Εργασία Υπολογιστική Νοημοσύνη: Μέρος Β'

Ιωάννα Γώγου – ΑΜ1054388 – gogou@ceid.upatras.gr

Χρησιμοποιήθηκαν οι εξής βιβλιοθήκες της Python: ***numpy, scipy, sklearn.metrics, deap***

Οι κώδικες της άσκησης βρίσκονται στο παρακάτω link:
https://github.com/joangog/genalgoproj

Β1. Σχεδιασμός ΓΑ

α) Κωδικοποίηση:

Με βάση τις υποθέσεις της εκφώνησης, θα θεωρήσουμε την δυαδική κωδικοποίηση. Θεωρούμε ότι κάθε βαθμολογία αναπαρίσταται από την δυαδική της κωδικοποίηση σε 3 bit. Χρειαζόμαστε σταθερό αριθμό bit για να μπορέσουμε να κάνουμε εύκολα αποκωδικοποίηση αν χρειαστεί, και συγκεκριμένα, τουλάχιστον 3 bit για να αναπαραστήσουμε τους ακέραιους αριθμού 1 έως 5.

β) Πλεονάζουσες Τιμές:

Χρησιμοποιώντας δυαδική κωδικοποίηση είναι γεγονός ότι κατά την μετάλλαξη μπορεί να αναστραφούν bit ή να διασταυρωθούν δυο άτομα δημιουργώντας έτσι δεκαδικές τιμές πάνω από 5 ή ίσες με 0 (π.χ. α) 1**0**1 -> 1**1**1, β) 1|00 + 0|11 -> 111 και 000). Μια λύση για να αντιμετωπιστεί αυτό θα ήταν να αποκωδικοποιούμε την συμβολοσειρά και να ελέγχουμε αν κάποια τιμή είναι εκτός των ορίων. Αν είναι, την αντικαθιστούμε με κάποια τυχαία εντός διαστήματος και επανακωδικοποιούμε. **Βέβαια αυτό είναι αρκετά κοστοβόρο και επομένως θα μπορούσαμε να μην κάνουμε κωδικοποίηση αλλά να θεωρήσουμε τους ακεραίους 1 έως 5 σαν σύμβολα του αντίστοιχου αριθμού.** Αυτό μας συμφέρει διότι οι αριθμοί μας είναι ακέραιοι και έτσι αποφεύγουμε και τις μη έγκυρες τιμές. Για την μετάλλαξη θα μεταλλάσσουμε κάποιο ψηφίο σε ένα τυχαίο στο σύνολο {1-5}. Με την διασταύρωση δεν θα αντιμετωπίσουμε πρόβλημα.

γ) Αρχικός Πληθυσμός

Αρχικά όσον αφορά τις τιμές για τις ταινίες που δεν έχει δει ο χρήστης, δεν μπορούμε να επιλέξουμε την μέση τιμή. Το να επιλέξουμε την μέση βαθμολογία του χρήστη δημιουργεί μόνο ένα άτομο στον πληθυσμό καθώς δίνει μόνο ένα πιθανό διάνυσμα βαθμολογιών. Το ίδιο ισχύει με το να συμπληρώσουμε 0 σε όλες τις κενές βαθμολογίες. Επομένως πρέπει με κάποιο τρόπο να δημιουργήσουμε διαφορετικούς συνδυασμούς βαθμολογιών στις κενές θέσεις του διανύσματος. **Θα μπορούσαμε να παράγουμε τυχαίες βαθμολογίες** δημιουργώντας διάφορους συνδυασμούς που θα αποτελέσουν τα άτομα του πληθυσμού μας.

Όσον αφορά τις τιμές των ταινιών που έχει δει, θα τις συμπεριλάβουμε στον πληθυσμό διότι θα χρειαστούν στην συνέχεια για να υπολογίσουμε την συνάρτηση καταλληλότητας. Παρόλα αυτά θα παραμένουν σταθερές διότι γνωρίζουμε τις πραγματικές τιμές και δεν είναι σωστό να βρούμε λύση η οποία δεν ανταποκρίνεται στις επιλογές του χρήστη. Αν αλλάξουν οι γνωστές βαθμολογίες, τότε χάνεται το νόημα της προσέγγισης των πιθανών επιλογών του χρήστη διότι οι τιμές που προέβλεψε ο

αλγόριθμος, επειδή αξιολογήθηκαν με βάση την απόσταση ολόκληρου του διανύσματος από τους γείτονες του, θα έχουν προκύψει από μη έγκυρα δεδομένα.

