, 7, 8, 2, 3, 5]     0.005376344086021506
offspring4      [0, 8, 2, 1, 5, 7, 4, 3, 6]     0.004761904761904762


crossover tournament winners
winner cromosome1 :     offspring3  [6, 1, 4, 0, 7, 8, 2, 3, 5] 0.005376344086021506
winner cromosome2 :     parent1     [1, 5, 4, 0, 7, 8, 2, 3, 6] 0.004761904761904762

New Population :  [[1, 5, 4, 0, 7, 8, 2, 3, 6], [8, 4, 2, 7, 1, 6, 5, 0, 3], [8, 4, 2, 5, 1, 6, 7, 0, 3],
    [8, 6, 2, 1, 5, 7, 4, 3, 0], [6, 1, 4, 0, 7, 8, 2, 3, 5]]

Results of Mutation
chromosome              [1, 5, 4, 0, 7, 8, 2, 3, 6]     0.004761904761904762
mutated chromosome      [1, 5, 6, 0, 7, 8, 2, 3, 4]     0.005376344086021506

mutated chromosome  [1, 5, 6, 0, 7, 8, 2, 3, 4] replaced chromosome [1, 5, 4, 0, 7, 8, 2, 3, 6]

New Population Created:
0   [1, 5, 6, 0, 7, 8, 2, 3, 4] 0.005376344086021506
1   [8, 4, 2, 7, 1, 6, 5, 0, 3] 0.004761904761904762
2   [8, 4, 2, 5, 1, 6, 7, 0, 3] 0.005376344086021506
3   [8, 6, 2, 1, 5, 7, 4, 3, 0] 0.004761904761904762
4   [6, 1, 4, 0, 7, 8, 2, 3, 5] 0.005376344086021506
```

Εικόνα 6.25

Κεφάλαιο 7

7 Γενετικός αλγόριθμος ομαδοποίησης, δοκιμές και αποτελέσματα

7.1 Υλοποίηση γενετικού αλγορίθμου ομαδοποίησης

Στην παράγραφο αυτή συγκεντρώνονται και συνοψίζονται όλα όσα ειπώθηκαν για τον γενετικό αλγόριθμο και υλοποιήθηκαν στα δύο προηγούμενα κεφάλαια με την χρήση της γλώσσας προγραμματισμού Python 3. Στα κεφάλαια αυτά (5 & 6) υλοποιήθηκαν οι απαραίτητες συναρτήσεις και ο κώδικάς τους έτρεξε στο programiz, από το οποίο εξήχθησαν και τα ανάλογα αποτελέσματα. Στην παρούσα παράγραφο υλοποιείται η βιβλιοθήκη ggabasicfunctions.py, η οποία περιλαμβάνει όλες τις συναρτήσεις υλοποίησης των βημάτων του γενετικού αλγορίθμου ομαδοποίησης, όπως φαίνεται στον πίνακα 7.1. Η βιβλιοθήκη αυτή είναι αποθηκευμένη τοπικά και δεν τρέχει στο programiz, θα είναι όμως χρήσιμη παρακάτω, στην παράγραφο 7.3, όπου υλοποιείται ο γενετικός αλγόριθμός. Στον πίνακα 7.2 αναφέρεται κατά αντιστοιχία και συνοπτικά, τί υλοποιεί η κάθε συνάρτηση.

ΣΥΝΑΡΤΗΣΕΙΣ ΒΙΒΛΙΟΘΗΚΗΣ ggabasicfunctions.py
1) learning_characteristics(number_students=48)
2) initial_population(number_students=48, number_chromosomes=100)
3) idealrepitition(j=0)
4) create_groups()
5) create_indexes()
6) repitition(ch, group=0, j=0)
7) fitness_function(c)
8) elitism()
9) tournament(k = 10)
10) crossover_one_point_per_group(parent1, parent2)
11) crossover_two_point(parent1, parent2)
12) parent_children_tournament(parent1, parent2)
13) produce_new_generation(k=10, crossover_rate=0.75)
14) mutation(mutation_rate=0.05)

Πίνακας 7.1

ΠΕΡΙΓΡΑΦΗ ΣΥΝΑΡΤΗΣΕΩΝ ΒΙΒΛΙΟΘΗΚΗΣ ggabasicfunctions.py
1) ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΦΟΙΤΗΤΩΝ ΜΕ 17 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ
2) ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΑΡΧΙΚΟΥ ΠΛΥΘΗΣΜΟΥ ΜΕ ΤΥΧΑΙΑ ΧΡΩΜΟΣΩΜΑΤΑ
3) ΣΥΝΑΡΤΗΣΗ ΙΔΑΝΙΚΗΣ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
4) ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΟΜΑΔΩΝ
5) ΣΥΝΑΡΤΗΣΗ ΔΕΙΚΤΩΝ ΠΡΩΤΟΥ ΜΕΛΟΥΣ ΚΑΘΕ ΟΜΑΔΑΣ
6) ΣΥΝΑΡΤΗΣΗ ΕΠΑΝΑΛΗΨΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ
7) ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΧΡΩΜΟΣΩΜΑΤΟΣ

8) ΣΥΝΑΡΤΗΣΗ ΕΦΑΡΜΟΓΗΣ ΕΛΙΤΙΣΜΟΥ
9) ΣΥΝΑΡΤΗΣΗ ΕΠΙΛΟΓΗΣ ΓΟΝΕΑ ΜΕ ΤΟΥΡΝΟΥΑ
10) ΣΥΝΑΡΤΗΣΗ ΔΙΑΣΤΑΥΡΩΣΗΣ ΕΝΟΣ ΣΗΜΕΙΟΥ ΑΝΑ ΟΜΑΔΑ
11) ΣΥΝΑΡΤΗΣΗ ΤΡΟΠΟΠΟΙΗΜΕΝΗΣ ΔΙΑΣΤΑΥΡΩΣΗΣ ΔΥΟ ΣΗΜΕΙΩΝ
12) ΣΥΝΑΡΤΗΣΗ ΕΠΙΛΟΓΗΣ ΤΩΝ ΔΥΟ ΚΑΤΑΛΛΗΛΟΤΕΡΩΝ ΧΡΩΜΟΣΩΜΑΤΩΝ ΜΕΤΑΞΥ ΓΟΝΕΩΝ ΚΑΙ ΠΑΙΔΙΩΝ ΜΕ ΧΡΗΣΗ ΤΟΥΡΝΟΥΑ
13) ΣΥΝΑΡΤΗΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΝΕΑΣ ΓΕΝΙΑΣ
14) ΣΥΝΑΡΤΗΣΗ ΜΕΤΑΛΛΑΞΗΣ

Πίνακας 7.2

7.2 Καθορισμός γενετικών παραμέτρων

Για να καθοριστεί η βέλτιστη διαμόρφωση γενετικού αλγορίθμου, δηλαδή οι γενιές, το μέγεθος του πληθυσμού, η διασταύρωση και ο ρυθμός μετάλλαξης, διεξήχθη ένα σύνολο πειραμάτων, στα οποία μεταβάλλεται κάθε φορά μία από τις προαναφερθείσες παραμέτρους, ενώ οι υπόλοιπες μένουν σταθερές. Αυτό περιγράφεται αναλυτικά στην δημοσίευση του παρακάτω συνδέσμου:

https://www.researchgate.net/publication/334382381_An_Enhanced_Genetic_Algorithm_for_Heterogeneous_Group_Formation_Based_on_Multi-Characteristics_in_Social-Networking-Based_Learning

Από αυτήν αντλήθηκαν οι απαραίτητες πληροφορίες, δηλαδή οι τιμές των παραμέτρων με βάση των οποίων έγιναν οι δοκιμές, καθώς επίσης και τα αντίστοιχα διαγράμματα που επιβεβαιώνουν τις βέλτιστες τιμές των παραμέτρων αυτών. Ο Πίνακας 7.3 παρουσιάζει το σύνολο των προαναφερθέντων αυτών παραμέτρων που χρησιμοποιούνται για τη δοκιμή, ενώ οι καλύτερες ρυθμίσεις που προέκυψαν από τα πειράματα σημειώνονται με έντονους χαρακτήρες.

ΠΑΡΑΜΕΤΡΟΙ ΓΙΑ ΔΟΚΙΜΕΣ

ΡΥΘΜΙΣΕΙΣ ΓΕΝΕΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ	ΠΕΙΡΑΜΑΤΙΚΕΣ ΤΙΜΕΣ
ΓΕΝΕΕΣ	50, 100, **150**, 200, 400, 800
ΜΕΓΕΘΟΣ ΠΛΗΘΥΣΜΟΥ	50, 80, **100**, 150
ΠΙΘΑΝΟΤΗΤΑ ΔΙΑΣΤΑΥΡΩΣΗΣ	60%, 70%, **75%**, 80%, 90%
ΠΙΘΑΝΟΤΗΤΑ ΜΕΤΑΛΛΑΞΗΣ	1%, 2%, 3%, **5%**, 7%

Πίνακας 7.3

Όλα τα πειράματα εκτελούνται στον ίδιο υπολογιστή, διασφαλίζοντας την αξιοποίηση των ίδιων δυνατοτήτων υλικού. Τα κριτήρια αξιολόγησης που χρησιμοποιήθηκαν ήταν ο υπολογιστικός χρόνος σε λεπτά και η βέλτιστη τιμή καταλληλόλητας, καθώς είναι τα κύρια χαρακτηριστικά που καθορίζουν την απόδοση του αλγορίθμου. Στο πρόβλημα της

ομαδοποίησής μας, ο στόχος είναι να ελαχιστοποιήσουμε την τιμή της συνάρτησης καταλληλότητας σε σύντομο χρονικό διάστημα.

Καθώς ο γενετικός αλγόριθμος είναι μια στοχαστική μέθοδος, που σημαίνει ότι κάθε εκτέλεση μπορεί να δώσει διαφορετικό αποτέλεσμα, κάθε σενάριο αξιολόγησης εκτελέστηκε 20 φορές και ο μέσος όρος των χαρακτηριστικών αξιολόγησης χρησιμοποιήθηκε για τη συγκριτική ανάλυση. Στη βιβλιογραφία, τα πειράματα εκτελούνται κυρίως περίπου 10 έως 30 φορές [18], [20], [22]. Ωστόσο, η επιλογή του αριθμού είναι μάλλον ad hoc. Σε αυτή την περίπτωση, επιλέγεται ο αριθμός των 20 εκτελέσεων, καθώς είναι αποδεκτός αριθμός εκτελέσεων, που χρησιμοποιούνται κυρίως στη βιβλιογραφία, δίνοντας αξιόπιστα αποτελέσματα. Επιπλέον, αυτός ο αριθμός εκτελέσεων είναι εφικτός να πραγματοποιηθεί στα πειράματα λαμβάνοντας υπόψη το εύρος των γενετικών παραμέτρων που πρέπει να αξιολογηθούν και τα δεδομένα του δείγματος. Στην εικόνα 7.2 απεικονίζονται τα αποτελέσματα των πειραμάτων.

Στον παρακάτω πίνακα (7.4) παρατίθεται κώδικας σε γλώσσα Python, ο οποίος ζητάει από τον χρήστη να εισάγει τις θεμελιώδης παραμέτρους του γενετικού αλγορίθμου που όπως προαναφέρθηκαν είναι οι γενιές, το μέγεθος του πληθυσμού, η διασταύρωση και ο ρυθμός μετάλλαξης, ελέγχοντας τις τιμές τους σύμφωνα με τον πίνακα 7.3. Εάν ο χρήστης δεν δώσει κάποια τιμή έγκυρη, τότε το πρόγραμμα ζητάει ξανά να εισάγει εκ νέου τιμή, μέχρις ότου να είναι έγκυρη.

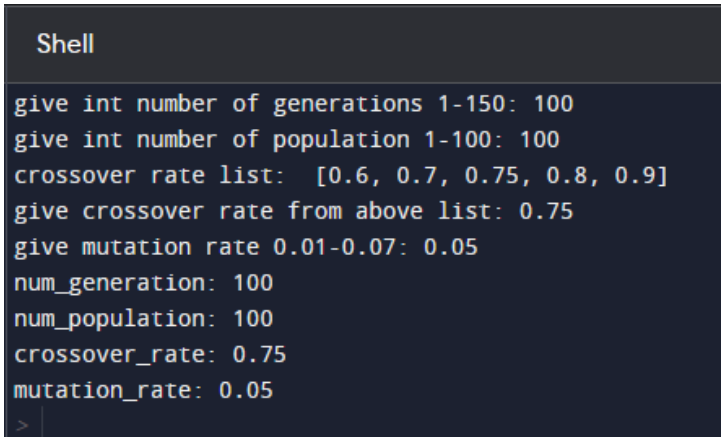
ΚΩΔΙΚΑΣ PYTHON: ΡΥΘΜΙΣΗ ΠΑΡΑΜΕΤΡΩΝ ΓΕΝΕΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ ΑΠΟ ΧΡΗΣΤΗ

