ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Επίλυση του προβλήματος του πλανόδιου πωλητή με τη χρήση γενετικών αλγορίθμων** **Solving travelling salesman problem using genetic algorithms**
Ονοματεπώνυμο Φοιτητή	**Πάττας Χρήστος**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΣΠ21044**
Επιβλέπων	**Παναγιωτόπουλος Θεμιστοκλής, Καθηγητής**

Ημερομηνία Παράδοσης **Φεβρουάριος 2023**

Τριμελής Εξεταστική Επιτροπή

Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

Περίληψη

Στην εργασία αυτή παρουσιάζεται το πρόβλημα του πλανόδιου πωλητή και γίνεται επίλυσή του με τη χρήση γενετικών αλγορίθμων. Το πρόβλημα είναι η ελαχιστοποίηση της διαδρομής που πρέπει να ακολουθήσει ένας πλανόδιος πωλητής, ώστε να επισκεφτεί όλες τις πόλεις, από ένα σύνολο πόλεων, ακριβώς μια φορά και να επιστρέψει στην αρχική του θέση. Στην προσπάθειά μας να αναλύσουμε περισσότερο το πρόβλημα, δε μένουμε στο θεωρητικό σκέλος, αλλά δημιουργήσαμε κώδικα για όλους τους αλγορίθμους.

Abstract

In this thesis we're trying to solve the travelling salesman problem or travelling salesperson problem or TSP using genetic algorithms. The travelling salesman problem asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?". In our attempt to do an in-depth analysis of the problem we wrote working code for all the algorithms we used.

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

1 Εισαγωγή

Στην παρούσα εργασία θα αναλύσουμε το πρόβλημα του πλανόδιου πωλητή (ή πρόβλημα του μετακινούμενου πωλητή ή πρόβλημα του περιπλανώμενου πωλητή ή στα αγγλικά travelling salesman problem ή travelling salesperson problem ή για συντομία TSP). Αν και υπάρχουν πολλοί τρόποι επίλυσης του προβλήματος, εμείς θα επικεντρωθούμε στη επίλυσή του με τη χρήση γενετικών αλγορίθμων. Θα δούμε αναλυτικά όλους τους τρόπους, επιλογής, διασταύρωσης και μετάλλαξης και θα αναλύσουμε τα πλεονεκτήματα και τα μειονεκτήματά τους.

Η υλοποίηση όλων των αλγορίθμων έγινε σε PHP 7.4 για τους εξής λόγους:

- Είναι μια απλή γλώσσα με αποτέλεσμα ο χρήστης να μπορεί να κατανοήσει τον αλγόριθμο χωρίς να γνωρίζει php. Αν για παράδειγμα είχε χρησιμοποιηθεί η C++ ο αναγνώστης θα

έπρεπε να γνωρίζει pointers, αν είχε χρησιμοποιηθεί η python, παρόλο που είναι μια εύκολη γλώσσα, το συντακτικό της είναι διαφορετικό από τις άλλες γλώσσες προγραμματισμού.

- Το μόνο που χρειάζεται για να τρέξει ο κώδικας σε php είναι ένας web server και υπάρχουν πολλά site που επιτρέπουν την εκτέλεση php κώδικα.
- Το πρόβλημα του πλανόδιου πωλητή εμφανίζεται σε αρκετές ιστοσελίδες που χρησιμοποιούν php. Από το 2018 και μετά η χρήση των χαρτών της google δεν είναι δωρεάν. Αν μια εταιρεία θέλει να βρει τη βέλτιστη διαδρομή που πρέπει να ακολουθήσει ένα φορτηγό της, θα πρέπει να πληρώσει για τις υπηρεσίες της google. Εναλλακτικά μπορεί να πάρει τα σημεία και τις αποστάσεις με JSON από τον χάρτη και να λύσει το αντίστοιχο TSP πρόβλημα.

Ο κώδικας σε καμία περίπτωση δεν είναι βέλτιστος, καθώς στόχος ήταν η δημιουργία κώδικα ο οποίος θα γίνεται εύκολα κατανοητός από τον αναγνώστη. Επίσης όπου αυτό ήταν δυνατό, αποφεύγουμε τη χρήση συναρτήσεων που υπάρχουν αποκλειστικά στην PHP, ώστε ο αναγνώστης να μπορεί να κατανοήσει των κώδικα χωρίς να γνωρίζει τη γλώσσα. Αν και το πρόβλημα του πλανόδιου πωλητή συνδέεται άμεσα με τη θεωρία γράφων, έγινε προσπάθεια ώστε να ελαχιστοποιήσουμε τις γνώσεις μαθηματικών που χρειάζεται ο αναγνώστης.

Η περιγραφή του προβλήματος του πλανόδιου πωλητή είναι η εξής: Έχουμε Ν πόλεις (σημεία). Η απόσταση μεταξύ των πόλεων είναι γνωστή δηλαδή είναι γνωστό το d_{ij} για κάθε $i, j \in [1, N]$. Έχουμε επίσης έναν πωλητή (σημείο εκκίνησης) και είναι γνωστή η απόστασή του από κάθε πόλη. Ποια είναι η διαδρομή που πρέπει να ακολουθήσει ο πωλητής ώστε να επισκεφτεί όλες τις πόλεις ακριβώς μια φορά και να επιστρέψει στο σημείο που ξεκίνησε, διανύοντας την ελάχιστη απόσταση. Πολλές φορές αντί για την απόσταση χρησιμοποιούνται κάποια βάρη που μπορεί να αντιπροσωπεύουν χρόνο, κόστος κτλ.

Το πρόβλημα μπορεί να φαίνεται απλό και πολλές φορές είναι εύκολο για το ανθρώπινο μάτι να εντοπίσει την ελάχιστη διαδρομή, αν τοποθετήσει τα σημεία πάνω σε ένα χάρτη και ο αριθμός των πόλεων είναι σχετικά μικρός, αλλά όπως θα δούμε στη συνέχεια όσο αυξάνεται ο αριθμός των πόλεων αυξάνεται και η δυσκολία του προβλήματος καθώς το TSP ανήκει στην κατηγορία των NP-hard προβλημάτων.

1.1 Μαθηματική διατύπωση του προβλήματος

Μία από τις κυριότερες προκλήσεις του τομέα της Τεχνητής Νοημοσύνης είναι η δημιουργία ευφυών Το πρόβλημα το πλανόδιου πωλητή μπορεί να διατυπωθεί ως πρόβλημα ακέραιου γραμμικού προγραμματισμού. Υπάρχουν αρκετοί διαφορετικοί τρόποι διατύπωσης με τον πιο γνωστό να προέρχεται από τους Dantzig, Fulkerson και Johnson (George B. Dantzig, 1954). Ζητείται η ελαχιστοποίηση της συνάρτησης:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \text{για } i \neq j$$

με τους περιορισμούς:

$$\sum_{i=1}^{n} x_{ij} \text{ για } j = 1,\dots, n \text{ και } i \neq j \quad (1)$$

$$\sum_{j=1}^{n} x_{ij} \text{ για } i = 1,\dots, n \text{ και } j \neq i \quad (2)$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1 \; \forall Q \subseteq \{1,\dots, n\}, |Q| \geq 2, \; i \neq j \quad (3)$$

Όπου

- n ο αριθμός των πόλεων του προβλήματος
- c_{ij} το κόστος μετάβασης από το σημείο i στο σημείο j
- x_{ij} μια δυαδική μεταβλητή που παίρνει την τιμή 1 αν η ακμή (i, j) υπάρχει στη λύση και την τιμή 0 αν δεν υπάρχει.

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \text{για } i \neq j$$

με τους περιορισμούς:

$$\sum_{i=1}^{n} x_{ij} \text{ για } j = 1,\dots, n \text{ και } i \neq j \quad (1)$$

$$\sum_{j=1}^{n} x_{ij} \text{ για } i = 1,\dots, n \text{ και } j \neq i \quad (2)$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1 \; \forall Q \subseteq \{1,\dots, n\}, |Q| \geq 2, \; i \neq j \quad (3)$$

Εικόνα 1 Μαθηματική διατύπωση του TSM

Οι περιορισμοί (1) και (2) διασφαλίζουν ότι ο πωλητής θα επισκεφθεί κάθε πόλη ακριβώς μια φορά και θα αναχωρήσει από αυτή ακριβώς μια φορά. Ο περιορισμός (3) δεν επιτρέπει τη δημιουργία υποδιαδρομών. Δηλαδή η λύση θα αποτελείται από μια μόνο διαδρομή και όχι από την ένωση υποδιαδρομών. Αν κάποιες πόλεις δεν ενώνονται μεταξύ τους τότε θέτουμε ως κόστος το άπειρο.

2 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι (Genetic algorithms - GA) είναι μεταευρετικοί (metaheuristic) αλγόριθμοι αναζήτησης οι οποίοι βασίζονται στη θεωρία της εξέλιξης και της φυσικής επιλογής όπως αυτές περιγράφονται από το Δαρβίνο και τη γενετική. Ανήκουν σε με ευρύτερη κατηγορία αλγορίθμων οι οποίοι ονομάζονται **Εξελικτικοί Αλγόριθμοι** και χωρίζεται στις εξής υποκατηγορίες.

- Γενετικοί αλγόριθμοι ΓΑ - Genetic algorithms GA
- Εξελικτικές Στρατηγικές (Evolution Strategies - ESs)
- Εξελικτικός Προγραμματισμός (Evolutionary Programming)
- Συστήματα Ταξινόμησης (Classifier Systems)
- Γενετικό Προγραμματισμό (Genetic Programming)

Ο πρώτος γενετικός αλγόριθμος, με τον τρόπο που γνωρίζουμε σήμερα τους γενετικούς αλγορίθμους, αναπτύχθηκε το 1975 από τον Holland (Holland, 1975) και είναι γνωστός με την ονομασία απλός γενετικός αλγόριθμος - simple genetic algorithm (SGA) . Ένα από τα χαρακτηριστικά των γενετικών αλγορίθμων είναι ότι 45 χρόνια μετά τη δημιουργία τους, δεν έχουν αποδειχθεί μαθηματικά. Υπάρχουν πολλές θεωρίες αλλά καμία απόδειξη. Αυτό έχει ως αποτέλεσμα να έχουν προταθεί αναρίθμητες εκδόσεις του απλού γενετικού αλγορίθμου.

Όπως θα δούμε αναλυτικά στα κεφάλαια “Πλεονεκτήματα” και “Μειονεκτήματα”, οι γενετικοί αλγόριθμοι είναι πιο αργοί από τις κλασικές μεθόδους βελτιστοποίησης. Για αυτό το λόγο συνήθως χρησιμοποιούνται σε πολύ δύσκολα προβλήματα όπως τα προβλήματα βελτιστοποίησης πολλαπλών στόχων.