δ) Διαδικασία Επιδιόρθωσης:

Ας αξιολογήσουμε τις προτεινόμενες μεθόδους. Αρχικά, η αντικατάσταση της μη νόμιμης λύσης με άλλη δεν αποτελεί απαραίτητα λύση στο πρόβλημα μας. Στην περίπτωση της τυχαίας αντικατάστασης, αν αντικατασταθεί η μη νόμιμη λύση με κάποια τυχαία, αυτό δεν εξασφαλίζει ότι η τυχαία λύση θα είναι καλή. Στην περίπτωση του ελιτισμού, εάν ένα νόμιμο άτομο του πληθυσμού φέρει στην επόμενη γενιά ένα παράνομο άτομο, τότε απορρίπτουμε το νέο άτομο και το αντικαθιστούμε με το καλύτερο άτομο της προηγούμενης γενιάς. Έτσι ο αλγόριθμος δεν χρειάζεται να ανακαλύψει πάλι καλές λύσεις που εμφανίστηκαν σε παλιές γενιές. Βέβαια, στην περίπτωση που στην νέα γενιά προκύψουν πάρα πολλές παράνομες λύσεις, θα αναγκαστούμε να διατηρήσουμε πολλές λύσεις τις προηγούμενης γενιάς, με αποτέλεσμα μικρή ποικιλομορφία και αργή σύγκλιση.

Με την μέθοδο εφαρμογής ποινής ναι μεν μειώνεται η πιθανότητα να επιλεγεί η παράνομη λύση, αλλά εξακολουθεί να υπάρχει αυτή η πιθανότητα. Στην περίπτωση που επιλεγεί, θα μπορούσε να αναπαράγει νέες παράνομες λύσεις κάτι που δεν είναι επιθυμητό διότι σπαταλάμε θέσεις στον πληθυσμό με παράνομες λύσεις οι οποίες γνωρίζουμε εκ των προτέρων ότι είναι παράνομες και θα μπορούσαμε να τις είχαμε απορρίψει εξαρχής. Δηλαδή παίρνουν την ευκαιρία από άλλες νόμιμες λύσεις να επιλεγούν. Αυτό κάνει τον αλγόριθμο να συγκλίνει πιο αργά σε μια επιθυμητή λύση. Μια διαδικασία εφαρμογής ποινής για τέτοιες λύσεις θα ήταν να απομονώναμε τα ψηφία που γνωρίζουμε από την λύση και να τα αποθηκεύαμε σε ένα διάνυσμα. Στην συνέχεια θα μετρήσουμε την ομοιότητα (Ευκλείδεια απόσταση) του διανύσματος αυτού με το διάνυσμα των πραγματικών τιμών. Η ποινή που θα εφαρμόσουμε θα είναι ανάλογη της απόστασης αυτής και αφού την πολλαπλασιάσουμε με κάποιο βάρος της επιλογής μας θα την αφαιρέσουμε από την συνάρτηση καταλληλότητας. Έτσι αν η λύση απέχει πολύ από την πραγματική τότε θα έχει πολύ χαμηλή καταλληλότητα.

Επομένως μία καλή λύση είναι η επιδιόρθωση διότι γνωρίζουμε εκ των προτέρων τις πραγματικές τιμές οπότε δεν υπάρχει λόγος να μην απορρίπτουμε τις παράνομες λύσεις επιτόπου. Επίσης, επειδή αντικαθιστούμε μόνο το κομμάτι της λύσης που γνωρίζουμε, το υπόλοιπο κομμάτι του ατόμου θα έχει ικανοποιητική ποικιλομορφία για να συγκλίνουμε με επαρκή ταχύτητα στην λύση.

ε) Εύρεση Γειτονιάς Χρήστη:

Αρχικά η μετρική Pearson είναι προτιμότερη από τις υπόλοιπες μετρικές διότι είναι ανεξάρτητη από την απόσταση των συγκρινόμενων διανυσμάτων αλλά εξαρτάται από τις διακυμάνσεις των στοιχείων τους γύρω από την μέση τιμή τους. Οι υπόλοιπες μετρικές δεν έχουν την ιδιότητα αυτή διότι συγκρίνουν τα διανύσματα ως προς την απόσταση (Manhattan,Ευκλείδεια) και την κατεύθυνση (cosine). Επομένως εάν ένας χρήστης βαθμολογεί συχνά χαμηλά και ένας άλλος συχνά υψηλά αυτοί οι δύο έχουν σχετικά μεγάλη ομοιότητα διότι η διακύμανση των βαθμολογιών που δίνουν είναι του ίδιου μεγέθους, παρόλο που οι τιμές που δίνουν είναι πολύ διαφορετικές. Αυτό είναι κάτι επιθυμητό διότι όπως είχε αναφερθεί και στο ερώτημα α) του Μέρους Α’ της εργασίας, ο κάθε χρήστης μπορεί να είναι γενικά αυστηρός ή επιεικής. Εμάς δεν μας αφορούν οι προσωπικές τάσεις στο πως βαθμολογεί αλλά οι διακυμάνσεις γύρω από την βαθμολογία που βάζει συνήθως. Για παράδειγμα αν βαθμολογεί συχνά με 2 και βαθμολογήσει μια ταινία με 5 τότε αυτή η ενέργεια έχει μεγάλη διαφορά από το να την βαθμολογούσε με 5 κάποιος χρήστης που βαθμολογεί συχνά με 5 και τις υπόλοιπες ταινίες.