```
while True:
   num_generation = input("give int number of generations 1-150: ")
   try: num_generation = int(num_generation)
   except ValueError: continue
   if num_generation > 150 or num_generation < 1: continue
   break
while True:
   num_population = input("give int number of population 1-100: ")
   try: num_population = int(num_population)
   except ValueError: continue
   if num_population > 100 or num_population < 1: continue
   break
list_crossover = [0.6, 0.7, 0.75, 0.8, 0.9]
while True:
```

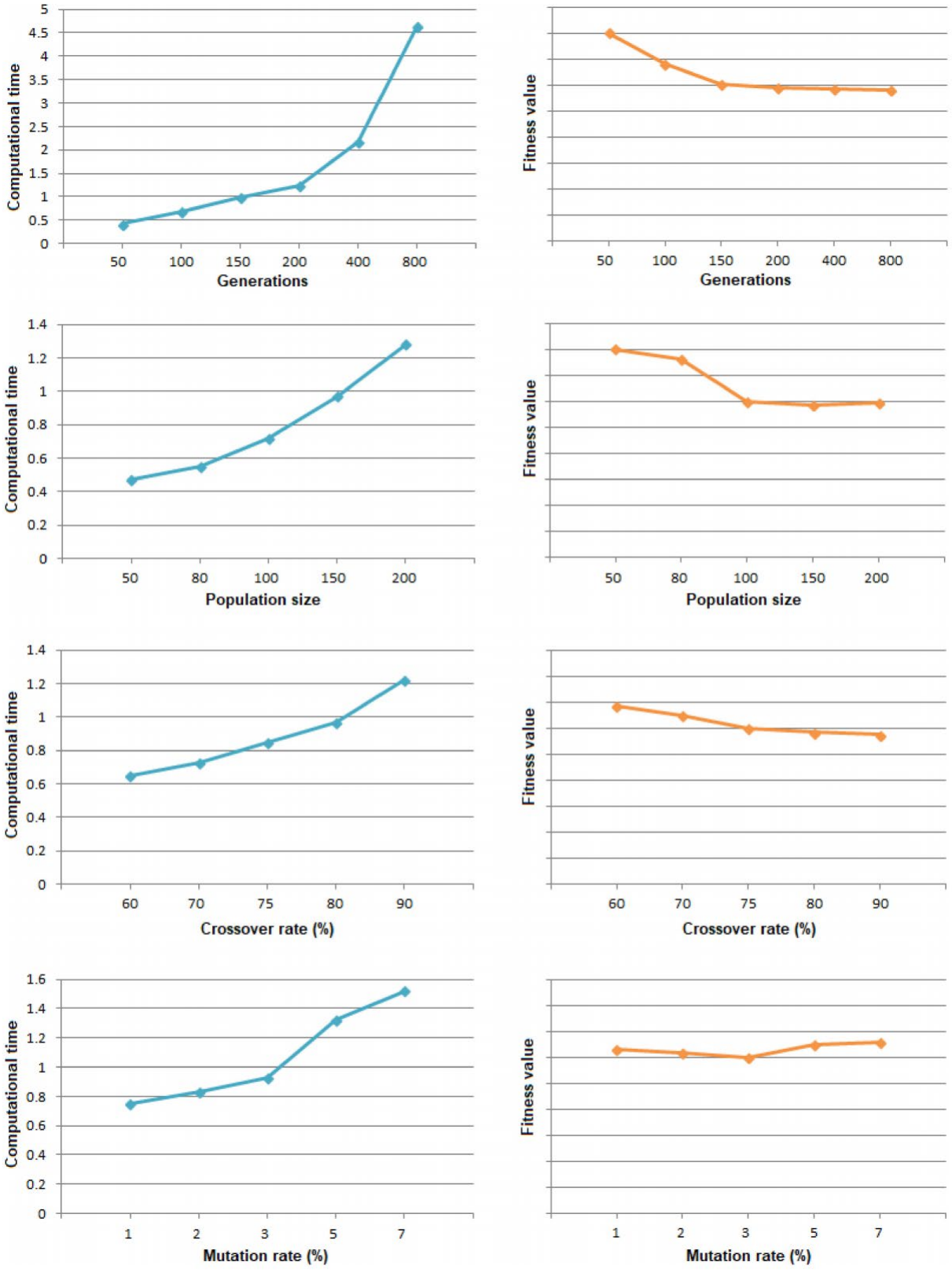
```
    print("crossover rate list: ",list_crossover)
    crossover_rate = input("give crossover rate from above list: ")
    try: crossover_rate = float(crossover_rate)
    except ValueError: continue
    if crossover_rate not in list_crossover: continue
    break
while True:
    mutation_rate = input("give mutation rate 0.01-0.07: ")
    try: mutation_rate = float(mutation_rate)
    except ValueError: continue
    if mutation_rate > 0.07 or mutation_rate < 0.01: continue
    break
print("num_generation:", num_generation)
print("num_population:", num_population)
print("crossover_rate:", crossover_rate)
print("mutation_rate:", mutation_rate)
```

Πίνακας 7.4

Δίνοντας για τις θεμελιώδης αυτές παραμέτρους τις τιμές 100, 100, 0.75 και 0.05 αντίστοιχα, ο παραπάνω κώδικας τις εκχωρεί στις αντίστοιχες μεταβλητές num_generation, num_population, crossover_rate και mutation_rate, όπως εμφανίζονται στο programiz και φαίνεται στην εικόνα 7.1.



Εικόνα 7.1


Computational time
Generations
Fitness value
Generations
Computational time
Population size
Fitness value
Population size
Computational time
Crossover rate (%)
Fitness value
Crossover rate (%)
Computational time
Mutation rate (%)
Fitness value
Mutation rate (%)

Εικόνα 7.2

Στην εικόνα 7.2 παρουσιάζονται συνολικά οκτώ διαγράμματα υπολογιστικού χρόνου και συνάρτησης καταλληλόλητας ως προς τις παραμέτρους του γενετικού αλγορίθμου ομαδοποίησης (γενεές, μέγεθος πληθυσμού, πιθανότητα διασταύρωσης, πιθανότητα μετάλλαξης). Όταν αυξάνεται ο αριθμός των γενεών, βελτιώνεται η αξία (η τιμή της συνάρτησης) καταλληλόλητας, ενώ αυξάνεται και ο χρόνος εκτέλεσης. Ωστόσο, θα πρέπει να σημειωθεί ότι η τιμή της συνάρτησης καταλληλόλητας φαίνεται να μη βελτιώνεται σημαντικά όταν οι γενιές είναι περισσότερες από 150 και ο χρόνος εκτέλεσης είναι σε σχετικά φυσιολογικό όριο λαμβάνοντας υπόψη τις περισσότερες επαναλήψεις. Έτσι, προτιμάται ένας αριθμός γενεών γύρω στις 150.

Όσον αφορά το μέγεθος του πληθυσμού, παρατηρείται ότι η τιμή των 100 χρωμοσωμάτων είναι η βέλτιστη επιλογή. Κάτω από αυτό το μέγεθος, παρά τον μικρότερο χρόνο υπολογισμού, η τιμή της συνάρτησης καταλληλόλητας είναι πολύ χειρότερη. Από την άλλη πλευρά, από ένα τέτοιο σημείο και πάνω, υπάρχει μόνο μια οριακή διαφορά μεταξύ των τιμών της συνάρτησης καταλληλόλητας, ενώ ο υπολογιστικός χρόνος αυξάνεται αισθητά.

Μεταβάλλοντας την πιθανότητα διασταύρωσης, ο αλγόριθμος συμπεριφέρεται καλά όταν είναι μεγαλύτερη από 75%. Ωστόσο, λόγω της δραματικής αύξησης του χρόνου εκτέλεσης και της μικρής αλλαγής της τιμής καταλληλόλητας όταν το ποσοστό διασταύρωσης είναι πολύ υψηλό, το ποσοστό 75% επιλέγεται ως η βέλτιστη πιθανότητα διασταύρωσης.

Η μετάλλαξη φαίνεται να μην επηρεάζει σημαντικά την απόδοση του αλγορίθμου. Ωστόσο, τα αποτελέσματα του πειράματος δείχνουν ότι υψηλότερο ποσοστό μετάλλαξης από 5% δεν βελτιώνει την τιμή καταλληλόλητας, ενώ αυξάνει τον υπολογιστικό χρόνο. Έτσι, η πιθανότητα μετάλλαξης για το πρόβλημα ομαδοποίησης έχει οριστεί στο 5% ως η καλύτερη επιλογή.

Συνοψίζοντας, η διαμόρφωση του γενετικού αλγορίθμου παίζει σημαντικό ρόλο στην απόδοση του αλγορίθμου, επηρεάζοντας την παραγωγή μιας βέλτιστης λύσης σε σύντομο χρονικό διάστημα. Μετά από μια σειρά πειραμάτων, συμπεραίνουμε ότι για την προτεινόμενη μέθοδο οι καλύτερες ρυθμίσεις είναι: 150 ως ο αριθμός των γενεών, 100 ως μέγεθος πληθυσμού, 75% ως πιθανότητα διασταύρωσης και 5% ως ποσοστό μετάλλαξης. Χρησιμοποιώντας αυτές τις ρυθμίσεις, διεξήχθησαν περαιτέρω πειράματα προκειμένου να συγκριθεί η αποτελεσματικότητα της μεθόδου μας με αυτή άλλων προσεγγίσεων γενετικού αλγορίθμου.

7.3 Γενετικός αλγόριθμος ομαδοποίησης σε γλώσσα προγραμματισμού Python

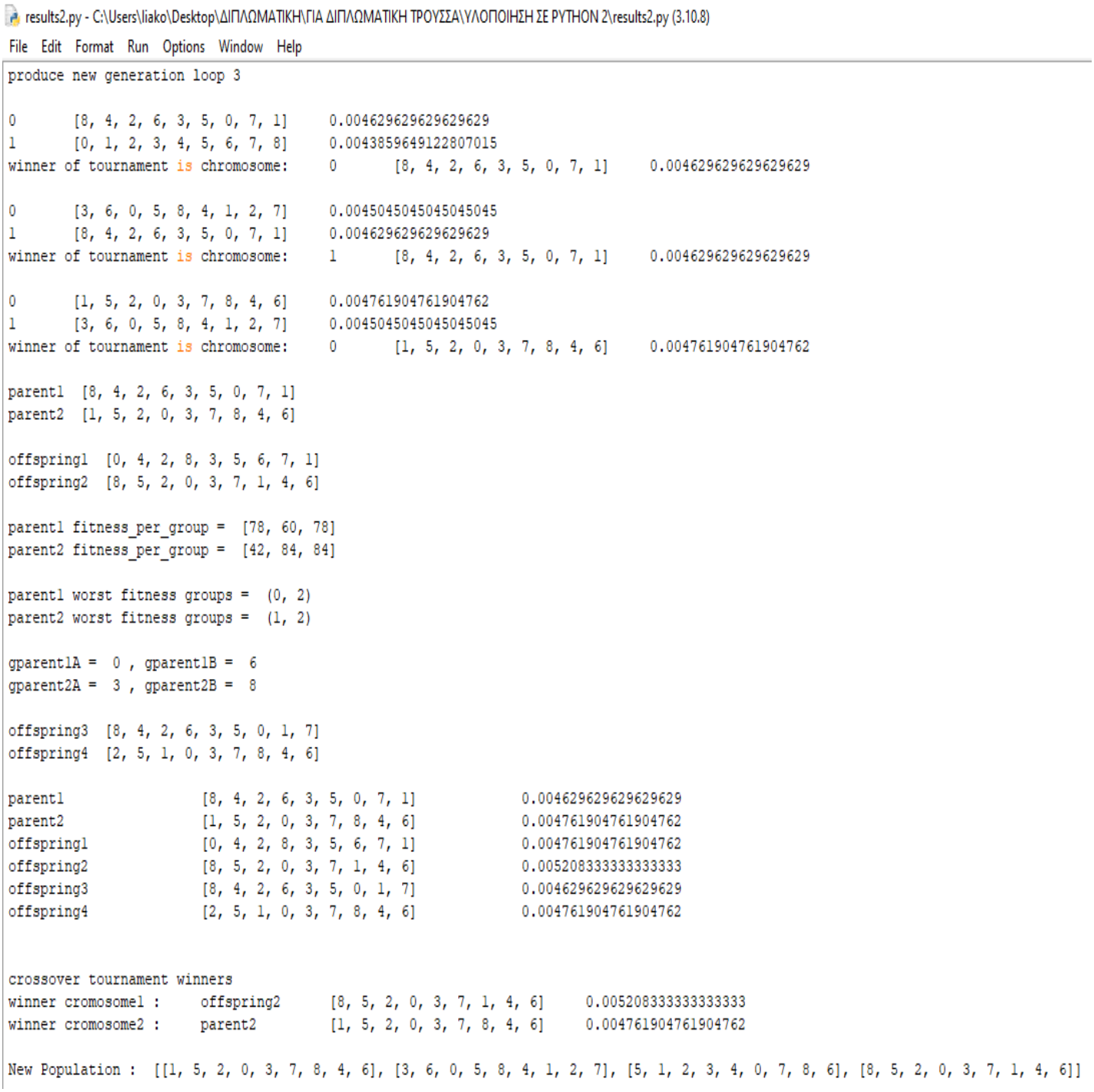
Παρακάτω παρουσιάζονται αποτελέσματα κώδικα αφού δημιουργήσει δύο γενεές πέντε χρωμοσωμάτων, με crossover rate 0,75 και mutation rate 0,05.