Ένα ακόμα χαρακτηριστικό των γενετικών αλγορίθμων είναι ότι μπορούν να χρησιμοποιηθούν, χωρίς να χρειάζεται να κάνει ο σχεδιαστής μεγάλες αλλαγές στον κώδικα, για την επίλυση σχεδόν οποιουδήποτε προβλήματος. Αντίθετα ένας ευρετικός αλγόριθμος πχ για το πρόβλημα του πλανόδιου πωλητή, δε μπορεί να χρησιμοποιηθεί για την επίλυση άλλου προβλήματος. Με την υπολογιστική ισχύ που διαθέτουμε στις μέρες μας, ίσως είναι προτιμότερο να αναπτυχθεί ένας γενετικός αλγόριθμος, καθώς αλλάζοντας μόνο τη συνάρτηση καταλληλόλητας, μπορεί να χρησιμοποιηθεί για οποιοδήποτε πρόβλημα. Αν η απόδοση δεν είναι ο πρωταρχικός μας στόχος, μπορούμε να γλυτώσουμε πολλές ώρες που απαιτούνται για το σχεδιασμό νέων αλγορίθμων για κάθε πρόβλημα.

Οι γενετικοί αλγόριθμοι χρησιμοποιούν ορολογία δανεισμένη από το χώρο της βιολογίας και πιο συγκεκριμένα από τη γενετική. Κάθε μεταβλητή του προς επίλυση προβλήματος κωδικοποιείται σε μια συμβολοσειρά η οποία ονομάζεται χρωμόσωμα΄(chromosome). Τα χρωμοσώματα αποτελούνται από γονίδια (genes), δηλαδή από τα σύμβολα που χρησιμοποιούνται για την αναπαράσταση των υποψήφιων λύσεων. Συνήθως όλες οι μεταβλητές κωδικοποιούνται σε μια συμβολοσειρά αλλά υπάρχουν και γενετικοί αλγόριθμοι με περισσότερες. Το σύνολο των χρωμοσωμάτων που χρησιμοποιούνται για την αναπαράσταση μιας υποψήφιας λύσης ονομάζεται άτομο (individual).

Οι γενετικοί αλγόριθμοι στην προσπάθειά τους να αντιγράψουν τη διαδικασία της φυσικής επιλογής, κάνουν αναζήτηση μέσα από ένα σύνολο πιθανών λύσεων (ατόμων) το οποίο ονομάζεται πληθυσμός (population). Αυτό έχει ως αποτέλεσμα σε κάθε επανάληψη να παράγονται πολλές πιθανές λύσεις σε αντίθεση με τους άλλους μεταυρετικούς αλγορίθμους οι οποίοι παράγουν μόνο μία.

Το πρώτο στάδιο στην ανάπτυξη ενός γενετικού αλγορίθμου, αφού έχουμε αποφασίσει με πιο τρόπο θα γίνει η αναπαράσταση των υποψηφίων λύσεων, είναι η δημιουργία ενός αρχικού πληθυσμού ο οποίος ονομάζεται αρχικός πληθυσμός (initial population) ή πρώτη γενιά (first generation). Συνήθως η επιλογή του αρχικού πληθυσμού γίνεται τυχαία. Αυτή η διαδικασία ονομάζεται αρχικοποίηση (Initialization). Στόχος του γενετικού αλγορίθμου είναι με τη χρήση των τελεστών της επιλογής, της διασταύρωσης και της μετάλλαξης, να εξελίξει τον αρχικό πληθυσμό δηλαδή να δημιουργήσει καλύτερες λύσεις.

Στον απλό γενετικό αλγόριθμο, μετά τη δημιουργία του αρχικού πληθυσμού γίνεται η επιλογή (selection). Η βασική ιδέα της επιλογής είναι βασισμένη στην αρχή της επιβίωσης του καταλληλότερου (survival of the fitness). Οι πιθανές λύσεις (άτομα) αξιολογούνται (evaluation) και αυτές με καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιβιώσουν και να παράγουν απογόνους. Ο υπολογισμός της

απόδοσης των ατόμων γίνεται με τη συνάρτηση καταλληλότητας (fitness function). Για απλά προβλήματα βελτιστοποίησης η συνάρτηση καταλληλότητας είναι η αντικειμενική συνάρτηση.

Ο τελεστής της επιλογής δεν εισάγει νέες λύσεις στον πληθυσμό. Αυτό που κάνει είναι να μειώνει την ποικιλομορφία του πληθυσμού, με το να αφαιρεί άτομα με χαμηλή απόδοση και να τοποθετεί στη θέση τους αντίγραφα των ατόμων με υψηλή απόδοση. Μετά τη εφαρμογή του τελεστή της επιλογής δημιουργείται ένας νέος προσωρινός πληθυσμός στον οποίο εφαρμόζονται οι τελεστές της διασταύρωσης και της μετάλλαξης.

Με τη διασταύρωση επιλέγονται τυχαία δύο (ή και περισσότερα) άτομα του προσωρινού πληθυσμού και ανταλλάσσουν γενετικό υλικό (γονίδια) δημιουργώντας νέα άτομα. Η λογική βασίζεται στην ιδέα ότι τα καλύτερα άτομα που προέκυψαν από τη διαδικασία της επιλογής, έχουν καλύτερο γενετικό υλικό το οποίο αν συνδυαστεί θα δημιουργήσει απογόνους με ακόμα καλύτερη απόδοση. Στην πράξη πολύ συχνά προκύπτουν απόγονοι με χειρότερη απόδοση από τους γονείς. Σε αντιστοιχία με τη βιολογία μπορούμε να πούμε ότι η διασταύρωση είναι η σεξουαλική αναπαραγωγή.

Μετά την διασταύρωση έχουμε την μετάλλαξη. Στη Βιολογία με τον όρο μετάλλαξη (mutation), χαρακτηρίζεται οποιαδήποτε μεταβολή που μπορεί να συμβεί στο γενετικό υλικό ενός οργανισμού. Οι μεταλλάξεις συμβαίνουν με τυχαίο τρόπο, χωρίς αυτό να σημαίνει ότι δεν υπόκεινται και στην επίδραση του περιβάλλοντος. Ειδικότερα, είναι τυχαίες με την έννοια ότι η πιθανότητα να εμφανιστεί μια μετάλλαξη δεν σχετίζεται με το βαθμό χρησιμότητάς της. Ανάλογα με τη σημασία τους στην εξέλιξη, διακρίνονται σε ευνοϊκές, επιβλαβείς ή ουδέτερες. Στους γενετικούς αλγορίθμους η μετάλλαξη αλλάζει τυχαία κάποιο/α γονίδιο/α κάποιων χρωμοσωμάτων, τροφοδοτώντας τον πληθυσμό των πιθανών λύσεων με νέες λύσεις. Η μετάλλαξη μπορεί να οδηγήσει το γενετικό αλγόριθμο σε νέες λύσεις, στις οποίες είναι αδύνατο να φτάσει μόνο με τη διασταύρωση, με αποτέλεσμα να αποτρέπεται η πρόωρη σύγκλιση.

Μετά την εφαρμογή των τελεστών της επιλογής, της διασταύρωσης και της μετάλλαξης, δημιουργείται ένας νέος πληθυσμός, ένα νέο σύνολο από χρωμοσώματα. Για ευκολία στους περισσότερους γενετικούς αλγορίθμους ο αριθμός των ατόμων του πληθυσμού παραμένει σταθερός από γενιά σε γενιά, αλλά αυτό δεν αποτελεί κανόνα. Ο νέος πληθυσμός χρησιμοποιείται στην επόμενη επανάληψη του γενετικού αλγορίθμου ως αρχικός πληθυσμός. Κάθε επανάληψη του αλγορίθμου περιλαμβάνει τα εξής βήματα: Αξιολόγηση του πληθυσμού με τη χρήση της συνάρτησης καταλληλότητας, επιλογή των ατόμων με βάση την απόδοσή τους, διασταύρωση κάποιων χρωμοσωμάτων, μετάλλαξη κάποιων γονιδίων. Ο γενετικός αλγόριθμος επαναλαμβάνει τα βήματα αξιολόγηση, επιλογή, διασταύρωση, μετάλλαξη, μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού. Στα σχήματα που ακολουθούν βλέπουμε την εξέλιξη του πληθυσμού κατά τη διάρκεια της εκτέλεσης ενός γενετικού αλγορίθμου.

Εικόνα 2 Αρχή γενετικού αλγορίθμου. Ο πληθυσμός βρίσκεται τυχαία σε όλο το χώρο

Εικόνα 3 Μετά από κάποιες εκτελέσεις, ο πληθυσμός πηγαίνει προς τα ακρότατα

Εικόνα 4 Τέλος αλγορίθμου. Ο Πληθυσμός έχει συγκεντρωθεί στα ακρότατα

2.1 Ιστορική αναδρομή γενετικών αλγορίθμων

Στις αρχές του 1950 οι επιστήμονες ήθελαν να χρησιμοποιήσουν τους υπολογιστές για την προσομοίωση βιολογικών συστημάτων. Η πρώτη ολοκληρωμένη έρευνα έγινε από τον Nils Barricelli το 1954 (Barricelli, Esempi numerici di processi di evoluzione, 1954) (Barricelli, Symbiogenetic evolution processes realized by artificial methods, 1957) ο οποίος ως στόχο είχε τη δημιουργία τεχνητής ζωής και όχι την προσομοίωση της βιολογικής εξέλιξης ή τη βελτιστοποίηση. Επειδή όμως η έρευνά του ήταν στα Ιταλικά, δεν έγινε γνωστή στην επιστημονική κοινότητα.

Το 1957 ο Alex Fraser (Frasher, 1957) δημιούργησε έναν αλγόριθμο με τον οποίο προσπάθησε να προσομοιώσει την τεχνητή επιλογή και δημοσίευσε μια σειρά από μελέτες που έκανε (Fogel D. , 2002) .Ήταν ο πρώτος που χρησιμοποίησε έναν αλγόριθμο, αποκλειστικά για τη μελέτη της εξέλιξης. Την επόμενη δεκαετία η χρήση των υπολογιστών για την προσομοίωση βιολογικών συστημάτων έγινε αρκετά διαδεδομένη και οι μέθοδοι που χρησιμοποιήθηκαν περιγράφονται αναλυτικά από τους Fraser και Burnell 1970 (Fraser & Burnell, 1970) και Crosby (Crosby, 1973). Σημαντική συνεισφορά στην εξέλιξη των γενετικών αλγορίθμων είχαν οι Hans-Joachim Bremermann, Richard Friedberg, George Friedman και Michael Conrad, με τον David B. Fogel να αναδημοσιεύει το 1998 (Fogel D. B., 1998) πολλές από τις αρχικές έρευνες.

Την ίδια δεκαετία (1960-1970) οι Fogel L. J., Owens A. J., Walsh M. J. έθεσαν τις βάσεις για τον εξελικτικό προγραμματισμό (evolutionary programming) (Fogel L. J., 1966) και οι Ingo Rechenberg και Hans-Paul Schwefel τις βάσεις για τις εξελικτικές στρατηγικές (evolution strategies) (Schwefel, 1965) (Rechenberg, 1973).