Θα επιλέξουμε να γεμίσουμε τις ελλιπείς τιμές με τον μέσο όρο βαθμολογίας της ταινίας από κάθε χρήστη ώστε η επιλογή αυτή να μην επηρεάσει την κατάταξη του. Δεν προτιμούμε το 0 γιατί επηρεάζει την κατάταξή του. Δεν επιλέγουμε να αγνοήσουμε τις ελλείψεις διότι τα δεδομένα μας είναι σχετικά αραιά και δεν θα μπορέσουμε να υπολογίσουμε σωστά την απόσταση για κάποιους χρήστες που έχουν βαθμολογήσει πολύ λίγες ταινίες. Αν το κάναμε αυτό, θα περιοριζόμασταν μόνο στις ταινίες που δεν έχουν nan για έναν από τους δυο συγκρινόμενους χρήστες, ακόμα κι αν ο δεύτερος έχει τιμή στις βαθμολογίες αυτές, με αποτέλεσμα να «πετάμε» και υπαρκτή πληροφορία.

στ) Συνάρτηση Καταλληλότητας:

Συμπεραίνουμε ότι ο συντελεστής ομοιότητας Pearson παίρνει τιμές κοντά στο 1 αν υπάρχει πλήρης ομοιότητα στον τρόπο βαθμολογίας των χρηστών, 0 αν δεν υπάρχει καμία ομοιότητα και -1 αν οι χρήστες βαθμολογούν εντελώς αντίθετα. Δεν είναι απαραίτητο να κλιμακωθεί η τιμή διότι και αρνητικές τιμές να πάρει γνωρίζουμε ότι όσο πιο χαμηλή είναι η μετρική τόσο πιο ανεπιθύμητη είναι μια λύση. Επίσης χρησιμοποιούμαι τον μέσο όρο και όχι το άθροισμα των μετρικών Pearson ως συνάρτηση καταλληλότητας, οπότε οι αρνητικές τιμές δεν τοποθετούν «τιμωρία» στην καταλληλότητα μας και επομένως δεν προκαλούν πρόβλημα στην αξιολόγηση. **Βέβαια, για λόγους καλύτερης επεξήγησης και οπτικοποιήση των γραφικών παραστάσεων θα κλιμακώσουμε την συνάρτηση στο διάστημα [0,1].**

ζ) Γενετικοί Τελεστές:

i) Επιλογή:

Αρχικά τόσο η επιλογή με ρουλέτα όσο και η επιλογή με βάση την κατάταξη δίνουν μεγαλύτερη πιθανότητα επιλογής στις λύσεις με υψηλή καταλληλότητα. Όμως η διαφορά τους είναι ότι στην ρουλέτα η πιθανότητα υπολογίζεται με βάση την καθαρή τιμή της συνάρτησης καταλληλότητας ενώ στην δεύτερη η πιθανότητα υπολογίζεται από την γενική κατάταξη της λύσης ως προς την καταλληλότητα. Επομένως η επιλογή με βάση την κατάταξη αγνοεί το μέγεθος των διαφορών στην καταλληλότητα και δεν επηρεάζεται από το εάν μια λύση είναι πολύ καταλληλότερη από κάποια άλλη. Αυτό δεν είναι επιθυμητό στην περίπτωσή μας διότι αν μια λύση είναι πολύ πιο κοντά στους γείτονες από τις υπόλοιπες, κάλλιστα επιθυμούμε να την επιλέξουμε. Περαιτέρω, μόλις βρούμε μια λύση που είναι κοντά στους γείτονες του χρήστη, δεν έχει νόημα να απομακρυνθούμε πολύ από αυτήν, μόνο να πλησιάζουμε την γειτονιά όλο και περισσότερο. Οπότε οι λύσεις που έχουν καλή αξιολόγηση πρέπει να «ενθαρρύνονται» στο αλγόριθμο, δηλαδή να επιλέγονται πάντα οι καλύτερες λύσεις. Δεδομένου αυτού, δεν θα επιλέξουμε ως μέθοδο επιλογής την ρουλέτα ή την επιλογή βάση κατάταξης διότι και στις δύο, ακόμα και η χειρότερη λύση μπορεί να επιλεγεί έστω και με μικρή πιθανότητα. Σε κάποια προβλήματα αυτό είναι χρήσιμο διότι ελπίζουμε ότι μέσω κάποιας διασταύρωσης ή μετάλλαξης μπορεί να δώσει καλή αξιολόγηση και να μας βοηθήσει να ξεφεύγουμε από τοπικά ελάχιστα. Όμως στο πρόβλημά μας ο στόχος μας είναι να ελαχιστοποιήσουμε την απόσταση του χρήστη από όλους τους γείτονές του. Οι λύσεις στο πρόβλημα αυτό είναι πολλές αλλά κινούνται στην ίδια περιοχή του χώρου λύσεων οπότε δεν έχουμε μεγάλη ανάγκη να περιηγηθούμε σε άλλες περιοχές με την ελπίδα ότι θα βρούμε νέα περιοχή καλών λύσεων. **Δεδομένου αυτού, η καταλληλότερη επιλογή είναι το τουρνουά.** Σε αυτήν την περίπτωση, όποιες λύσεις και αν επιλεγούν για το τουρνουά, ποτέ δεν θα επιλεγεί η χειρότερη μεταξύ αυτών, έστω κι αν δεν είναι η καλύτερη από όλο τον πληθυσμό. Έτσι ο αλγόριθμος συγκλίνει σχετικά πιο γρήγορα διότι δεν δίνει την ευκαιρία σε κακές λύσεις να περάσουν στις επόμενες γενιές.