```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\ggasolution2.py
0       (2, 3, 5, 5, 4, 2, 3, 2, 1, 1, 1, 1, 1, 2, 1, 2, 3)
1       (3, 5, 3, 1, 2, 1, 2, 3, 2, 1, 1, 1, 1, 1, 2, 3, 1)
2       (4, 3, 3, 2, 3, 3, 2, 1, 3, 2, 1, 1, 1, 1, 3, 2, 2)
3       (5, 1, 1, 1, 1, 2, 2, 1, 4, 2, 1, 2, 1, 1, 2, 1, 1)
4       (4, 1, 3, 4, 3, 3, 1, 3, 2, 2, 1, 2, 1, 3, 1, 1, 3)
5       (1, 1, 2, 5, 4, 3, 3, 2, 3, 1, 2, 2, 2, 3, 2, 3, 3)
6       (4, 3, 1, 3, 1, 2, 2, 4, 2, 1, 2, 2, 1, 2, 1, 3, 2)
7       (4, 5, 5, 1, 3, 2, 1, 1, 1, 1, 1, 2, 2, 3, 2, 1, 1)
8       (3, 2, 2, 4, 5, 5, 2, 4, 1, 2, 1, 2, 2, 2, 1, 3, 1)

give int number of generations 1-150: 2
give int number of population 1-100: 5
crossover rate list:  [0.6, 0.7, 0.75, 0.8, 0.9]
give crossover rate from above list: 0.75
give mutation rate 0.01-0.07: 0.05
num_generation: 2
num_population: 5
crossover_rate: 0.75
mutation_rate: 0.05


0       [0, 1, 2, 3, 4, 5, 6, 7, 8]
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]
2       [3, 6, 0, 5, 8, 4, 1, 2, 7]
3       [2, 6, 4, 1, 8, 0, 5, 7, 3]
4       [1, 5, 2, 0, 3, 7, 8, 4, 6]
3       (0, 1, 2)

(0, 3, 6)

generation 1
chromosome      0       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
chromosome      1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
chromosome      2       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
chromosome      3       [2, 6, 4, 1, 8, 0, 5, 7, 3]     0.004273504273504274
chromosome      4       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762


elitism chromosomes
chromosome      0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

produce new generation loop 1

0       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

0       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
winner of tournament is chromosome:     1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045

parent1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2  [3, 6, 0, 5, 8, 4, 1, 2, 7]

New Population :  [[1, 5, 2, 0, 3, 7, 8, 4, 6], [3, 6, 0, 5, 8, 4, 1, 2, 7]]
```

Εικόνα 7.3

```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
produce new generation loop 2

0       [2, 6, 4, 1, 8, 0, 5, 7, 3]     0.004273504273504274
1       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
winner of tournament is chromosome:     1       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015

0       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

parent1  [0, 1, 2, 3, 4, 5, 6, 7, 8]
parent2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

offspring1  [0, 1, 2, 3, 4, 5, 6, 7, 8]
offspring2  [0, 5, 2, 1, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [84, 72, 72]
parent2 fitness_per_group =  [42, 84, 84]

parent1 worst fitness groups =  (0, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  1 , gparent1B =  4
gparent2A =  5 , gparent2B =  8

offspring3  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring4  [0, 1, 2, 3, 5, 7, 8, 4, 6]

parent1                 [0, 1, 2, 3, 4, 5, 6, 7, 8]             0.0043859649122807015
parent2                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring1              [0, 1, 2, 3, 4, 5, 6, 7, 8]             0.0043859649122807015
offspring2              [0, 5, 2, 1, 3, 7, 8, 4, 6]             0.003787878787878788
offspring3              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring4              [0, 1, 2, 3, 5, 7, 8, 4, 6]             0.003968253968253968


crossover tournament winners
winner cromosome1 :     offspring3      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     parent2         [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