Αν και αναφορές στον όρο γενετικός αλγόριθμος υπάρχουν από το 1967 (Bagley, 1967), οι γενετικοί αλγόριθμοι με τη μορφή που τους γνωρίζουμε σήμερα δημιουργήθηκαν από τον John Henry Holland και την ομάδα του στο πανεπιστήμιο του Μίσιγκαν. Ο Holland ξεκίνησε την έρευνά του στις αρχές της δεκαετίας του 60 και την ολοκλήρωσε το 1975 με τη δημοσίευση του βιβλίου του “Adaptation in Natural and Artificial Systems” (Holland, 1975). Όπως λέει ο ίδιος ο Holland, ήθελε να δημιουργήσει προγράμματα υπολογιστών που "εξελίσσονται" με τρόπους που προσομοιώνουν τη φυσική επιλογή και μπορούν να λύσουν περίπλοκα προβλήματα τα οποία ούτε οι δημιουργοί τους δεν είναι σε θέση να κατανοήσουν πλήρως. “Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand” - John H. Holland

Το 1975 ο K.A. De Jong (Jong, 1975) επίσης από το πανεπιστήμιο του Μίσιγκαν χρησιμοποίησε τη θεωρία του Holland για να κάνει διάφορα υπολογιστικά πειράματα, προτείνοντας τρόπους για να αντιμετωπίσουν οι γενετικοί αλγόριθμοι δύσκολα προβλήματα.

Μέχρι τις αρχές του 1980 η έρευνα στους γενετικούς αλγορίθμους ήταν κυρίως θεωρητική. Το 1980 ο J.J. Grefenstette δημιουργεί το GENESIS, ένα πρόγραμμα ανάπτυξης γενετικών αλγορίθμων.

Από τις αρχές της δεκαετίας του 80, το ενδιαφέρον για τους γενετικούς αλγορίθμους αρχίζει να μεγαλώνει με πολλές έρευνες και πληθώρα εφαρμογών για την επίλυση διαφόρων προβλημάτων. Αυτό είχε ως αποτέλεσμα να γίνει το 1985 το πρώτο διεθνές συνέδριο για τους γενετικούς αλγορίθμους (Pittsburgh, Pennsylvania). Στην αρχή το συνέδριο λάμβανε χώρα κάθε δύο χρόνια. Το 1999 έγινε ένωση του συνεδρίου των γενετικών αλγορίθμων με αυτό του γενετικού προγραμματισμού. Το συνέδριο διεξάγεται κάθε χρόνο και ονομάζεται συνέδριο γενετικών και εξελικτικών υπολογισμών - Genetic and Evolutionary Computation Conference (GECCO).

Το 1989 η Axcelis, Inc. δημιούργησε το Evolver, το πρώτο εμπορικό πρόγραμμα βελτιστοποίησης για προσωπικούς υπολογιστές βασισμένο στους γενετικούς αλγορίθμους.

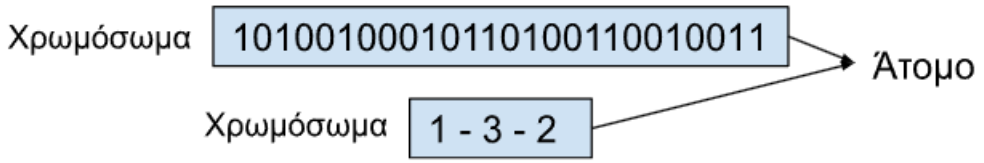
Το 1992 ένας από τους μαθητές του John Henry Holland, ο John Koza βασιζόμενος στους γενετικούς αλγορίθμους, δημιούργησε μια νέα κατηγορία εξελικτικών αλγορίθμων την οποία ονόμασε γενετικό προγραμματισμό (genetic programming).

Τα τελευταία τριάντα χρόνια, οι έρευνες έχουν απομακρύνει τους γενετικούς αλγορίθμους από τον απλό γενετικό αλγόριθμο. Οι γενετικοί αλγόριθμοι δε χρησιμοποιούν πλέον μόνο τη δυαδική κωδικοποίηση, μπορεί να έχουν δυναμική πιθανότητα διασταύρωσης ή και μετάλλαξης, χρησιμοποιούν ντετερμινιστικούς αλγορίθμους διασταύρωσης ή και μετάθεσης ανάλογα με την κωδικοποίηση ή το πρόβλημα, προκειμένου να εκμεταλλευτούν τη γνώση και να συγκλίνουν γρηγορότερα, η επιλογή μπορεί να συμβαίνει σε δύο στάδια (επιλογή επιζώντα) κτλ. Επειδή ακόμα και σήμερα ο τρόπος λειτουργίας των γενετικών αλγορίθμων δεν έχει αποδειχθεί μαθηματικά, υπάρχουν αναρίθμητες υλοποιήσεις και συνέχεια δημιουργούνται καινούργιες.

2.2 Ορολογία

Οι Γ.Α. χρησιμοποιούν ορολογία δανεισμένη από το χώρο της γενετικής. Στη συνέχεια θα δούμε τους κυριότερους ορισμούς που χρησιμοποιούμε στους Γ.Α. στα ελληνικά και στα αγγλικά.

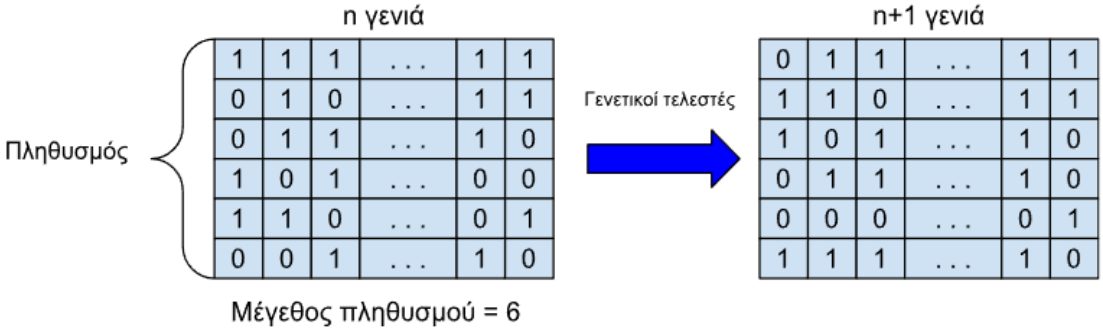
- **Αρχικοποίηση (Initialization).** Ονομάζεται η διαδικασία δημιουργίας του αρχικού πληθυσμού. Τις περισσότερες φορές ο αρχικός πληθυσμός δημιουργείται τυχαία.
- **Άτομο (individuals - creatures).** Είναι μια πιθανή λύση του προβλήματος, δηλαδή ένα σύνολο χρωμοσωμάτων. Τις περισσότερες φορές όλες οι μεταβλητές του προβλήματος κωδικοποιούνται σε ένα μόνο χρωμόσωμα.
- **Χρωμόσωμα (chromosome).** Κάθε άτομο αποτελείται από χρωμοσώματα δηλαδή μια συμβολοσειρά που συνήθως είναι δυαδική, η οποία αναπαριστά την κωδικοποίηση μιας ή και περισσοτέρων μεταβλητών. Τις περισσότερες φορές στους γενετικούς αλγορίθμους τα άτομα έχουν ένα μόνο χρωμόσωμα για αυτό το λόγο στη συνέχεια της εργασίας με τους όρους άτομο και χρωμόσωμα θα εννοούμε το ίδιο.



Εικόνα 5 Ένα άτομο μπορεί να αποτελείται από χρωμοσώματα με διαφορετική κωδικοποίηση

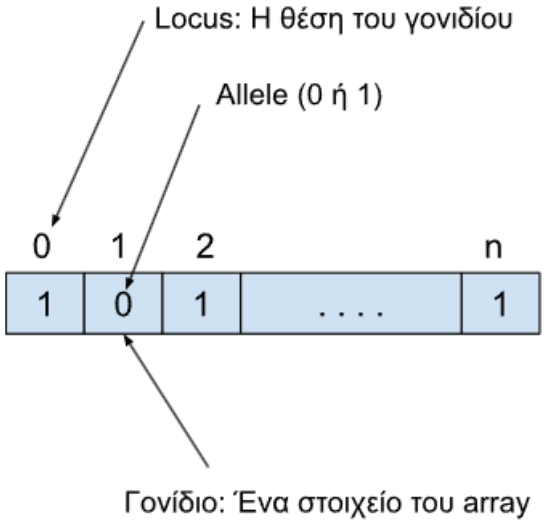
- **Πληθυσμός (population).** Ένα σύνολο από άτομα, δηλαδή ένα σύνολο από υποψήφιες λύσεις.
- **Αρχικός πληθυσμός (initial population).** Το αρχικό σύνολο από άτομα. Συνήθως δημιουργείται τυχαία.
- **Μέγεθος πληθυσμού (population size).** Αριθμός των ατόμων υπάρχουν σε έναν πληθυσμό. Στους περισσότερους Γ.Α. ο αριθμός αυτός παραμένει σταθερός, αλλά υπάρχουν και κάποιες υλοποιήσεις με δυναμικό μέγεθος πληθυσμού (B.R.Rajakumara, 2013) (Erna Budhiarti Nab, Genetic Algorithms Dynamic Population Size with Cloning in Solving Traveling Salesman Problem, 2018).

- **Γενιά (generation).** Ο αρχικός πληθυσμός ενός Γ.Α. αποτελεί την πρώτη γενιά. Σε κάθε κύκλο του αλγορίθμου, ο τρέχων πληθυσμός μέσα από μια σειρά πράξεων μας δίνει νέα γενιά.



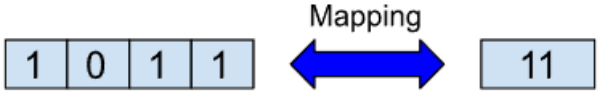
Εικόνα 6 Δημιουργία νέου πληθυσμού

- **Γονίδιο (gene).** Ένα σύμβολο της συμβολοσειράς του χρωμοσώματος ονομάζεται γονίδιο.
- **Αλληλόμορφο (allele).** Οι τιμές που μπορεί να πάρει ένα γονίδιο.
- **Θέση (locus).** Η θέση που βρίσκεται το γονίδιο στο χρωμόσωμα.