ii) Διασταύρωση:

Αρχικά, όσο πιο λίγα σημεία διασταύρωση έχουμε τόσο πιο πολύ διατηρούμε λύσεις προηγούμενων γενεών, ενώ αυξάνοντας τα σημεία αυξάνουμε την πιθανότητα να εμφανιστούν εντελώς καινούργιες λύσεις. Επομένως η διασταύρωση ενός σημείου είναι πολύ καλή στο να διατηρεί καλές λύσεις του παρελθόντος, αλλά δεν καταφέρνει να εξερευνήσει τον χώρο τον λύσεων όσο καλά όσο η περίπτωση την ομοιόμορφης διασταύρωσης όπου εξαιτίας της μάσκας που χρησιμοποιούμε μπορούμε να έχουμε σημείο διασταύρωσης ακόμα και σε κάθε σημείο του ατόμου. Άρα η διασταύρωση μονού σημείου συγκλίνει γρήγορα αλλά κινδυνεύει πιο πολύ από τοπικά ελάχιστα, ενώ η ομοιόμορφη διασταύρωση συγκλίνει πιο αργά αλλά μπορεί να ξεφύγει από τοπικά ελάχιστα. Η διασταύρωση διπλού σημείου είναι μια μέση λύση (πιο κοντά στην πρώτη λύση) που συνδυάζει τα πλεονεκτήματα και των δύο. Οι διασταυρώσεις OX και PMX δεν είναι χρήσιμες για το πρόβλημά μας διότι αφορούν προβλήματα μεταθέσεων. Επομένως, δεδομένων των παραπάνω και της απάντησής μας στο ερώτημα i, ενδιαφερόμαστε περισσότερο να συγκλίνουμε γρήγορα σε κάποια καλή λύση καθώς στο πρόβλημά μας οι καλές λύσεις βρίσκονται στην ίδια περιοχή. Δεν μας νοιάζει τόσο να εξερευνήσουμε τον χώρο, **και γι' αυτό επιλέγουμε διασταύρωση ενός σημείου.**

iii) Μετάλλαξη:

Σε αντίθεση με την διασταύρωση όπου αξιοποιούμαι κομμάτι από ήδη υπαρκτές καλές λύσεις, στην μετάλλαξη τα άτομα που δημιουργούνται παράγονται τυχαία και δεν γνωρίζουμε αν θα οδηγήσει αυτό σε καλή αξιολόγηση. Επομένως βγάζει νόημα να αντιγράφουμε πανομοιότυπα κάποια N καλύτερα άτομα της προηγούμενης γενιάς στην επόμενη (ελιτισμός) έτσι ώστε να μην χαθούν οι καλές λύσεις και στα υπόλοιπα να εφαρμόζουμε τους τελεστές του αλγορίθμου μας. Έτσι εξασφαλίζουμε ότι ακόμα και αν λόγω μετάλλαξης δημιουργηθούν άτομα με πολύ κακή αξιολόγηση, στον πληθυσμό θα έχουμε πάντα αυτά τα N καλύτερα αποτελέσματα των προηγούμενων γενεών. **Επομένως στον αλγόριθμό μας θα χρησιμοποιήσουμε τον ελιτισμό με N=1 για να κρατάμε μόνο τον πρώτο καλύτερο στις επόμενες γενιές.**