New Population :  [[1, 5, 2, 0, 3, 7, 8, 4, 6], [3, 6, 0, 5, 8, 4, 1, 2, 7], [5, 1, 2, 3, 4, 0, 7, 8, 6]]
```

Εικόνα 7.4

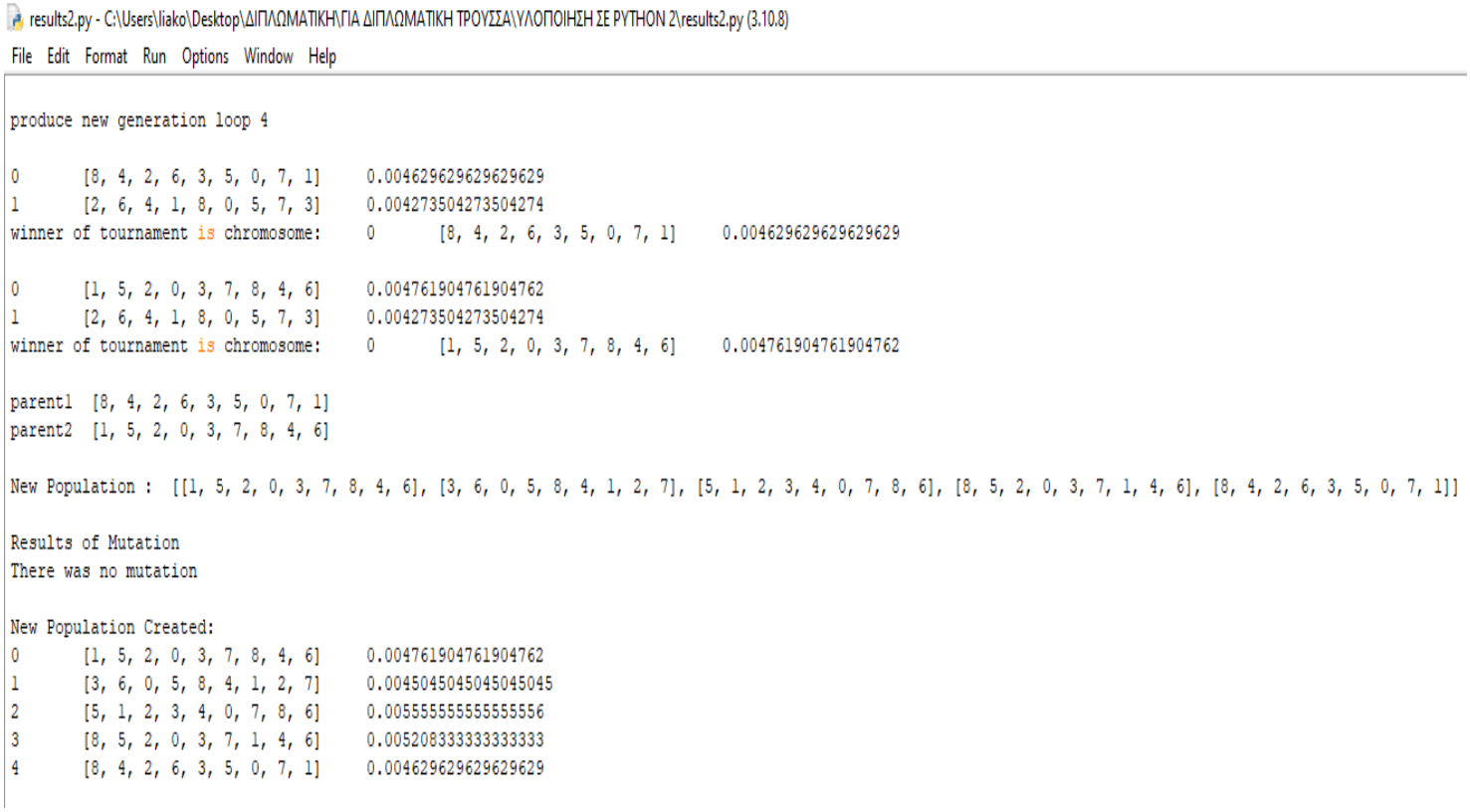


results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 3

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [0, 1, 2, 3, 4, 5, 6, 7, 8]     0.0043859649122807015
winner of tournament is chromosome:     0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629

0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
winner of tournament is chromosome:     0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

parent1  [8, 4, 2, 6, 3, 5, 0, 7, 1]
parent2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

offspring1  [0, 4, 2, 8, 3, 5, 6, 7, 1]
offspring2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

parent1 fitness_per_group =  [78, 60, 78]
parent2 fitness_per_group =  [42, 84, 84]

parent1 worst fitness groups =  (0, 2)
parent2 worst fitness groups =  (1, 2)

gparent1A =  0 , gparent1B =  6
gparent2A =  3 , gparent2B =  8

offspring3  [8, 4, 2, 6, 3, 5, 0, 1, 7]
offspring4  [2, 5, 1, 0, 3, 7, 8, 4, 6]

parent1                 [8, 4, 2, 6, 3, 5, 0, 7, 1]             0.004629629629629629
parent2                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring1              [0, 4, 2, 8, 3, 5, 6, 7, 1]             0.004761904761904762
offspring2              [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring3              [8, 4, 2, 6, 3, 5, 0, 1, 7]             0.004629629629629629
offspring4              [2, 5, 1, 0, 3, 7, 8, 4, 6]             0.004761904761904762


crossover tournament winners
winner cromosome1 :     offspring2      [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner cromosome2 :     parent2         [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

New Population :  [[1, 5, 2, 0, 3, 7, 8, 4, 6], [3, 6, 0, 5, 8, 4, 1, 2, 7], [5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6]]
```

Εικόνα 7.5

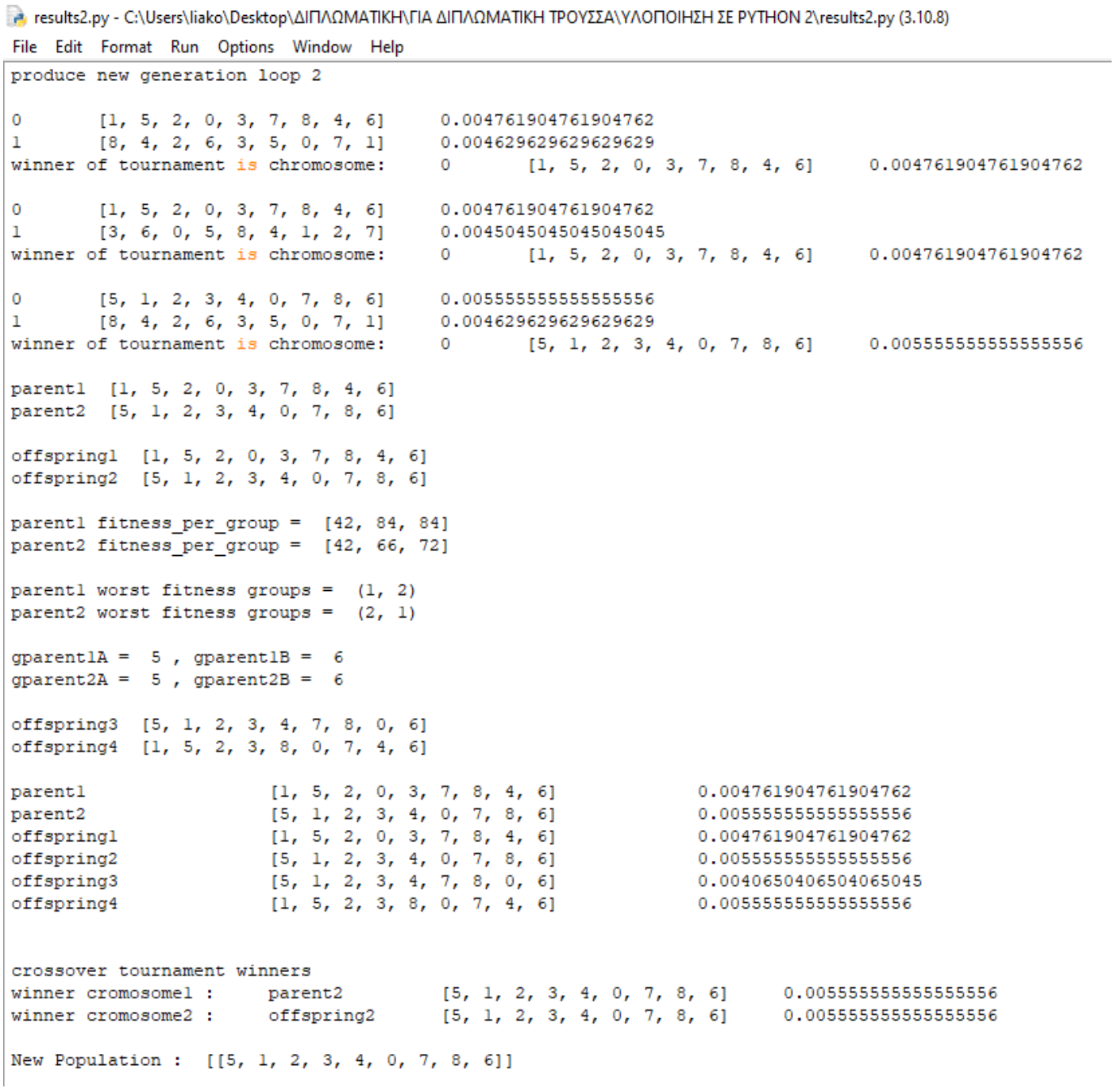


Εικόνα 7.6

Στις εικόνες 7.3 ως 7.6 έχουμε την δημιουργία πρώτης γενεάς 5 χρωμοσωμάτων με 9 φοιτητές (τυχαίες τιμές).

```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
generation 2
chromosome      0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
chromosome      1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
chromosome      2       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
chromosome      3       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
chromosome      4       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629


elitism chromosomes
chromosome      0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

produce new generation loop 1

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [42, 84, 84]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  5 , gparent1B =  6
gparent2A =  4 , gparent2B =  7

offspring3  [5, 1, 2, 3, 4, 7, 8, 0, 6]
offspring4  [1, 5, 2, 3, 4, 0, 7, 8, 6]

parent1                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 1, 2, 3, 4, 7, 8, 0, 6]             0.0040650406504065045
offspring4              [1, 5, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6]]
```