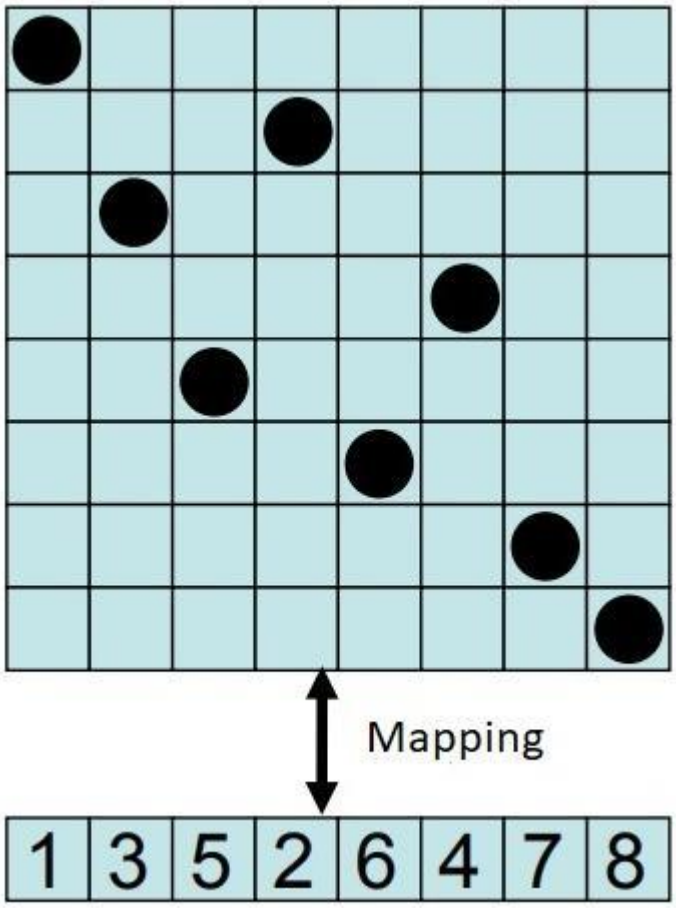
Εικόνα 7 Γονίδιο - Αλληλόμορφο - Θέση

- **Γονότυπος - γενότυπος (genotype).** Στα περισσότερα βιβλία είναι το ίδιο με το χρωμόσωμα, αλλά σε μερικά το ίδιο με το άτομο.
- **Φαινότυπος (phenotype).** Φαινότυπος είναι όλα τα μορφολογικά, παραγωγικά, ηθολογικά κ.λπ.. χαρακτηριστικά που εκδηλώνει ένας οργανισμός σε μία δεδομένη στιγμή, δηλαδή το μέρος του γονοτύπου του οργανισμού το οποίο μπορούμε (άμεσα ή έμμεσα) να παρατηρήσουμε. Στους Γ.Α. με τον όρο φαινότυπος εννοούμε την αποκωδικοποιημένη λύση.
- **Χαρτογράφηση (mapping).** Η αντιστοίχιση του γονότυπου στον φαινότυπο.



Εικόνα 8 Mapping δυαδικής κωδικοποίησης ακεραίου

Για το πρόβλημα των 8 βασιλισσών όπου κάθε χρωμόσωμα είναι μια μετάθεση των αριθμών 1 έως 8, η μετατροπή από γονότυπο σε φαινότυπο και αντίστροφα, φαίνεται στο σχήμα που ακολουθεί.



Εικόνα 9 Mapping για το πρόβλημα των 8 βασιλισσών

- **Αναπαράσταση - Κωδικοποίηση (representation - encoding).** Η διαδικασία μετατροπής της πραγματικής λύσης σε μια συμβολοσειρά (συνήθως δυαδική). Οι όροι αναπαράσταση και κωδικοποίηση δεν είναι το ίδιο. Για παράδειγμα λέμε ότι χρησιμοποιήθηκε η δυαδική κωδικοποίηση, ενώ με τον όρο αναπαράσταση της λύσης εννοούμε, τον αριθμό των δυαδικών ψηφίων που χρησιμοποιήθηκαν και την θέση του κάθε γονιδίου στο χρωμόσωμα. Στη συγκεκριμένη εργασία οι όροι αναπαράσταση και κωδικοποίηση θα σημαίνουν ακριβώς το ίδιο, καθώς αυτό συμβαίνει σε πολλές έρευνες.
- **Αποκωδικοποίηση (decoding).** Η μετατροπή του γονότυπου της λύσης στον αντίστοιχο φαινότυπο, δηλαδή την ίδια τη λύση. Για παράδειγμα αν έχουμε δυαδική κωδικοποίηση ενός ακέραιου αριθμού, ο γενότυπος (0 1 1), αντιστοιχεί στη λύση 3.
- **Απόδοση - Καταλληλότητα (fitness).** Κάθε άτομο (υποψήφια λύση) αξιολογείται από μια συνάρτηση καταλληλότητας (fitness function). Οι καλύτερες λύσεις έχουν μεγαλύτερη τιμή καταλληλότητας.
- **Αντικειμενική συνάρτηση - Συνάρτηση ικανότητας - Συνάρτηση καταλληλότητας - Συνάρτηση αξιολόγησης (objective function - fitness function).** Η συνάρτηση με την οποία υπολογίζουμε πόσο καλές είναι οι λύσεις. Ανάλογα με το πρόβλημα ο προσδιορισμός της συνάρτησης καταλληλότητας μπορεί να είναι κάτι πολύ απλό, ή να αποτελεί ένα ξεχωριστό πρόβλημα. Για το πρόβλημα του πλανόδιου πωλητή η συνάρτηση καταλληλότητας είναι η

f(x)=1/Cx, όπου Cx το κόστος της διαδρομής του ατόμου x. Σημείωση: Σε περίπλοκα προβλήματα η αντικειμενική συνάρτηση και η συνάρτηση καταλληλότητας δεν είναι το ίδιο.

- **Αξιολόγηση (evaluation).** Η διαδικασία ταξινόμησης των γονοτύπων με βάση την καταλληλότητά τους.
- **Ελιτισμός (elitism).** Η εξασφάλιση της αντιγραφής του ατόμου με την καλύτερη απόδοση στην επόμενη γενιά, πριν τη διαδικασία επιλογής.
- **Επιλογή (selection).** Σε κάθε κύκλο του αλγορίθμου επιλέγονται με βάση την καταλληλότητα (fitness) τα άτομα που θα παράγουν το νέο πληθυσμό. Τα άτομα με μεγαλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιλεγούν. Χωρίς την επιλογή ο γενετικός αλγόριθμος μετατρέπεται σε έναν αλγόριθμο τυχαίας αναζήτησης. Σημείωση: Στον απλό Γ.Α. αλγόριθμο η επιλογή γίνεται πριν τη διασταύρωση και τη μετάλλαξη. Σε κάποιες νεότερες υλοποιήσεις Γ.Α. έχουμε δύο στάδια επιλογής. Η επιλογή που γίνεται πριν τους τελεστές διασταύρωσης και μετάλλαξης ονομάζεται επιλογή γονέα (parent selection). Επιλογή όμως μπορεί να γίνει και μετά τους γενετικούς τελεστές. Τα L άτομα που προέκυψαν από τους γενετικούς τελεστές (απόγονοι) ενώνονται με τα N άτομα του πληθυσμού. Από τα N+L άτομα επιλέγονται τα N που θα αποτελέσουν το νέο πληθυσμό. Η διαδικασία αυτή ονομάζεται επιλογή επιζώντων (survivor selection) και μπορεί να γίνει με δύο τρόπους.
    - Βάσει ηλικίας (age based). Δεν λαμβάνεται υπόψη η καταλληλότητα των ατόμων αλλά μόνο η ηλικία τους, δηλαδή το πόσο παλιά είναι. Τα L παλαιότερα άτομα αντικαθίστανται από τους L απογόνους για να έχει ο νέος πληθυσμός N άτομα.

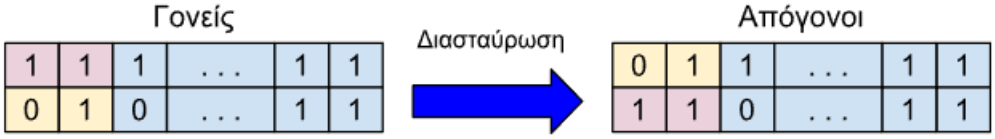
Εικόνα 10 Age based

- Βάσει καταλληλότητας (fitness based). Από τα N+L άτομα επιλέγουμε N με τα άτομα που έχουν καλύτερη καταλληλότητα να έχουν μεγαλύτερη πιθανότητα να επιλεγούν.

Εικόνα 11 Fitness based

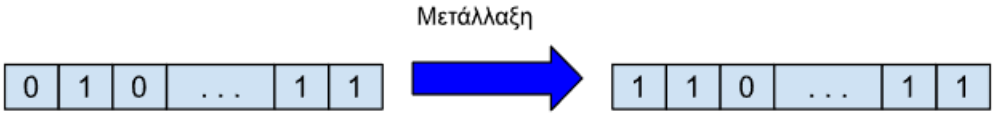
- **Γονείς (parents).** Το σύνολο των ατόμων που δημιουργήθηκε από τη διαδικασία της επιλογής.
- **Απόγονοι (offsprings).** Το σύνολο των ατόμων που προέκυψε μετά την εφαρμογή των γενετικών τελεστών και θα αποτελέσει το νέο πληθυσμό στην επόμενη επανάληψη του γενετικού αλγορίθμου.

- **Διασταύρωση (crossover - recombination).** Η διαδικασία κατά την οποία δύο ή και περισσότερα άτομα (γονείς) ανταλλάσσουν γονίδια για να δημιουργήσουν ένα ή και περισσότερα άτομα (απογόνους). Στους περισσότερους Γ.Α. από δύο γονείς προκύπτουν δύο απόγονοι. Στη εικόνα που ακολουθεί βλέπουμε έναν απλό τρόπο διασταύρωσης κατά τον οποίο οι δύο γονείς ανταλλάσσουν τα πρώτα 2 γονίδιά τους για να παράγουν δύο απογόνους. Προφανώς αυτός ο τρόπος διασταύρωσης δεν μπορεί να εφαρμοστεί όταν τα γονίδια ενός χρωμοσώματος πρέπει να είναι μοναδικά.
- **Ποσοστό διασταύρωσης (crossover rate).** Είναι η πιθανότητα με την οποία γίνεται η διασταύρωση. Η τιμή της καθορίζεται από το σχεδιαστή του Γ.Α.



Εικόνα 12 Διασταύρωση ενός σημείου

- **Μετάλλαξη (mutation).** Κάποια από τα γονίδια, ενός ποσοστού των ατόμων αλλάζουν τιμή για να βοηθήσουν τον Γ.Α. να εξερευνήσει το χώρο των λύσεων. Στην περίπτωση της δυαδικής κωδικοποίησης ένας απλός τρόπος μετάλλαξης είναι κάποια από τα γονίδια ενός ατόμου να γίνουν από μηδέν ένα ή από ένα μηδέν. Στο σχήμα που ακολουθεί μεταλλάχθηκε το πρώτο γονίδιο του ατόμου.



Εικόνα 13 Παράδειγμα μετάλλαξης

- **Ποσοστό μετάλλαξης (mutation rate).** Η πιθανότητα με την οποία γίνεται η μμετάλλαξη. Η τιμή της καθορίζεται από το σχεδιαστή του ΓΑ.
- **Κριτήριο τερματισμού (termination condition**). Η συνθήκη που πρέπει να ικανοποιηθεί, προκειμένου να σταματήσει η εκτέλεση του αλγορίθμου. Ένας γενετικός αλγόριθμος μπορεί να έχει πολλά κριτήρια τερματισμού.

2.3 Πλεονεκτήματα Γ.Α.