B2. Υλοποίηση ΓΑ

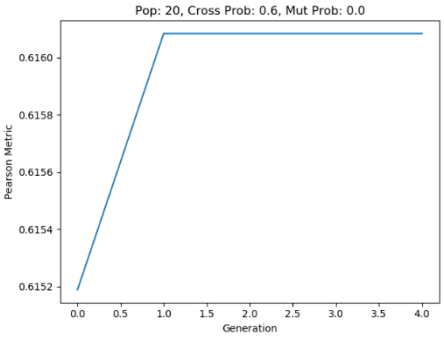
Ο κώδικας βρίσκεται στο αρχείο "gen_algo.py". Ο επιλεγμένος χρήστης είναι ο 12.

B3. Αξιολόγηση και Επίδραση Παραμέτρων

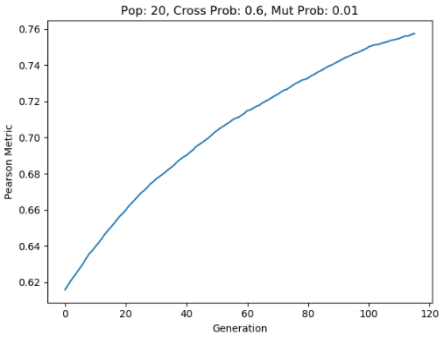
α) Αρχικά υλοποιήσαμε και τις τρεις συνθήκες τερματισμού με περιθώριο τερματισμού 5 γενιές, ελάχιστο δέλτα βελτίωσης 0.0001 και μέγιστο αριθμό γενεών 1000.

	Μέγεθος Πληθυσμού	Πιθανότητα Διασταύρωσης	Πιθανότητα Μετάλλαξης	Μέση Τιμή Βέλτιστου	Μέσος Αριθμός Γενεών
1	20	0.6	0	0.6161	4
2	20	0.6	0.01	0.7554	115
3	20	0.6	0.1	0.6708	24
4	20	0.9	0.01	0.7617	121
5	20	0.1	0.01	0.7523	117
6	200	0.6	0	0.6497	11
7	200	0.6	0.01	0.8189	129
8	200	0.1	0.01	0.8112	145
9	200	0.9	0.01	0.8238	127

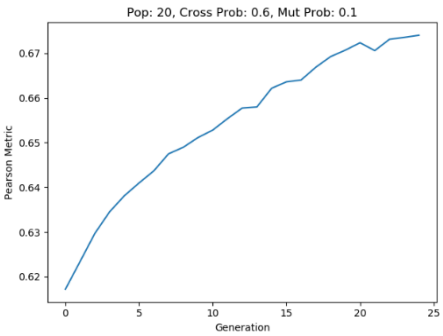
β) Τα διαγράμματα περιλαμβάνουν τιμές μόνο για τον μέσο όρο του πλήθους των γενεών που έτρεξε ο αλγόριθμος σε κάθε επανάληψη.