Εικόνα 7.7


results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File Edit Format Run Options Window Help
produce new generation loop 2
0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
1 [8, 4, 2, 6, 3, 5, 0, 7, 1] 0.004629629629629629
winner of tournament is chromosome: 0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
1 [3, 6, 0, 5, 8, 4, 1, 2, 7] 0.0045045045045045045
winner of tournament is chromosome: 0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
0 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
1 [8, 4, 2, 6, 3, 5, 0, 7, 1] 0.004629629629629629
winner of tournament is chromosome: 0 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
parent1 [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring1 [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent1 fitness_per_group = [42, 84, 84]
parent2 fitness_per_group = [42, 66, 72]
parent1 worst fitness groups = (1, 2)
parent2 worst fitness groups = (2, 1)
gparent1A = 5 , gparent1B = 6
gparent2A = 5 , gparent2B = 6
offspring3 [5, 1, 2, 3, 4, 7, 8, 0, 6]
offspring4 [1, 5, 2, 3, 8, 0, 7, 4, 6]
parent1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
offspring1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
offspring3 [5, 1, 2, 3, 4, 7, 8, 0, 6] 0.0040650406504065045
offspring4 [1, 5, 2, 3, 8, 0, 7, 4, 6] 0.005555555555555556
crossover tournament winners
winner cromosome1 : parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
winner cromosome2 : offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
New Population : [[5, 1, 2, 3, 4, 0, 7, 8, 6]]

Εικόνα 7.8

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 3

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

parent1  [8, 4, 2, 6, 3, 5, 0, 7, 1]
parent2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

offspring1  [0, 4, 2, 8, 3, 5, 6, 7, 1]
offspring2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

parent1 fitness_per_group =  [78, 60, 78]
parent2 fitness_per_group =  [42, 84, 84]

parent1 worst fitness groups =  (0, 2)
parent2 worst fitness groups =  (1, 2)

gparent1A =  0 , gparent1B =  8
gparent2A =  4 , gparent2B =  6

offspring3  [8, 4, 2, 6, 3, 5, 0, 7, 1]
offspring4  [4, 2, 6, 5, 3, 7, 8, 0, 1]

parent1                 [8, 4, 2, 6, 3, 5, 0, 7, 1]             0.004629629629629629
parent2                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring1              [0, 4, 2, 8, 3, 5, 6, 7, 1]             0.004761904761904762
offspring2              [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring3              [8, 4, 2, 6, 3, 5, 0, 7, 1]             0.004629629629629629
offspring4              [4, 2, 6, 5, 3, 7, 8, 0, 1]             0.004273504273504274


crossover tournament winners
winner cromosome1 :     offspring2      [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner cromosome2 :     parent2         [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

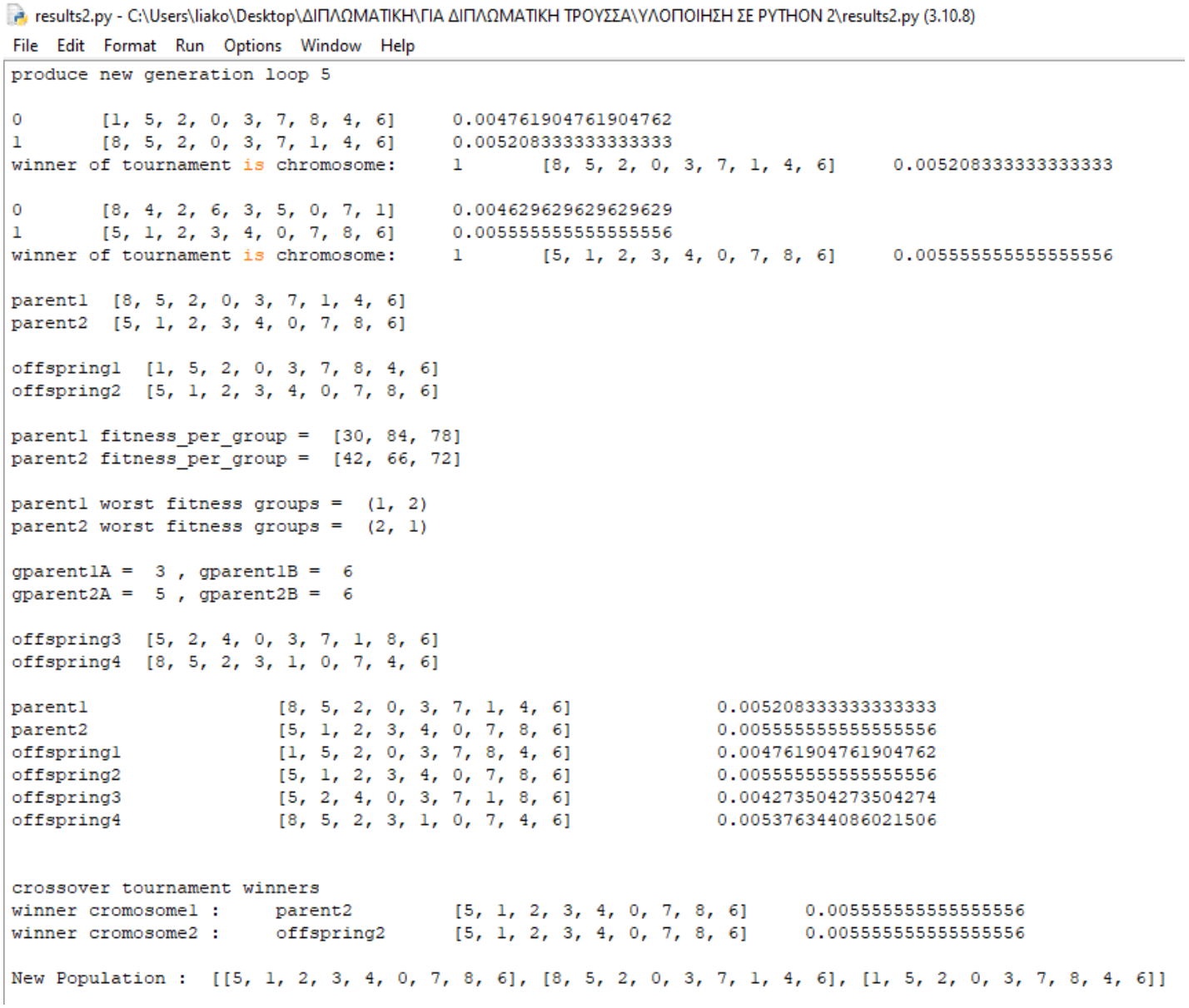
Εικόνα 7.9

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 4

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

offspring1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 66, 72]
parent2 fitness_per_group =  [30, 84, 78]

parent1 worst fitness groups =  (2, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  4 , gparent1B =  8
gparent2A =  4 , gparent2B =  6

offspring3  [5, 2, 3, 1, 4, 0, 7, 8, 6]
offspring4  [5, 2, 4, 0, 3, 7, 1, 8, 6]

parent1                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
parent2                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring1              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [5, 2, 3, 1, 4, 0, 7, 8, 6]             0.005555555555555556
offspring4              [5, 2, 4, 0, 3, 7, 1, 8, 6]             0.004273504273504274