Σε αυτό το κεφάλαιο θα δούμε τους λόγους για τους οποίους κάποιος θα επιλέξει τους Γ.Α. αντί για κάποια άλλη μετα-ευρετική μέθοδο αναζήτησης.

- Οι γενετικοί αλγόριθμοι κάνουν αναζήτηση σε πολλές κατευθύνσεις ταυτόχρονα, αφού σε κάθε κύκλο παράγουν αριθμό λύσεων ίσο με το μέγεθος του πληθυσμού. Αυτή η ιδιότητα των γενετικών αλγορίθμων τους δίνει δύο πλεονεκτήματα.
  - Αν το πρόβλημα έχει μεγάλο αριθμό τοπικών ακροτάτων ενώ οι άλλες μέθοδοι αναζήτησης εγκλωβίζονται, ένας γενετικός αλγόριθμος είναι σε θέση να τα ξεπεράσει.
  - Οι γενετικοί αλγόριθμοι στο τέλος δεν μας δίνουν μία μόνο λύση, αλλά ένα πλήθος λύσεων ίσο με το μέγεθος του πληθυσμού. Έτσι μας δίνεται η δυνατότητα να διαλέξουμε από ένα πλήθος σχεδόν ισάξιων λύσεων, ανάλογα με τις ανάγκες μας.

Για παράδειγμα μπορούμε χρησιμοποιήσουμε έναν Γ.Α. για να δημιουργήσουμε το πρόγραμμα του εξαμήνου σε ένα πανεπιστήμιο. Οι λύσεις που θα δώσει ο Γ.Α. πρέπει να ακολουθούν κάποιους κανόνες π χ ένας καθηγητής να μην έχει δύο διαφορετικά μαθήματα την ίδια ώρα, να μην υπάρχουν δύο μαθήματα στην ίδια αίθουσα την ίδια ώρα, οι φοιτητές κάποιου έτους να μην έχουν δύο διαφορετικά μαθήματα την ίδια ώρα κτλ., αλλά και κάποιοι πιο

ελαστικοί περιορισμοί που επηρεάζουν τη συνάρτηση καταλληλότητας όπως να μην υπάρχει μεγάλο χρονικό κενό ανάμεσα στις ώρες που διδάσκει ένας καθηγητής, να μην υπάρχει μεγάλο χρονικό κενό στις ώρες διδασκαλίας των φοιτητών κάποιου έτους κτλ. Όταν ο γενετικός αλγόριθμος ολοκληρωθεί, οι καθηγητές μπορούν να δουν όλες τις πιθανές λύσεις και να συμφωνήσουν σε αυτή που βολεύει τους περισσότερους.

- Δεν υπάρχουν περιορισμοί στη συνάρτηση του προβλήματος που επεξεργάζονται. Οι περισσότερες άλλες μέθοδοι έχουν τέτοιους περιορισμούς όπως:
    - Η συνάρτηση να είναι συνεχής.
    - Η συνάρτηση να είναι διαφορίσιμη.
    - Να μην υπάρχει "θόρυβος" στη συνάρτηση.
- Οι Γ.Α. μπορούν να εφαρμοστούν σχεδόν σε οποιοδήποτε πρόβλημα. Μερικές από τις σχεδόν **αναρίθμητες εφαρμογές των Γ.Α.** είναι οι εξής:
    - Εκπαίδευση νευρωνικών δικτύων (Risi & Stanley, 2012). Η διαδικασία αυτή είναι γνωστή ως Neuroevolution.
    - Χρονοπρογραμματισμός (scheduling) (Wall, 1996).
    - Μηχανική μάθηση (genetics based machine learning GBML) (Adarsh Sehgal, 2019).
    - Σχεδιασμός ρομποτικών τροχιών (robot trajectory generation) (Sandi Baressi Šegota, 2020).
    - Οικονομική επιστήμη (Shu-heng Chen, 1996) (Sefiane Slimane, 2012).
    - Πρόβλεψη κλιματολογικών αλλαγών (Stanislawska, Krawiec, & Kundzewicz, 2012).
    - Βελτιστοποίηση χαρτοφυλακίου (Sefiane, 2012).
    - Βελτιστοποίηση "προβληματικών" (μη συνεχών, μη παραγωγίσιμων, με θόρυβο") συναρτήσεων, ή στη βελτιστοποίηση πολλαπλών στόχων (Abdullah Konak, 2006).
    - Επεξεργασία εικόνας (A. dos Santos-Paulino, 2014).
    - Σχεδιασμός αντιτρομοκρατικών συστημάτων (Buurman, Zhang, & Babovic, 2009).
    - Έλεγχος ποιότητας.
    - Μελέτη υπόγειων νερών (Fisher, 2013).
    - Επίλυση του προβλήματος του πλανόδιου πωλητή και των παραλλαγών του.
- Μπορούν εύκολα να συνεργαστούν με άλλες μεθόδους αναζήτησης. Για παράδειγμα για το πρόβλημα του πλανόδιου πωλητή έχουν αναπτυχθεί πολλοί ευρετικοί αλγόριθμοι, οι οποίοι κάνοντας χρήση της γνώσης που έχουν για το πρόβλημα μας δίνουν πολύ καλές λύσεις αρκετά γρήγορα. Μπορούμε πολύ εύκολα να συνδυάσουμε κάποια από αυτές τις μεθόδους με τον γενετικό αλγόριθμο και με τη χρήση αυτού του υβριδικού αλγορίθμου να πάρουμε μία ακόμα καλύτερη λύση.
- Οι Γ.Α. μπορούν πολύ εύκολα να συνεργαστούν με τα υπάρχοντα μοντέλα και συστήματα χωρίς να χρειάζεται η επανασχεδίασή τους. Αυτό συμβαίνει γιατί οι Γ.Α. χρησιμοποιούν μόνο πληροφορίες της συνάρτησης που πρόκειται να βελτιστοποιήσουν χωρίς να τους ενδιαφέρει άμεσα ο ρόλος της συνάρτησης στο σύστημα, ή η δομή του συστήματος.
- Πολλά στάδια του Γ.Α. μπορούν να υλοποιούν παράλληλα, κάνοντας τη διαδικασία πιο γρήγορη. Για παράδειγμα κατά την εφαρμογή του τελεστή διασταύρωσης μπορούμε να δημιουργήσουμε όλους τους απογόνους σε ένα βήμα αν υποθέσουμε ότι έχουμε N/2 επεξεργαστές, όπου N ο αριθμός των απογόνων (υποθέτουμε ότι δύο γονείς παράγουν δύο απογόνους).
- Οι γενετικοί αλγόριθμοι είναι από τη φύση τους παράλληλοι. Όπως θα δούμε στο κεφάλαιο "θεωρία σχημάτων" αν το μέγεθος του πληθυσμού είναι n ο αλγόριθμος επεξεργάζεται σε κάθε κύκλο n^3 σχήματα. Η επεξεργασία των σχημάτων γίνεται μέσω των ατόμων που αντιπροσωπεύουν χωρίς τα σχήματα να εμφανίζονται στον κώδικα του Γ.Α. και να τον

επιβαρύνουν. Το φαινόμενο αυτό ονομάστηκε από τον Holland έμμεσος παραλληλισμός (implicit parallelism).

- Μπορούν να λύσουν γρήγορα και με μεγάλη ακρίβεια πολύ δύσκολα προβλήματα. Τόσο στη θεωρία όσο και στην πράξη οι γενετικοί αλγόριθμοι είναι πολύ αποδοτικοί στην αντιμετώπιση προβλημάτων με πολλές και δύσκολα προσδιορισμένες λύσεις ή προβλήματα στα οποία οι συναρτήσεις παρουσιάζουν μεγάλες διακυμάνσεις.
- Είναι εύκολοι στην κατανόηση. Η σωστή υλοποίηση και παραμετροποίηση ενός γενετικού αλγορίθμου είναι μια δύσκολη διαδικασία, όμως η κατανόηση της βασικής ιδέας είναι αρκετά εύκολη γιατί οι γενετικοί αλγόριθμοι έχουν τη βάση τους στη θεωρία της εξέλιξης. Έννοιες όπως η επιλογή, η διασταύρωση και η μετάλλαξη μπορούν εύκολα να γίνουν κατανοητές.
- Κάνουν αναζήτηση στο χώρο εκμεταλλευόμενοι τις πληροφορίες από τα προηγούμενα βήματα. Ένας αλγόριθμος τυχαίας αναζήτησης κάνει καλή αναζήτηση στο χώρο, αλλά δεν εκμεταλλεύεται τις πληροφορίες από τα προηγούμενα βήματα, με αποτέλεσμα να είναι πολύ αργός. Ένας αλγόριθμος τοπικής αναζήτησης όπως ο Hill climbing εκμεταλλεύεται τις πληροφορίες από τα προηγούμενα βήματα, αλλά ο χώρος αναζήτησης που εξετάζει είναι πολύ μικρός, με αποτέλεσμα να παγιδεύεται σε τοπικά ακρότατα. Οι γενετικοί αλγόριθμοι συνδυάζουν με τον καλύτερο τρόπο την αναζήτηση και την εκμετάλλευση της πληροφορίας από τα προηγούμενα βήματα.
- Είναι εύκολα επεκτάσιμοι και εξελίξιμοι. Οι γενετικοί αλγόριθμοι στην κλασική τους μορφή είναι αρκετά αργοί και δεν εγγυώνται την εύρεση βέλτιστης λύσης. Αυτό όμως δεν τους εμποδίζει να χρησιμοποιούνται για την επίλυση μεγάλου αριθμού προβλημάτων, καθώς μπορούν πολύ εύκολα να επεκταθούν, ακόμα και με λειτουργίες που δεν βασίζονται στη φύση.
- Μπορούν να χρησιμοποιηθούν στη βελτιστοποίηση με πολλαπλά κριτήρια (Abdullah Konak, 2006).
- Το σημαντικότερο πλεονέκτημα τον γενετικών αλγορίθμων, είναι ότι η μόνο γνώση που χρειάζονται για το πρόβλημα είναι η συνάρτηση καταλληλότητας. Για παράδειγμα έστω ότι θέλουμε να βρούμε το ελάχιστο της συνάρτησης Rosenbrock για δύο μεταβλητές. Οι περισσότεροι ευρετικοί αλγόριθμοι, αν δεν ξεκινήσουμε από μια καλή λύση όπως η (-1,1) δε θα δώσουν καλό αποτέλεσμα. Αντίθετα ένας γενετικός αλγόριθμος θα δώσει μια αρκετά καλή λύση όποιος και να είναι ο αρχικός πληθυσμός ακόμα και για μεγάλο αριθμό μεταβλητών.

2.4 Μειονεκτήματα Γ.Α.

Οι γενετικοί αλγόριθμοι έχουν και αρκετά μειονεκτήματα και ανάλογα με τις απαιτήσεις του χρήστη μπορεί να χρειαστεί να καταφύγουμε σε κάποια διαφορετική μέθοδο επίλυσης του προβλήματος.