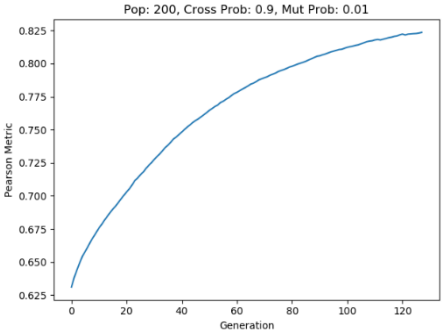
Εικόνα 1



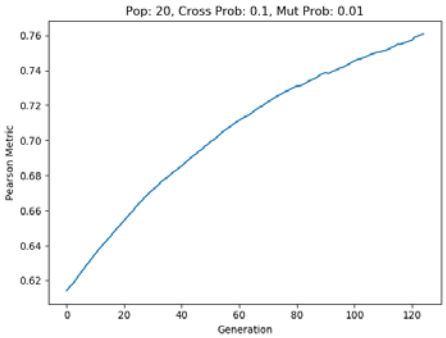
Εικόνα 2



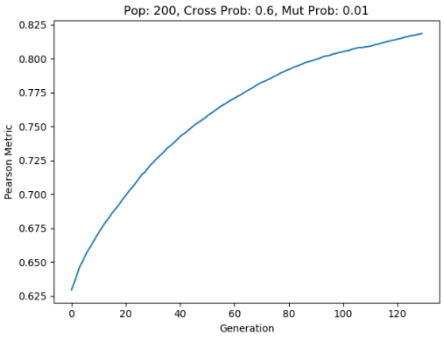
Εικόνα 3



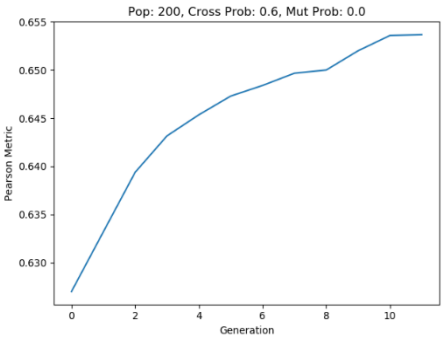
Εικόνα 4



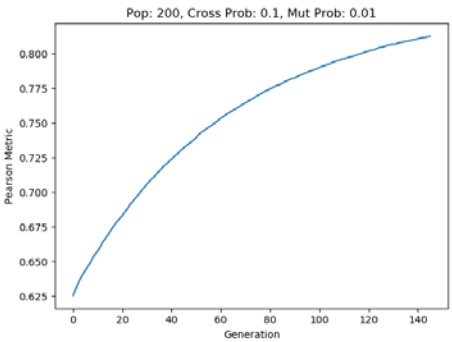
Εικόνα 5



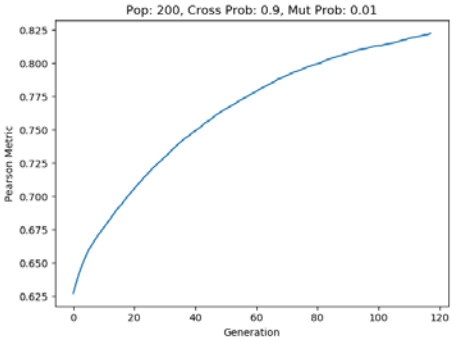
Εικόνα 6



Εικόνα 7



Εικόνα 8



Εικόνα 9

γ) Αρχικά, όσον αφορά την επιρροή του μεγέθους του πληθυσμού συμπεραίνουμε τα εξής. Μεγαλύτερο μέγεθος πληθυσμού σε μερικές περιπτώσεις αύξησε ελάχιστα το πλήθος γενεών για την σύγκλιση του αλγορίθμου (Εικόνες 5,8) ενώ σε άλλες έμεινε σταθερό (Εικόνες 4,9 και 2,6). Αυτό συμβαίνει διότι η αλλαγή στον πληθυσμό δεν έχει σημαντικό αντίκτυπο στο πόσο μεγάλες θα είναι οι μεταβολές στην βελτίωση ανά γενιά καθώς και στις δύο περιπτώσεις μπορούν να υπάρξουν στον πληθυσμό πολύ καλές λύσεις. Η μόνη διαφορά είναι ότι αυξάνοντας τον πληθυσμό δίνουμε στον αλγόριθμο της ευκαιρία να εξερευνήσει πιο πολλές λύσεις και έτσι έχει μεγαλύτερη πιθανότητα να βρει τις καλύτερες. Γι' αυτό τον λόγο παρατηρούμε σημαντική αύξηση στην απόδοση με τα 200 άτομα σε σχέση με τα 20.

Παρατηρούμε ότι η επιρροή της πιθανότητας διασταύρωσης δεν έχει ουσιαστική επίδραση στην σύγκλιση του αλγορίθμου (Εικόνες 2,4,5 και 6,8,9) με ποικίλα σκαμπανεβάσματα χωρίς να ακολουθούν κάποιο μοτίβο. Αυτό συμβαίνει διότι η διασταύρωση δεν παράγει αρκετά μεγάλο βαθμό ποικιλομορφίας όσο η μετάλλαξη καθώς διατηρούμε πάνω κάτω τα χαρακτηριστικά των προηγούμενων γενεών. Επίσης επειδή χρησιμοποιούμε ελιτισμό, η καλύτερα λύση της προηγούμενης γενιάς διατηρείται, οπότε εάν η καλύτερη λύση βρεθεί σχετικά νωρίς στον αλγόριθμο, η διασταύρωση μπορεί να παράγει λύσεις που δεν μπορούν να την ξεπεράσουν καθώς δεν προκαλούν δραματικές αλλαγές στα χαρακτηριστικά του πληθυσμού. Έτσι η απόδοση θα βελτιώνεται με τον ίδιο ρυθμό ανεξάρτητα από τις διασταυρώσεις που θα γίνουν και επομένως εναλλαγή στην πιθανότητα αυτή δεν επιφέρει γρηγορότερη σύγκλιση. Για τον ίδιο λόγο δεν παρατηρείται καμία βελτίωση και στην τελική απόδοση.