crossover tournament winners
winner cromosome1 :     parent1         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring1      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.10



```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
produce new generation loop 5

0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [30, 84, 78]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  3 , gparent1B =  6
gparent2A =  5 , gparent2B =  6

offspring3  [5, 2, 4, 0, 3, 7, 1, 8, 6]
offspring4  [8, 5, 2, 3, 1, 0, 7, 4, 6]

parent1                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 2, 4, 0, 3, 7, 1, 8, 6]             0.004273504273504274
offspring4              [8, 5, 2, 3, 1, 0, 7, 4, 6]             0.005376344086021506


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.11

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 6

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

offspring1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 66, 72]
parent2 fitness_per_group =  [30, 84, 78]

parent1 worst fitness groups =  (2, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  3 , gparent1B =  8
gparent2A =  3 , gparent2B =  7

offspring3  [5, 2, 1, 3, 4, 0, 7, 8, 6]
offspring4  [5, 2, 8, 0, 3, 7, 1, 4, 6]

parent1                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
parent2                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring1              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [5, 2, 1, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring4              [5, 2, 8, 0, 3, 7, 1, 4, 6]             0.005208333333333333


crossover tournament winners
winner cromosome1 :     parent1         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring1      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.12

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 7

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [30, 84, 78]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  3 , gparent1B =  8
gparent2A =  4 , gparent2B =  7

offspring3  [5, 2, 8, 0, 3, 7, 1, 4, 6]
offspring4  [5, 2, 3, 1, 4, 0, 7, 8, 6]

parent1                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 2, 8, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring4              [5, 2, 3, 1, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

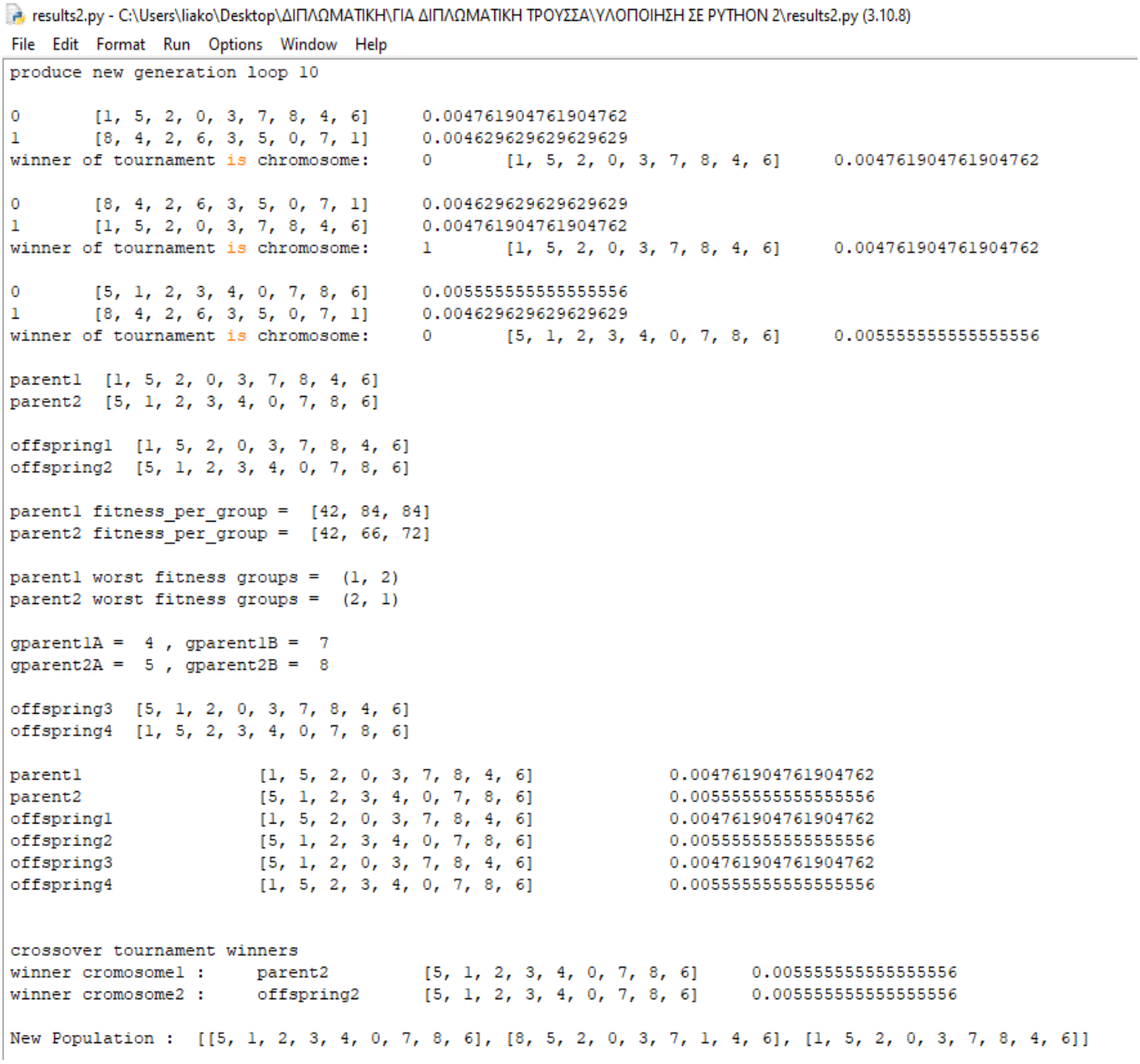
Εικόνα 7.13

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 8

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]

produce new generation loop 9

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [30, 84, 78]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  5 , gparent1B =  8
gparent2A =  3 , gparent2B =  7

offspring3  [5, 2, 3, 0, 8, 7, 1, 4, 6]
offspring4  [5, 2, 1, 3, 4, 0, 7, 8, 6]

parent1                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 2, 3, 0, 8, 7, 1, 4, 6]             0.005208333333333333
offspring4              [5, 2, 1, 3, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.14


results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File Edit Format Run Options Window Help
produce new generation loop 10
0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
1 [8, 4, 2, 6, 3, 5, 0, 7, 1] 0.004629629629629629
winner of tournament is chromosome: 0 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
0 [8, 4, 2, 6, 3, 5, 0, 7, 1] 0.004629629629629629
1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
winner of tournament is chromosome: 1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
0 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
1 [8, 4, 2, 6, 3, 5, 0, 7, 1] 0.004629629629629629
winner of tournament is chromosome: 0 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
parent1 [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring1 [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent1 fitness_per_group = [42, 84, 84]
parent2 fitness_per_group = [42, 66, 72]
parent1 worst fitness groups = (1, 2)
parent2 worst fitness groups = (2, 1)
gparent1A = 4 , gparent1B = 7
gparent2A = 5 , gparent2B = 8
offspring3 [5, 1, 2, 0, 3, 7, 8, 4, 6]
offspring4 [1, 5, 2, 3, 4, 0, 7, 8, 6]
parent1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
offspring1 [1, 5, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
offspring3 [5, 1, 2, 0, 3, 7, 8, 4, 6] 0.004761904761904762
offspring4 [1, 5, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
crossover tournament winners
winner cromosome1 : parent2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
winner cromosome2 : offspring2 [5, 1, 2, 3, 4, 0, 7, 8, 6] 0.005555555555555556
New Population : [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]

Εικόνα 7.15

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 11

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [30, 84, 78]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  4 , gparent1B =  8
gparent2A =  3 , gparent2B =  7

offspring3  [5, 2, 0, 8, 3, 7, 1, 4, 6]
offspring4  [5, 2, 1, 3, 4, 0, 7, 8, 6]

parent1                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 2, 0, 8, 3, 7, 1, 4, 6]             0.004273504273504274
offspring4              [5, 2, 1, 3, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.16

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 12

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

offspring1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 84, 84]
parent2 fitness_per_group =  [30, 84, 78]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (1, 2)

gparent1A =  4 , gparent1B =  7
gparent2A =  5 , gparent2B =  6

offspring3  [5, 2, 0, 1, 3, 7, 8, 4, 6]
offspring4  [5, 2, 0, 3, 8, 7, 1, 4, 6]

parent1                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
parent2                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring1              [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [5, 2, 0, 1, 3, 7, 8, 4, 6]             0.003787878787878788
offspring4              [5, 2, 0, 3, 8, 7, 1, 4, 6]             0.004273504273504274


crossover tournament winners
winner cromosome1 :     parent2         [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner cromosome2 :     offspring1      [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6]]
```

Εικόνα 7.17

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 13

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

offspring1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 66, 72]
parent2 fitness_per_group =  [30, 84, 78]

parent1 worst fitness groups =  (2, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  3 , gparent1B =  6
gparent2A =  3 , gparent2B =  7

offspring3  [8, 5, 2, 3, 4, 0, 7, 1, 6]
offspring4  [5, 2, 8, 0, 3, 7, 1, 4, 6]

parent1                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
parent2                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring1              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [8, 5, 2, 3, 4, 0, 7, 1, 6]             0.005952380952380952
offspring4              [5, 2, 8, 0, 3, 7, 1, 4, 6]             0.005208333333333333


crossover tournament winners
winner cromosome1 :     offspring3      [8, 5, 2, 3, 4, 0, 7, 1, 6]     0.005952380952380952
winner cromosome2 :     parent1         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]
```

Εικόνα 7.18

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 14

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

offspring1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 66, 72]
parent2 fitness_per_group =  [42, 84, 84]

parent1 worst fitness groups =  (2, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  4 , gparent1B =  8
gparent2A =  4 , gparent2B =  6

offspring3  [1, 5, 2, 3, 4, 0, 7, 8, 6]
offspring4  [5, 1, 2, 4, 3, 7, 8, 0, 6]

parent1                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
parent2                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring1              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [1, 5, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring4              [5, 1, 2, 4, 3, 7, 8, 0, 6]             0.0040650406504065045


crossover tournament winners
winner cromosome1 :     parent1         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring1      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]

produce new generation loop 15

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]
```

Εικόνα 7.19

```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
produce new generation loop 15

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [8, 5, 2, 0, 3, 7, 1, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]

produce new generation loop 16

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [42, 84, 84]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  5 , gparent1B =  6
gparent2A =  5 , gparent2B =  8

offspring3  [5, 1, 2, 3, 4, 7, 8, 0, 6]
offspring4  [1, 5, 2, 3, 4, 0, 7, 8, 6]

parent1                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 1, 2, 3, 4, 7, 8, 0, 6]             0.0040650406504065045
offspring4              [1, 5, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]
```