- Δεν εγγυώνται ότι θα βρουν τη βέλτιστη λύση ακόμα και μετά από άπειρο χρόνο.
- Για τη σωστή λειτουργία ενός γενετικού αλγορίθμου απαιτείται η ρύθμιση μεγάλου αριθμού παραμέτρων με τις ρυθμίσεις να είναι διαφορετικές σε κάθε πρόβλημα. Για παράδειγμα πρέπει να ρυθμίσουμε το μέγεθος του αρχικού πληθυσμού, τον τρόπο με τον οποίο θα γίνει η διασταύρωση, τον τρόπο με τον οποίο θα γίνει η μετάλλαξη, το ποσοστό μετάλλαξης, το ποσοστό διασταύρωσης, αν θα χρησιμοποιηθεί ελιτισμός, με πιο τρόπο θα γίνει η επιλογή κτλ.
- Χρειάζονται μεγαλύτερη υπολογιστική ισχύ σε σχέση με άλλους ευρετικούς αλγορίθμους. Σε πολλά προβλήματα ο υπολογισμός της καταλληλότητας είναι μια χρονοβόρα διαδικασία. Ο γενετικός αλγόριθμος πρέπει σε κάθε κύκλο να υπολογίσει την καταλληλότητα για όλα τα άτομα του πληθυσμού.
- Σε μερικά προβλήματα η εύρεση της σωστής συνάρτησης καταλληλότητας ή ακόμα και της αναπαράστασης που πρέπει να χρησιμοποιηθεί, είναι μια αρκετά δύσκολη διαδικασία.

- ❖ Δεν κάνουν χρήση συγκεκριμένων πληροφοριών για κάποιο πρόβλημα. Αυτό τους δίνει την δυνατότητα να μπορούν να χρησιμοποιηθούν σε μεγάλο αριθμό προβλημάτων, όμως τους κάνει λιγότερο αποδοτικούς σε σχέση με άλλους ευρετικούς αλγορίθμους που είναι σχεδιασμένοι για ένα συγκεκριμένο πρόβλημα.
- ❖ Στους γενετικούς αλγορίθμους αν δεν έχει γίνει σωστή παραμετροποίηση, μπορεί πολύ εύκολα να εμφανιστεί το φαινόμενο της πρόωρης σύγκλισης. Τα άτομα με υψηλή καταλληλότητα θα αρχίσουν να παράγουν αντίγραφα του εαυτού τους και η μόνη έξοδος του γενετικού αλγορίθμου από το τοπικό ακρότατο είναι η μετάλλαξη. Η μετάλλαξη όμως έχει πολύ μικρή πιθανότητα να συμβεί και όταν συμβεί θα πρέπει να οδηγήσει τον αλγόριθμο σε μια καλύτερη λύση. Κάτι τέτοιο απαιτεί πολλούς κύκλους και πρακτικά σημαίνει ότι ο αλγόριθμος έχει εγκλωβιστεί.

2.5 Προετοιμασία

Ένα από τα βασικά μειονεκτήματα των γενετικών αλγορίθμων είναι ότι δεν έχουν αποδειχθεί μαθηματικά. Αυτό έχει ως αποτέλεσμα κατά την υλοποίηση ο σχεδιαστής να πρέπει να ρυθμίσει ένα μεγάλο αριθμό παραμέτρων.

- ➢ Με πιο τρόπο θα γίνει η κωδικοποίηση των μεταβλητών του προβλήματος. Τα πρώτα χρόνια οι γενετικοί αλγόριθμοι χρησιμοποιούσαν αποκλειστικά τη δυαδική κωδικοποίηση, αλλά όπως θα δούμε αναλυτικά στη συνέχεια, η ιδέα αυτή θεωρείται πλέον ξεπερασμένη.
- ➢ Από πόσα χρωμοσώματα θα αποτελείται ένα άτομο. Συνήθως ένα άτομο έχει ένα χρωμόσωμα.
- ➢ Πιο θα είναι το μέγεθος του αρχικού πληθυσμού. Από κάποιο σημείο και μετά η αύξηση του μεγέθους του πληθυσμού, δεν αυξάνει την απόδοση του Γ.Α..
- ➢ Πως θα δημιουργηθεί ο αρχικός πληθυσμός. Συνήθως η δημιουργία του αρχικού πληθυσμού γίνεται τυχαία, αλλά μπορεί να χρησιμοποιηθεί και κάποιος γρήγορος ευρετικός αλγόριθμος.
- ➢ Ποια θα είναι η συνάρτηση καταλληλότητας. Σε απλά προβλήματα βελτιστοποίησης η συνάρτηση καταλληλότητας είναι η ίδια η αντικειμενική συνάρτηση, αλλά σε κάποια άλλα μπορεί να μην υπάρχει καν συνάρτηση καταλληλότητας, ή ο υπολογισμός της καταλληλότητας να είναι πολύ χρονοβόρος και να πρέπει να χρησιμοποιήσουμε μια προσεγγιστική συνάρτηση καταλληλότητας.
- ➢ Με πιο τρόπο θα γίνει η επιλογή. Ανάλογα με τον αλγόριθμο επιλογής που θα χρησιμοποιήσουμε, αλλάζει και η πίεση της επιλογής.
- ➢ Με πιο τρόπο θα γίνει η διασταύρωση των χρωμοσωμάτων. Η επιλογή του κατάλληλου τελεστή διασταύρωσης παίζει το σημαντικότερο ρόλο στη σωστή λειτουργία ενός γενετικού αλγορίθμου.
- ➢ Επιλογή της πιθανότητας διασταύρωσης. Διασταύρωση δε συμβαίνει σε όλα τα χρωμοσώματα αλλά σε ένα ποσοστό (συνήθως μεγάλο).
- ➢ Με πιο τρόπο θα γίνει η μετάλλαξή. Όπως και στη διασταύρωση, υπάρχουν πάρα πολλοί αλγόριθμοι μετάλλαξης.
- ➢ Επιλογή της πιθανότητας μετάλλαξης. Όπως και στη διασταύρωση, η μετάλλαξη συμβαίνει σε ένα ποσοστό χρωμοσωμάτων/γονιδίων. Το ποσοστό είναι συνήθως μικρό καθώς ένα μεγάλο ποσοστό μετάλλαξης θα μετέτρεπε το γενετικό αλγόριθμο, σε έναν αλγόριθμο τυχαίας αναζήτησης.
- ➢ Επιλογή κριτηρίου τερματισμού. Συνήθως ο αλγόριθμος τερματίζει μετά από ένα συγκεκριμένο αριθμό επαναλήψεων.
- ➢ Επιλογή επιπρόσθετων παραμέτρων όπως δυναμικό μέγεθος πληθυσμού, δυναμική πιθανότητα διασταύρωσης/μετάλλαξης, χρήση επιλογής επιζώντων, ελιτισμός κτλ.

2.6 Βήματα απλού γενετικού αλγορίθμου

Σε αυτό το κεφάλαιο θα δούμε τα βήματα του απλού γενετικού αλγορίθμου [(D.E., 1989) σελ. 59].

Βήμα 1ο: Καθορίζουμε τις παραμέτρους του γενετικού αλγορίθμου δηλαδή το μέγεθος του αρχικού πληθυσμού, την πιθανότητα διασταύρωσης, την πιθανότητα μετάλλαξης και τον αριθμό των επαναλήψεων. Επίσης υπολογίζουμε τη συνάρτηση καταλληλότητας η οποία για απλά προβλήματα βελτιστοποίησης είναι η αντικειμενική συνάρτηση. Ο απλός γενετικός αλγόριθμος χρησιμοποιεί τη δυαδική κωδικοποίηση και όλες οι μεταβλητές κωδικοποιούνται σε ένα χρωμόσωμα.

Βήμα 2ο: Αρχικοποίηση (Initialization). Δημιουργούμε τυχαία τον αρχικό πληθυσμό.

Βήμα 3ο: Αξιολόγηση (Evaluation). Αξιολογούμε όλα τα άτομα του πληθυσμού με τη συνάρτηση καταλληλότητας.

Βήμα 4ο: Επιλογή (Selection). Με βάση την απόδοση των ατόμων που προέκυψε από το Βήμα 3, δημιουργούμε ένα προσωρινό πληθυσμό. Κάθε άτομο έχει πιθανότητα να επιλεγεί:

$$P^{i}_{select} = \frac{f(pop_i)}{\sum_{j=1}^{N} f(pop_j)}$$

Εικόνα 14 Πιθανότητα επιλογής ατόμου

Όπου N το μέγεθος του πληθυσμού και $f(pop_i)$ είναι η απόδοση του ατόμου i του πληθυσμού.

Βήμα 5ο: Διασταύρωση (Crossover). Στον απλό γενετικό αλγόριθμο, η διασταύρωση είναι η λεγόμενη διασταύρωση ενός σημείου. Δύο άτομα επιλέγονται τυχαία με πιθανότητα ίση με την πιθανότητα διασταύρωσης και χωρίζονται τυχαία σε δύο κομμάτια. Το πρώτο κομμάτι του πρώτου ατόμου ενώνεται με το δεύτερο κομμάτι του δεύτερου ατόμου και το πρώτο κομμάτι του δεύτερου ατόμου ενώνεται με το δεύτερο κομμάτι του πρώτου ατόμου, σχηματίζοντας δύο νέα άτομα.

Βήμα 6ο: Μετάλλαξη (mutation). Στον απλό γενετικό αλγόριθμο με τον όρο μετάλλαξη εννοούμε την αντιστροφή ενός bit από 0 σε 1 ή από 1 σε 0. Κάθε bit αντιστρέφεται με πιθανότητα ίση με την πιθανότητα μετάλλαξης.

Βήμα 7ο: Τερματισμός ή επανάληψη (Termination or repeat). Αν ο αριθμός των επαναλήψεων ξεπεράσει το όριο που έχει δοθεί στην αρχή από τον χρήστη, ο αλγόριθμος τερματίζει. Αν όχι ο αλγόριθμος επιστρέφει στο Βήμα 3.

3 Αναπαράσταση - Κωδικοποίηση - Representation – Encoding

Το πρώτο βήμα σε έναν Γ.Α. είναι η γενετική αναπαράσταση των υποψήφιων λύσεων, δηλαδή κωδικοποίηση τους με ένα τρόπο ο οποίος να μπορεί να γίνει κατανοητός από τον υπολογιστή. Για παράδειγμα ο υπολογιστής δεν μπορεί να καταλάβει την πρόταση “από τους πέντε διακόπτες ο διακόπτης δύο και ο διακόπτης τρία είναι στη θέση ΟΝ” αλλά μπορεί να καταλάβει τη συμβολοσειρά (0 1 1 0 0).

Όπως είπαμε στην παράγραφο “Ορολογία” κάθε πιθανή λύση (άτομο) αποτελείται από ένα ή και περισσότερα χρωμοσώματα. Επειδή συνήθως κάθε άτομο αποτελείται από ένα χρωμόσωμα, με τους όρους άτομο και χρωμόσωμα θα εννοούμε το ίδιο. Το χρωμόσωμα αποτελείται από γονίδια και το κάθε γονίδιο περιέχει κάποια πληροφορία για τη λύση.