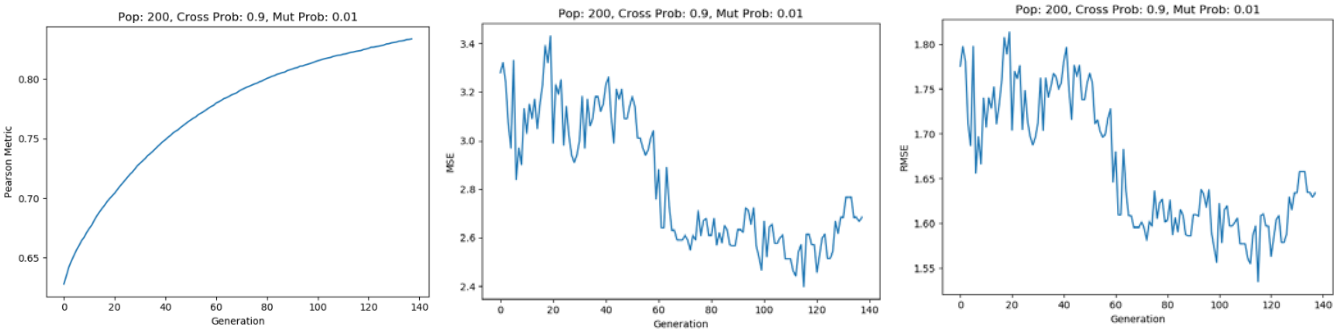
Τέλος, θα εξετάσουμε την επιρροή της πιθανότητας μετάλλαξης. Όσο πιο μικρή (Εικόνα 1) είναι η πιθανότητα, τόσο λιγότερες είναι οι γενιές που χρειάζονται για να συγκλίνει ο αλγόριθμος. Αυτό συμβαίνει διότι ο αλγόριθμος αδυνατεί να παράγει αρκετά ποικιλόμορφες λύσεις καθώς με μικρή πιθανότητα μετάλλαξης οι λύσεις παράγονται κατά κανόνα μέσω της διασταύρωσης παλιών λύσεων η διατήρησης τους. Έτσι εξερευνεί πολύ αργά τον χώρο των λύσεων με αποτέλεσμα η βελτίωση στην απόδοση να είναι πολύ μικρή. Επομένως συγκλίνει μετά από πολύ λίγες γενιές καταλήγοντας σε τοπικό ελάχιστο. Αυξάνοντας την πιθανότητα (Εικόνα 2), ο αλγόριθμος εξερευνεί νέες λύσεις οι οποίες δεν συνδέονται με την προηγούμενη γενιά σε αντίθεση με τον τελεστή διασταύρωσης. Έτσι έχει την δυνατότητα να βρει πρωτόγνωρες, καλύτερες λύσεις βελτιώνοντας την απόδοση. Επομένως συγκλίνει σε περισσότερες γενιές διότι παρατηρεί συνεχώς μεγάλες βελτιώσεις στην απόδοση και συνεχίζει να τρέχει. Παρόλα αυτά, αν η πιθανότητα μετάλλαξης είναι πολύ μεγάλη (Εικόνα 3) τότε η τελική απόδοση του αλγορίθμου δεν είναι καλύτερη και συγκλίνει πάλι σε λίγες γενιές. Αυτό συμβαίνει διότι τα νέα άτομα που προκύπτουν είναι εντελώς τυχαία χωρίς να συγκρατούν τις καλές ιδιότητες των παλιών γενιών. Έτσι ο αλγόριθμος δυσκολεύεται να βρει λύσεις που βελτιώνουν την απόδοση διότι βαδίζει τυχαία στον χώρο και τελικά συγκλίνει όταν η βελτίωση γίνει ελάχιστη.

Συμπερασματικά, το μοντέλο το οποίο έδωσε τις καλύτερες τιμές ήταν το 9^{ο}, το οποίο έδωσε την υψηλότερη τιμή μέσης απόδοσης και την μικρότερη μέση τιμή γενεών σε σχέση με λύσεις αντίστοιχης απόδοσης.