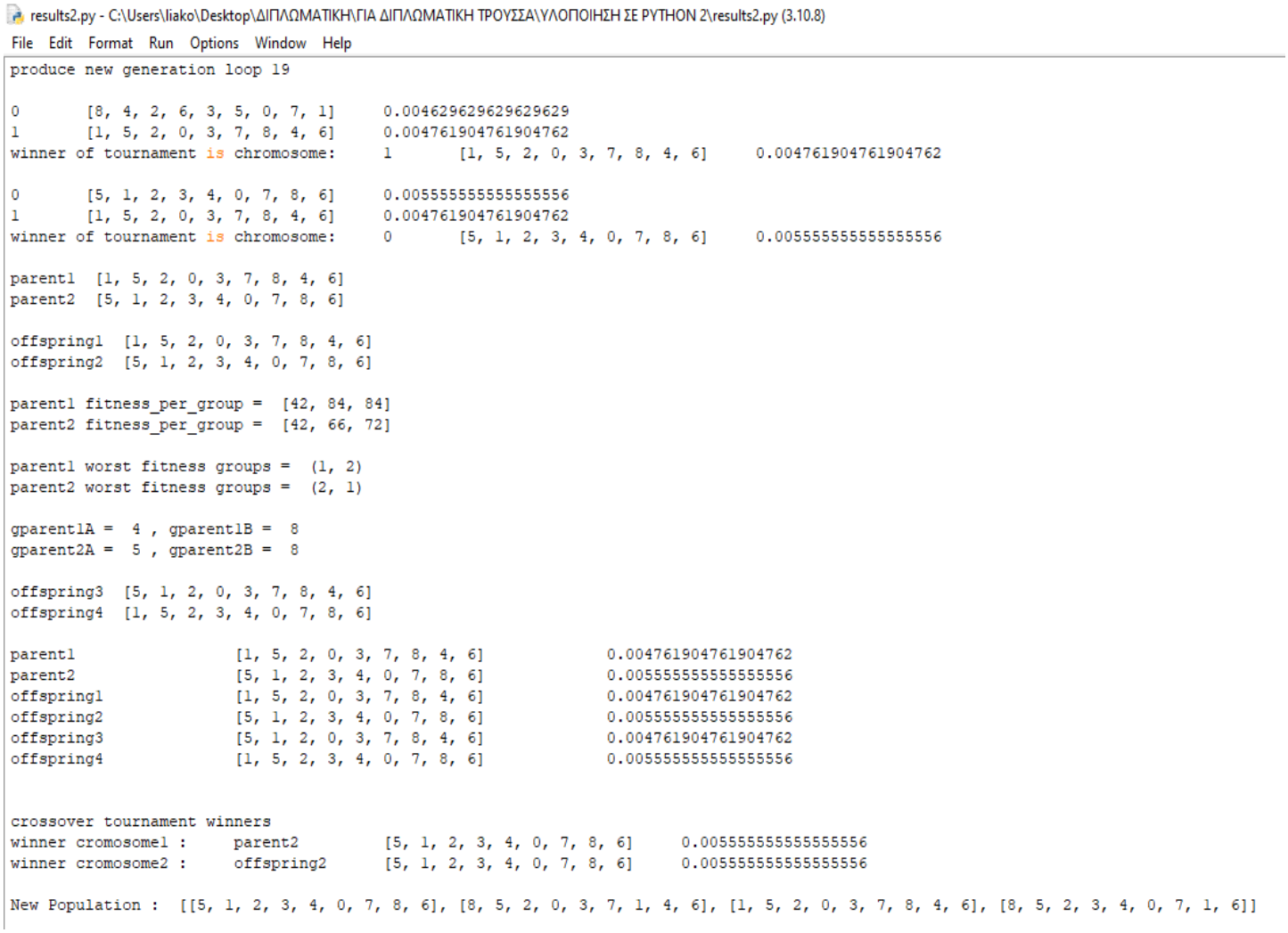
Εικόνα 7.20

results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)

File Edit Format Run Options Window Help

```
produce new generation loop 17

0       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

offspring1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
offspring2  [1, 5, 2, 0, 3, 7, 8, 4, 6]

parent1 fitness_per_group =  [42, 66, 72]
parent2 fitness_per_group =  [30, 84, 78]

parent1 worst fitness groups =  (2, 1)
parent2 worst fitness groups =  (1, 2)

gparent1A =  5 , gparent1B =  7
gparent2A =  4 , gparent2B =  8

offspring3  [5, 2, 3, 1, 4, 0, 7, 8, 6]
offspring4  [5, 2, 0, 8, 3, 7, 1, 4, 6]

parent1                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
parent2                 [8, 5, 2, 0, 3, 7, 1, 4, 6]             0.005208333333333333
offspring1              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring2              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring3              [5, 2, 3, 1, 4, 0, 7, 8, 6]             0.005555555555555556
offspring4              [5, 2, 0, 8, 3, 7, 1, 4, 6]             0.004273504273504274


crossover tournament winners
winner cromosome1 :     parent1         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring1      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]

produce new generation loop 18

0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner of tournament is chromosome:     1       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 5, 2, 0, 3, 7, 1, 4, 6]

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]
```

Εικόνα 7.21



```
results2.py - C:\Users\liako\Desktop\ΔΙΠΛΩΜΑΤΙΚΗ\ΓΙΑ ΔΙΠΛΩΜΑΤΙΚΗ ΤΡΟΥΣΣΑ\ΥΛΟΠΟΙΗΣΗ ΣΕ PYTHON 2\results2.py (3.10.8)
File  Edit  Format  Run  Options  Window  Help
produce new generation loop 19

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

parent1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
parent2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

offspring1  [1, 5, 2, 0, 3, 7, 8, 4, 6]
offspring2  [5, 1, 2, 3, 4, 0, 7, 8, 6]

parent1 fitness_per_group =  [42, 84, 84]
parent2 fitness_per_group =  [42, 66, 72]

parent1 worst fitness groups =  (1, 2)
parent2 worst fitness groups =  (2, 1)

gparent1A =  4 , gparent1B =  8
gparent2A =  5 , gparent2B =  8

offspring3  [5, 1, 2, 0, 3, 7, 8, 4, 6]
offspring4  [1, 5, 2, 3, 4, 0, 7, 8, 6]

parent1                 [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
parent2                 [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring1              [1, 5, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring2              [5, 1, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556
offspring3              [5, 1, 2, 0, 3, 7, 8, 4, 6]             0.004761904761904762
offspring4              [1, 5, 2, 3, 4, 0, 7, 8, 6]             0.005555555555555556


crossover tournament winners
winner cromosome1 :     parent2         [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
winner cromosome2 :     offspring2      [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6]]
```

Εικόνα 7.22

Στις εικόνες 7.7 ως 7.22 εμφανίζεται η δημιουργία δεύτερης γενεάς 5 χρωμοσωμάτων με 9 φοιτητές (τυχαίες τιμές).

```
produce new generation loop 20

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
winner of tournament is chromosome:     0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556

0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629
1       [3, 6, 0, 5, 8, 4, 1, 2, 7]     0.0045045045045045045
winner of tournament is chromosome:     0       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629

parent1  [5, 1, 2, 3, 4, 0, 7, 8, 6]
parent2  [8, 4, 2, 6, 3, 5, 0, 7, 1]

New Population :  [[5, 1, 2, 3, 4, 0, 7, 8, 6], [8, 5, 2, 0, 3, 7, 1, 4, 6], [1, 5, 2, 0, 3, 7, 8, 4, 6], [8, 5, 2, 3, 4, 0, 7, 1, 6], [8, 4, 2, 6, 3, 5, 0, 7, 1]]

Results of Mutation
There was no mutation

New Population Created:
0       [5, 1, 2, 3, 4, 0, 7, 8, 6]     0.005555555555555556
1       [8, 5, 2, 0, 3, 7, 1, 4, 6]     0.005208333333333333
2       [1, 5, 2, 0, 3, 7, 8, 4, 6]     0.004761904761904762
3       [8, 5, 2, 3, 4, 0, 7, 1, 6]     0.005952380952380952
4       [8, 4, 2, 6, 3, 5, 0, 7, 1]     0.004629629629629629


Best Chromosomes of Grouping Genetic Algorithm :
solution        1       [8, 5, 2, 3, 4, 0, 7, 1, 6]     0.005952380952380952
```

Εικόνα 7.23

Στην εικόνα 7.23 φαίνονται τα αποτελέσματα του γενετικού αλγόριθμού 5 χρωμοσωμάτων με 9 φοιτητές (τυχαίες τιμές) μετά από τη δημιουργία δύο γενεών.

7.4 Συμπεράσματα

Αυτή η διπλωματική εργασία παρουσίασε τον τρόπο με τον οποίο μπορούν να σχηματιστούν οι βέλτιστες συνεργατικές ομάδες εργασίας φοιτητών χρησιμοποιώντας έναν νέο γενετικό αλγόριθμο. Οι καινοτομίες της προτεινόμενης μεθόδου αφορούν τα χαρακτηριστικά βάσει των οποίων σχηματίστηκαν ετερογενείς ομάδες και οι γενετικοί τελεστές που εφαρμόστηκαν. Στη βιβλιογραφία, οι περισσότερες μελέτες, στον τομέα του σχηματισμού ομάδων φοιτητών χρησιμοποιώντας προσεγγίσεις γενετικού αλγορίθμου, επικεντρώνονται στην ομαδοποίηση με βάση τις επιδόσεις αυτών και εκτελούν έναν τελεστή διασταύρωσης.

Ο προτεινόμενος αλγόριθμος εκμεταλλεύεται τα χαρακτηριστικά που προέκυψαν από τις τρεις κύριες διαστάσεις της μάθησης, δηλαδή το ακαδημαϊκό, το γνωστικό και το κοινωνικό, προκειμένου να ενισχυθεί η συνεργασία και να βελτιωθούν τα μαθησιακά αποτελέσματα τόσο σε ομαδικό όσο και σε ατομικό επίπεδο. Επιπλέον, χρησιμοποιεί δύο χειριστές διασταύρωσης, δηλαδή μια τροποποίηση της διασταύρωσης 2 σημείων και μια νέα προσέγγιση που ονομάζεται διασταύρωση 1 σημείου ανά ομάδα, έτσι ώστε να μπορεί να

αναζητηθεί πιο διεξοδικά ο χώρος λύσεων του προβλήματος και να εισαχθούν νέες γενετικές πληροφορίες στον πληθυσμό, αποτρέποντας την σύγκλιση του αλγόριθμου σε ένα τοπικό ελάχιστο. Για τον προτεινόμενο αλγόριθμο ομαδοποίησης οι καλύτερες τιμές των παραμέτρων διαμόρφωσης είναι: 150 ως ο αριθμός των γενεών, 100 ως μέγεθος πληθυσμού, 75% ως πιθανότητα διασταύρωσης και 5% ως ποσοστό μετάλλαξης.