Ο τρόπος με τον οποίο θα μετατρέψουμε τη λύση από κάτι που δεν είναι κατανοητό στον υπολογιστεί σε μια σειρά από γονίδια, τα οποία θα μπορεί να κατανοήσει και να επεξεργαστεί ο υπολογιστής, ονομάζεται αναπαράσταση της λύσης. Η κωδικοποίηση που θα επιλέξουμε εξαρτάται από το πρόβλημα, τους περιορισμούς του και την πείρα του σχεδιαστή (Sumati Jaggi, 2013) (García-Hernández, Araúzo-Azofra, & Salas-Morera, 2009) (Kumar, 2013). Στα περισσότερα προβλήματα μπορούμε να εφαρμόσουμε περισσότερες από μία μορφές κωδικοποίησης και η επιλογή της σωστής θα επηρεάσει σε μεγάλο βαθμό την απόδοση του γενετικού αλγορίθμου.

Για την επιλογή της σωστής κωδικοποίησης δεν υπάρχει κάποιος κανόνας. Στα περισσότερα προβλήματα χρησιμοποιείται η δυαδική κωδικοποίηση γιατί σύμφωνα με τον Goldberg [(D.E., 1989) σελ. 80] και την υπόθεση των δοκιμών στοιχείων "building block hypothesis" που θα αναλύσουμε στη συνέχεια, παράγει τον μεγαλύτερο αριθμό σχημάτων.

Εκτός από τον μεγάλο αριθμό σχημάτων, αν το πρόβλημα έχει περιορισμούς, η ιδανική κωδικοποίηση θα πρέπει να μας δίνει λύσεις που ικανοποιούν αυτούς τους περιορισμούς. Αν δε συμβαίνει αυτό, τότε θα πρέπει με κάποιο τρόπο να απορρίψουμε αυτές τις λύσεις. Αυτό μπορεί να γίνει με τους εξής τρόπους:

- ***Penalty functions**:* Οι λύσεις που παραβιάζουν κάποιους περιορισμούς δέχονται ποινή στην καταλληλότητά τους, ανάλογα με τον αριθμό των περιορισμών που παραβίασαν. Έτσι σιγά σιγά μέσα από τη διαδικασία της επιλογής διαγράφονται από τον πληθυσμό. Για παράδειγμα αν ο γενετικός αλγόριθμος δημιουργεί το πρόγραμμα του εξαμήνου ενός πανεπιστημίου και μια από τις λύσεις έχει την ίδια ώρα δύο μαθήματα στην ίδια αίθουσα, τότε λαμβάνει μια ποινή στην καταλληλότητά της. Αν η λύση έχει τρία μαθήματα στην ίδια αίθουσα την ίδια ώρα, τότε λαμβάνει ακόμα μεγαλύτερη ποινή.
- **Repair functions**: Όταν κάποιος από τους τελεστές διασταύρωσης ή μετάλλαξης, μας δώσει μια λύση που δεν ικανοποιεί τους περιορισμούς, τότε εφαρμόζεται στη λύση μια συνάρτηση διόρθωσης. Για παράδειγμα αν χρησιμοποιηθεί η δυαδική κωδικοποίηση για το πρόβλημα του πλανόδιου πωλητή, τότε μπορεί να προκύψουν διαδρομές στις οποίες μια πόλη θα εμφανίζεται περισσότερες από μία φορές και κάποιες πόλεις δε θα εμφανίζονται καθόλου.
- **Οι λύσεις που δεν ικανοποιούν τους περιορισμού, δεν επιτρέπεται να εισέλθουν στον πληθυσμό**: Με το που θα δημιουργηθεί μια τέτοια λύση ο γενετικός αλγόριθμος την απορρίπτει και είτε δημιουργεί μια νέα, είτε χρησιμοποιεί τα/το άτομα/ο που την δημιούργησαν στη νέα γενιά.

Προφανώς και οι τρεις διαδικασίες που αναφέραμε είναι αρκετά χρονοβόρες, οπότε είναι καλύτερο να επιλέξουμε μια κωδικοποίηση η οποία θα παράγει λύσεις που θα ικανοποιούν πάντα τους περιορισμούς του προβλήματος, αν αυτό είναι δυνατό.

Σύμφωνα με τον Goldberg [(D.E., 1989) σελ. 80] η κωδικοποίηση που θα επιλεγεί πρέπει να είναι τέτοια ώστε τα σχήματα μικρής τάξης και μικρού μήκους, να είναι σχετικά με το πρόβλημα και να σχετίζονται όσο το δυνατό λιγότερο με τα σχήματα στις άλλες σταθερές θέσεις. Αυτό που μας λέει αυτή η αρχή είναι το εξής. Έστω ότι θέλουμε να κωδικοποιήσουμε την πληροφορία "μπλε μάτια". Η κωδικοποίηση θα πρέπει να γίνει με όσο το δυνατό λιγότερα γονίδια και τα γονίδια αυτά να βρίσκονται κοντά το ένα στο άλλο, ώστε όταν γίνει η ανταλλαγή γενετικού υλικού, να μη διαχωριστούν. Για παράδειγμα αν το χρώμα ματιών μπορεί να πάρει τέσσερις τιμές, δυο bit είναι αρκετά για να αποθηκεύσουν αυτή την πληροφορία και θα πρέπει να βρίσκονται το ένα δίπλα στο άλλο μέσα στο χρωμόσωμα. Η τήρηση αυτής της αρχής είναι πολύ δύσκολη.

Στη συνέχεια θα αναλύσουμε τις βασικότερες μορφές κωδικοποίησης που χρησιμοποιούνται στους γενετικούς αλγορίθμους.

3.1 Δυαδική κωδικοποίηση - Binary encoding

Ο κλασικός γενετικός αλγόριθμος (Holland, 1975) χρησιμοποιεί τη δυαδική κωδικοποίηση. Για πολλά χρόνια ήταν η μοναδική μορφή κωδικοποίησης αλλά σταδιακά οι ερευνητές άρχισαν να πειραματίζονται με άλλες μορφές κωδικοποίησης οι οποίες σε κάποια προβλήματα είχαν καλύτερα αποτελέσματα (C. Janikow, 1991). Στη δυαδική αναπαράσταση η υποψήφια λύση κωδικοποιείται σε μια συμβολοσειρά από bit και κάθε bit είναι ένα γονίδιο του χρωμοσώματος.

Παραδείγματα δυαδικής αναπαράστασης στους Γ.Α.

- **0-1 knapsack problem [Παράρτημα 1].** Κάθε αντικείμενο μπορεί να πάρει την τιμή 0 ή 1. Για παράδειγμα αν έχουμε 7 αντικείμενα η συμβολοσειρά (χρωμόσωμα) 0100101 σημαίνει ότι το σακίδιο περιέχει το δεύτερο, το πέμπτο και το έβδομο αντικείμενο.

- **Μεγιστοποίηση - ελαχιστοποίηση συνάρτησης**. Θέλουμε να βρούμε το μέγιστο της συνάρτησης $f(x,y) = 5 + x * \sin(\pi * y) + y * \cos(5 * \pi * y)$ όταν $-3 \leq x \leq 4.5$ και $3.1 \leq y \leq 5.2$ με ακρίβεια δύο δεκαδικών ψηφίων. Για την αναπαράσταση της μεταβλητής x με ακρίβεια δύο δεκαδικών ψηφίων θα πρέπει να χωρίσουμε το διάστημα [-3, 4.5] σε 7.5*10^2 = 750 ίσα υπό-διαστήματα. Για να το πετύχουμε αυτό χρειαζόμαστε 10 bits αφού $2^9 < 750 \leq 2^{10}$. Ομοίως για την μεταβλητή y χρειαζόμαστε 8 bits. Αν ο Γ.Α. χρησιμοποιεί ένα χρωμόσωμα τότε η συμβολοσειρά 1111111111 | 00000000 αντιστοιχεί στους αριθμούς x=4.5 και y=3.1[Παράρτημα 2].

Χρωμόσωμα 1	1010001001110100110010 1
Χρωμόσωμα 2	1010110100110000010110 0

Εικόνα 15 Δυαδική κωδικοποίηση

Τα **πλεονεκτήματα** της δυαδικής κωδικοποίησης είναι τα εξής:

- Παράγουν τον μεγαλύτερο αριθμό σχημάτων (περισσότερα για τα σχήματα στο κεφάλαιο "θεωρία σχημάτων"). Σύμφωνα με τον Holland (Holland, 1975), οι γενετικοί αλγόριθμοι δεν επεξεργάζονται άτομα, αλλά τα σχήματα που αντιστοιχούν σε αυτά. Επομένως εφόσον στη δυαδική κωδικοποίηση έχουμε το μέγιστο αριθμό σχημάτων, θα έχουμε και τη μέγιστη παράλληλη επεξεργασία πληροφορίας για κάθε άτομο (implicit parallelism). Για παράδειγμα στο σχήμα που ακολουθεί αν χρησιμοποιήσουμε την οκταδική κωδικοποίηση δεν υπάρχει καμία ομοιότητα μεταξύ των τεσσάρων ατόμων. Δεν μπορούμε να εξηγήσουμε γιατί το άτομο 000 έχει την καλύτερη απόδοση. Με την δυαδική κωδικοποίηση παρατηρούμε ότι τα άτομα 000 και 010 ανήκουν στο σχήμα 0** και τα άτομα 000 και 101 στο σχήμα *0*.

Binary	Octal	Fitness
000	0	22
011	3	8
101	5	11
111	7	3

Εικόνα 16 Δυαδική και οκταδική κωδικοποίηση

- Η δυαδική αναπαράσταση μπορεί να χρησιμοποιηθεί σε όλα τα προβλήματα. Ο σχεδιαστής του Γ.Α. δε χρειάζεται να αφιερώσει χρόνο ψάχνοντας την αναπαράσταση που πρέπει να χρησιμοποιήσει, ούτε να δημιουργήσει καινούργιους τελεστές διασταύρωσης και μετάλλαξης.
- Ο Reeves (1993) (Reeves, 1993) στην έρευνά του προσπάθησε να υπολογίσει το ελάχιστο μέγεθος του πληθυσμού που πρέπει να έχει ο γενετικός αλγόριθμος και κατέληξε στο συμπέρασμα ότι από έναν αρχικό πληθυσμό που δημιουργήθηκε τυχαία, ο Γ.Α. πρέπει να μπορεί να πάει σε όλες τις λύσεις, χρησιμοποιώντας μόνο τον τελεστή της διασταύρωσης. Αυτό έχει ως επακόλουθο οι γενετικοί αλγόριθμοι να χρειάζονται μικρότερο μέγεθος πληθυσμού, με αποτέλεσμα όταν ο υπολογισμός της συνάρτησης καταλληλότητας είναι πολύ χρονοβόρος να αποδίδουν καλύτερα (F. Herrera M. L., 1998).