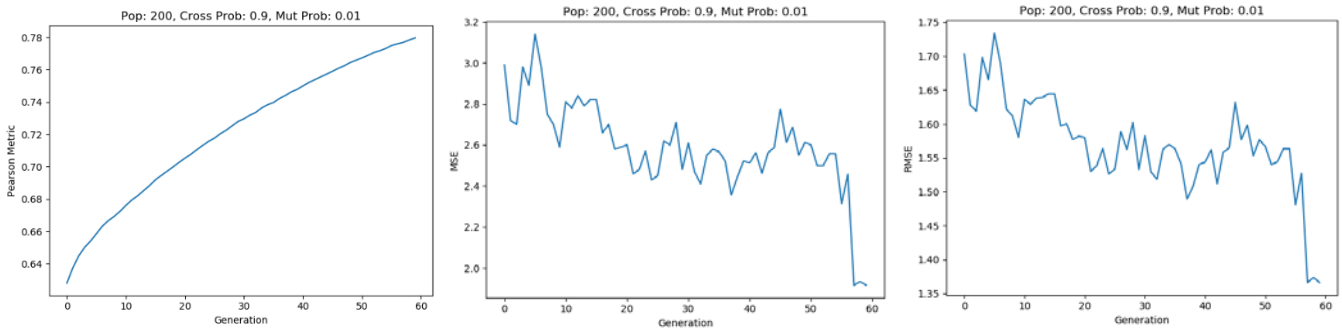
B4. Αξιολόγηση Συστάσεων

α) Το ερώτημα αυτό υλοποιήθηκε στον κώδικα “gen_algo_eval.py”

β) Τα παρακάτω διαγράμματα (Pearson, MSE, RMSE αντίστοιχα) χρησιμοποιούν τις παραμέτρους που αναφέρθηκαν προηγουμένως. Τα διαγράμματα περιλαμβάνουν τιμές μόνο για τον μέσο όρο του πλήθους των γενεών που έτρεξε ο αλγόριθμος σε κάθε επανάληψη (10 επαναλήψεις).



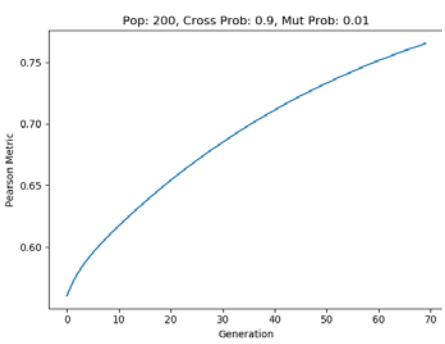
Το μέσο τελικό MSE και RMSE είναι 2.4899 και 1.5598 αντίστοιχα Παρατηρούμε γενικά σημαντική μείωση του σφάλματος και αυτό γιατί, αφού με το πέρασμα των γενεών η απόδοση βελτιώνεται, τότε το διάνυσμα των βαθμολογιών προσομοιάζει όλο και περισσότερο τις πραγματικές βαθμολογίες του χρήστη. Παρόλα αυτά παρατηρούμε στο τέλος μια μικρή αύξηση του σφάλματος. Αυτό σημαίνει ότι ο αλγόριθμός μας έτρεξε περισσότερες γενιές απ’ ότι έπρεπε με αποτέλεσμα να υπερεκπαιδευτεί, δηλαδή να μοιάσει πιο πολύ στους γείτονες του χρήστη παρά στον ίδιο. **Επομένως μια λύση είναι να αυξήσουμε το ελάχιστο δέλτα τερματισμού του αλγορίθμου από 0.0001 σε 0.001.**

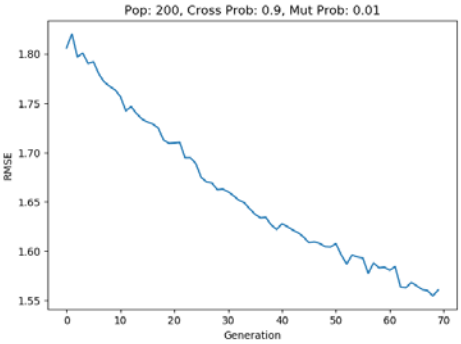


Το μέσο τελικό MSE και RMSE είναι τώρα 2.2499 και 1.4707 αντίστοιχα. Με την νέα μας επιλογή ναι μεν η απόδοση μειώθηκε, δηλαδή η λύση δεν έφτασε όσο το δυνατόν πιο κοντά στους γείτονες του χρήστη, αλλά το σφάλμα μειώθηκε που σημαίνει ότι κατάφερε να φτάσει πιο κοντά στις πραγματικές βαθμολογίες που έδωσε ο ίδιος.

γ) Ο κώδικας για το ερώτημα βρίσκεται στο αρχείο “gen_algo_eval_50.py” για τους 50 πρώτους χρήστες.

Τρέχοντας τον αλγόριθμο για 50 χρήστες με 10 επαναλήψεις ανά χρήστη, υπολογίζουμε στο τέλος τα διαγράμματα μέσης τιμής ανά γενιά για όλους αυτούς και για όλες τις επαναλήψεις. Παρατηρούμε προφανή αύξηση της μετρικής Pearson με συνεχή μείωση των MSE και RMSE μετά από κάθε γενιά.





Το μέσο τελικό MSE και RMSE είναι αντίστοιχα 2.5324 και 1.5513 αντίστοιχα.