Κεφάλαιο 8

8. Πηγές – Αναφορές

Κεφαλαίου 1:

https://link.springer.com/chapter/10.1007/978-1-4613-3437-8_17

https://www.researchgate.net/publication/307863862_Comparative_Analysis_of_K-means_and_Genetic_Algorithm_based_Data_Clustering

Κεφαλαίου 2:

http://ikee.lib.auth.gr/record/128526/files/GRI-2012-8048.pdf (Κεφ. 2.4 Συνεργατική μάθηση και συστήματα σχηματισμού ομάδων)

Κεφαλαίου 3:

http://artemis.cslab.ece.ntua.gr:8080/jspui/bitstream/123456789/13036/1/DT2016-0016.pdf (ΚΕΦΑΛΑΙΟ 1 Γενετικοί Αλγόριθμοι)

[1] Charles Darwin, “The Origin of Species on the basis of Natural Selection”, 1859

[2] Stuart J.Russell and Peter Norvig, “Artificial Intelligence: A Modern

Aprroach”, Prentice Hall, 1995

[3] Melanie Mitchell, “An intro duction to genetic algorithms”, (5 t h edition) The MIT press, Cambridge, MA, 1999

[4] John H. Holland, “Adaptation in Natural and Artificial Systems”, Ann Arbor: University of Michigan Press, 1975

[5] Goldberg D.E., Genetic algorithms in search, o ptimizatio n and machine learning, Addison Wesley, Reading, 1989

[6] P. Fleming and R.C Purshouse, “Evolutionary algorithms in control systems engineering: a survey”, Control Engineering Practice, 10: pp1223-1241, 2002

[7] Lowen R. and Verschoren A., Mathematical Modelling: Theory and application, Foundations of generic o ptimization Volume2: Applications of fuzzy control, genetic algorithms and neural netw orks, Springer, 2008

[8] Kumar R., Novel Encoding Scheme in Genetic Algorithms for Better Fitness, Internatio nal Journal of Engineering and Advanced Technology, Vol.1 , 6, 2012

[9] Patvichaichod S., Application of hybrid encoding genetic algorithms on pickup and delivery traveling salesman problem with traffic conditions, Journal Computer Science, vol.7, 5 , pp605-610, 2011

[10] H.Kitano, “Designing neural networks using genetic algorithms with graph generation system, Co mplex Systems, 4, 1990

[11] F.Rothlauf, “Representations for Genetic and Evolutio nary Algorithms”, Springer, 2006

[12] Smith M.J., Evolutionary Genetics, Oxfo rd University Press, 19 98

[13] Haupt R.L. and Haupt S.E., “Practical genetic algorithms”, Wiley-Interscience, 2004

[14] K.Jebari and M. Madiafi, “Selection methods for Genetic Algorithms”, Internatio nal Journal of Emerginh Sciences 3, 4: pp333-334, 2013

[15] Baeck T., Evolutionary algorithms in theory and practice: evolution strategies, evolutionary pro gramming, genetic algorithms, Oxford University Press, New York, 1996

[16] Sivanadam S.N. and Deepa S.N., Introduction to Genetic Algorithms, Springer, 2008

[17] De Jong K.A., Spears W.M., “A formal analysis of the role of multi-point crossover in genetic algorithms”, Annals of Mathematics and Artificial Intelligence, 5, pp1 - 26, 1992

[18] De Jong K.A., Spears W.M., “An analysis of multi-point crossover”, Foundations of Genetic Algorithms, Morgan Kaufmann Publishers, 1991

[19] Yang S., “Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism, Artificial Life, 8, Massachusetts Institute of Technology, 2003

[20] Schaffer J.D, Caruna R.A., Eshelman L.J., Das R., “A study of control parameters affecting online performance of genetic algorithms for function optimization”, Proceeding of the 3 r d Internatio nal Conference on Genetic Algorithms and their applicatio ns, 1989

[21] Koumousis V.K. and Katsaras C., “The effect of population variatio n in genetic algorithms- The saw-tooth GA”, IEEE Transactions on Evolutionary Computation, 10, 1, pp19-28, 2006

[22] Vasconcelos J.A., Ramirez J.A., Takahashi R.H.C., Saldanha R.R., “Improvements in genetic algorithms”, Magnetics, IEEE Transactions, vol. 37, 5, pp3414-3417, 2001

[23] Λυκοθανάσης Σ., «Γενετικοί Αλγόριθμοι και Εφαρμογές», vol.3, ΕΑΠ, 2000

Κεφαλαίων 4,5,6,7:

https://www.researchgate.net/publication/334382381_An_Enhanced_Genetic_Algorithm_for_Heterogeneous_Group_Formation_Based_on_Multi-Characteristics_in_Social-Networking-Based_Learning

[7] R. Costaguta, "Algorithms and machine learning techniques in collaborative group formation," in Advances in Artificial Intelligence and Its Applications, O. Pichardo Lagunas, O. Herrera Alc_antara, and G. Arroyo Figueroa, Eds., Cham, Switzerland, Springer, Oct. 2015, pp. 249–258, doi:10.1007/978-3-319-27101-9_1.

[8] A. Sukstrienwong, "A genetic-algorithm approach for balancing learning styles and academic attributes in heterogeneous grouping of students," Int. J. Emerg. Technol. Learn., vol. 12, no. 3, pp. 4–25, Mar. 2017, doi:10.3991/ijet.v12i03.5803.

[12] R. M. Felder and R. Brent, "Effective strategies for cooperative learning," J. Cooperation Collaboration College Teaching, vol. 10, no. 2, pp. 69–75, Sept. 2001.

[13] W. M. Cruz and S. Isotani, "Group formation algorithms in collaborative learning contexts: A systematic mapping of the literature," in Collaboration and Technology, vol. 8658, N. Baloian, F. Burstein, H. Ogata, F. Santoro, and G. Zurita, Eds. Cham, Switzerland, Springer, Sep. 2014, pp. 199–214, doi:10.1007/978-3-319-10166-8_18.

[14] S. Sankaranarayanan, C. Dashti, C. Bogart, X. Wang, M. Sakr, and C. P. Ros_e, "When optimal team formation is a choice-self-selection versus intelligent team formation strategies in a large online project-based course," in Artificial Intelligence in Education, vol. 10947, C. Penstein Ros_e, Ed., Cham, Switzerland, Springer, Jun. 2018, pp. 518–531, doi:10.1007/978- 3-319-93843-1_38.

[15] M. Wen, K. Maki, S. Dow, J. D. Herbsleb, and C. Rose, "Supporting virtual team formation through community-wide deliberation," Proc. ACM Human-Comput. Interact., vol. 1, Nov. 2017, Art. no. 109, doi : 10.1145/ 3134744.

[16] J. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[17] E. Falkenauer, "The grouping genetic algorithms-widening the scope of the GAs,"Belg. J.Oper. Res.,Statist.Comput. Sci., vol. 33, no. 1, pp.79–102, 1992.

[18] J. Moreno, D. A. Ovalle, and R. M. Vicari, "A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics," Comput. Educ., vol. 58, no. 1, pp. 560–569, Jan. 2012, doi:10.1016/j.compedu.2011.09.011.

[19] Y. H. Wang, Y. C. Li, and H. C. Liao, "Using a genetic algorithm to determine optimal complementary learning clusters for ESL in Taiwan," Expert Syst. Appl., vol. 38, no. 12, pp. 14832–14837, Nov. 2011, doi:10.1016/j.eswa.2011.05.065.

[20] R. Contreras and P. Salcedo, "Genetic algorithms as a tool for structuring collaborative groups," Natural Comput., vol. 16, no. 2, pp. 231–239, June 2017, doi: 10.1007/s11047-016-9574-1.

[21] G. J. Hwang, P. Y. Yin, C. W. Hwang, and C. C. Tsai, "An enhanced genetic approach to composing cooperative learning groups for multiple grouping criteria," Educ. Technol. Soc., vol. 11, no. 1, pp. 148–167, Jan. 2008. [Online]. Available: https://www.j-ets.net/ets/journals/11_1/11.pdf

[22] Y. Zheng, C. Li, S. Liu, and W. Lu, "An improved genetic approach for composing optimal collaborative learning groups," Knowl.-Based Syst., vol. 139, pp. 214–225, Jan. 2018, doi: 0.1016/j.knosys.2017.10.022.

[23] A. Wichmann, T. Hecking, M. Elson, N. Christmann, T. Herrmann, and H. U. Hoppe, "Group formation for small-group learning: Are heterogeneous groups more productive?," in Proc. 12th Int. Symp. Open Collab., NewYork, NY, USA, Aug. 2016, pp. 1–4, doi: 10.1145/2957792.2965662.

[24] J. Bernacki and A. Kozierkiewicz-Hetma_nska, "The comparison of creating homogeneous and heterogeneous collaborative learning groups in intelligent tutoring systems," in Intelligent Information and Database Systems, N. Nguyen, B. Trawin´ski, and R. Kosala, Eds., Cham, Springer, 2015, pp. 46–55, doi: 10.1007/978-3-319-15702-3_5.

[25] I. Srba and M. Bielikova, "Dynamic group formation as an approach to collaborative learning support," IEEE Trans. Learn. Technol., vol. 8, no. 2, pp. 173–186, Apr. 2015, doi: 10.1109/TLT.2014.2373374.

[26] E. Alfonseca, R. M. Carro, E. Mart_ın, A. Ortigosa, and P. Paredes, "The impact of learning styles on student grouping for collaborative learning: A case study," User Model. User-Adapted Interact., vol. 16, nos. 3/4, pp. 377–401, Sep. 2006, doi: 10.1007/s11257-006-9012-7.

[27] N. Devasenathipathi and K. M. Nilesh, "Contemplating crossover operators of genetic algorithm for student group formation problem," Int. J. Emerg. Technol. Adv. Eng., vol. 2, no. 2, pp. 192–197, Feb. 2012. [Online].

Available: https://ijetae.com/files/Volume2Issue2/IJETAE_0212_31.pdf

[28] S. Yadav and A. Soha, “Comparative study of different selection techniques in genetic algorithm,” Int. J. Eng., Sci. Math., vol. 6, no. 3, pp. 174–180, July 2017. [Online]. Available: http://www.ijesm.co.in/past-articles.php?issueid0362

[29] N. Soni and T. Kumar, “Study of various mutation operators in genetic algorithms,” Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 3, pp. 4519–4521, Jun. 2014. [Online]. Available: http://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit20140503404.pdf