Τα **μειονεκτήματα** της δυαδικής κωδικοποίησης είναι τα εξής:

- Υπάρχει το λεγόμενο hamming cliffs problem για το οποίο θα μιλήσουμε αναλυτικά στο κεφάλαιο της "κωδικοποίησης με αριθμούς κινητής υποδιαστολής". Δύο γειτονικές λύσεις μπορεί να έχουν όλα τα bit διαφορετικά. Για παράδειγμα οι αριθμός 31 (01111) και 32

(10000). Για να αποφύγουμε το hamming cliffs problem αντί για τη δυαδική κωδικοποίηση, μπορεί να χρησιμοποιηθεί ο αντίστοιχος κώδικας Gray, αν και από έρευνες προκύπτει (Janikow) ότι δεν έχουν σημαντικές διαφορές.

- Η ακρίβεια εξαρτάται από τον αριθμό των bit που χρησιμοποιήθηκαν στην αναπαράσταση.
- Μεγάλος μέγεθος χρωμοσώματος σε προβλήματα στα οποία οι μεταβλητές είναι πραγματικοί αριθμοί. Αν έχουμε μια μεταβλητή $x \in [-500,500]]$ και θέλουμε ακρίβεια 6 δεκαδικών ψηφίων, το χρωμόσωμα θα έχει 30 γονίδια (bits). Σύμφωνα με τους Nicol N. Schraudolph και Richard K. Belew η απόδοση του γενετικού αλγορίθμου σε αυτή την περίπτωση δεν είναι καλή. Για τον γενετικό αλγόριθμο όλα τα bit έχουν την ίδια αξία με αποτέλεσμα ο γενετικός αλγόριθμος να σπαταλάει πόρους προσπαθώντας να συγκλίνει τα λιγότερο σημαντικά bit.
- Μεγάλος μέγεθος χρωμοσώματος σε προβλήματα στα οποία παίζει ρόλο η σειρά. Για παράδειγμα στο πρόβλημα του πλανόδιου πωλητή, αν έχουμε n πόλεις κάθε πόλη χρειάζεται $log_2 n + 1$ bits για την αναπαράστασή της. Αν για παράδειγμα έχουμε 4 πόλεις θέλουμε 2 bits και η αναπαράσταση της κάθε πόλης είναι 00, 01, 10, 11. Αν το χρωμόσωμα αποτελεί μια διαδρομή μεταξύ όλων των πόλεων τότε θα για κάθε χρωμόσωμα χρειαζόμαστε $n * (\lfloor log_2 n \rfloor + 1)$ bits. Για τις 4 πόλεις και τη διαδρομή 2->3->4->1 το χρωμόσωμα θα είναι το 01101100.
- Εμφάνιση μη έγκυρων λύσεων. Για παράδειγμα στο πρόβλημα του πλανόδιου πωλητή, αν το πλήθος των πόλεων δεν είναι δύναμη του 2 τότε μέσα στο χρωμόσωμα θα υπάρχουν ακολουθίες δυαδικών ψηφίων που δε θα αντιστοιχούν σε κάποια πόλη. Για 5 πόλεις χρειαζόμαστε 3 δυαδικά ψηφία. 000 001 010 011 100. Αν όμως μετά την εφαρμογή των τελεστών προκύψει το χρωμόσωμα 001|010|000|011|101 η ακολουθία των 3 τελευταίων δυαδικών ψηφίων 101 δεν αντιστοιχεί σε κάποια πόλη. Όποιον από τους τρεις τρόπους που είδαμε στην αρχή του κεφαλαίου για την αντιμετώπιση του προβλήματος της εμφάνισης μη έγκυρων λύσεων και να χρησιμοποιήσουμε, επιβαρύνουμε το γενετικό αλγόριθμο. Είναι καλύτερα να επιλέξουμε μια κωδικοποίηση η οποία θα παράγει λύσεις που θα ικανοποιούν πάντα τους περιορισμούς του προβλήματος, αν αυτό είναι δυνατό.
- Εμφάνιση μη έγκυρων χρωμοσωμάτων σε προβλήματα μεταθέσεων. Σε αυτά τα προβλήματα κάθε γονίδιο πρέπει να εμφανίζεται ακριβώς μια φορά. Για παράδειγμα στο πρόβλημα του πλανόδιου πωλητή, κατά την εφαρμογή των τελεστών διασταύρωσης και μετάλλαξης κάποια πόλη μπορεί να εμφανιστεί περισσότερες από μία φορές (και κατ' επέκταση κάποια πόλη δε θα εμφανίζεται καθόλου). Αν για παράδειγμα κατά τη μετάλλαξη το χρωμόσωμα 01101100 αλλάξει σε 11101100 τότε παίρνουμε τη διαδρομή 4->3->4->1 στην οποία η πόλη τέσσερα εμφανίζεται δύο φορές και η πόλη δύο καμία, άρα η διαδρομή δεν είναι έγκυρη.

3.2 Οκταδική κωδικοποίηση - Octal Encoding

Η οκταδική κωδικοποίηση είναι παρόμοια με την δυαδική, μόνο που αντί για τα σύμβολα (0,1) χρησιμοποιούνται τα σύμβολα (0,1,2,3,4,5,6,7). Στην πράξη πολύ σπάνια ένας γενετικός αλγόριθμος χρησιμοποιεί οκταδική κωδικοποίηση.

Χρωμόσωμα	1034226701027765523443

Εικόνα 17 Οκταδική κωδικοποίηση

3.3 Δεκαεξαδική κωδικοποίηση - Hexadecimal encoding

Η δεκαεξαδική κωδικοποίηση είναι παρόμοια με την δυαδική, μόνο που αντί για τα σύμβολα (0,1) χρησιμοποιούνται τα σύμβολα (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Στην πράξη πολύ σπάνια ένας γενετικός αλγόριθμος χρησιμοποιεί δεκαεξαδική κωδικοποίηση.

Χρωμόσωμα	F1089AE6523BC4D7FA189

Εικόνα 18 Δεκαεξαδική κωδικοποίηση

3.4 Κωδικοποίηση με ακέραιους αριθμούς - Integer encoding

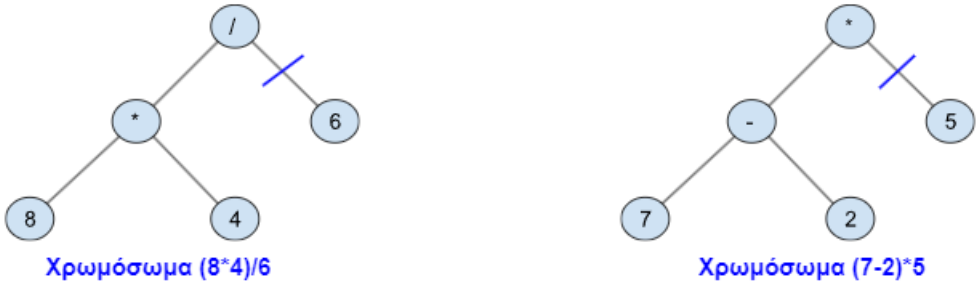
Η κωδικοποίηση με ακέραιους αριθμούς είναι υποπερίπτωση της κωδικοποίησης τιμών που θα αναλύσουμε στη συνέχεια. Τα γονίδια του χρωμοσώματος είναι ακέραιοι αριθμοί. Χρησιμοποιείται συχνά σε προβλήματα ανάλυσης εικόνας. Ένα από τα πλεονεκτήματά της είναι ότι μπορούν να χρησιμοποιηθούν οι κλασικοί τελεστές διασταύρωσης και μετάλλαξης που χρησιμοποιούνται στη δυαδική κωδικοποίηση.

Χρωμόσωμα	1 3 8 7 0 10 28 7 6 234 8 31

Εικόνα 19 Αναπαράσταση με ακέραιους αριθμούς

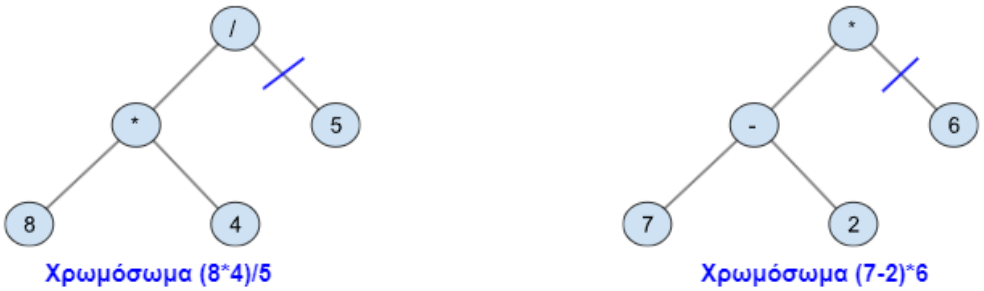
3.5 Κωδικοποίηση δέντρου - Tree encoding

Στην κωδικοποίηση δέντρου κάθε χρωμόσωμα είναι ένα δέντρο αντικειμένων με τα αντικείμενα να είναι συναρτήσεις ή/και εντολές σε κάποια γλώσσα προγραμματισμού. Συνήθως χρησιμοποιείται στο γενετικό προγραμματισμό. Σε αυτή την περίπτωση το χρωμόσωμα είναι ένα δέντρο αντικειμένων, με τα φύλλα του δέντρου να είναι ένας αριθμός ή μια μεταβλητή και οι εσωτερικοί κόμβοι του δέντρου μια συνάρτηση. Για παράδειγμα οι συναρτήσεις (8*4)/6 και (7-2)*5 αντιστοιχούν στα χρωμοσώματα:



Εικόνα 20 Παράδειγμα κωδικοποίησης δέντρου

Μια πιθανή παραλλαγή της διασταύρωσης ενός σημείου για την κωδικοποίηση δέντρου φαίνεται στο παρακάτω σχήμα:



Εικόνα 21 Παραλλαγή διασταύρωσης ενός σημείου για κωδικοποίηση δέντρου

3.6 Κωδικοποίηση τιμών - Value encoding

Η συγκεκριμένη κωδικοποίηση χρησιμοποιείται κυρίως σε προβλήματα που υπάρχουν πολύπλοκες τιμές, όπως οι πραγματικοί αριθμοί. Η κωδικοποίηση ενός τέτοιο προβλήματος με τη χρήση της δυαδικής κωδικοποίησης είναι πολύ δύσκολη.

Τη συγκεκριμένη κωδικοποίηση τη συναντάμε συχνά στην αναζήτηση των βαρών ενός νευρωνικού δικτύου (Montana, 1995). Σε αυτή την περίπτωση κάθε γονίδιο του χρωμοσώματος είναι ένας πραγματικός αριθμός.

Γενικά στην κωδικοποίηση τιμών κάθε χρωμόσωμα είναι μια ακολουθία τιμών (γονιδίων) άμεσα σχετιζόμενων με το προς βελτιστοποίηση πρόβλημα, όπως ακέραιοι, πραγματικοί αριθμοί, χαρακτήρες κτλ. Στις περισσότερες περιπτώσεις όταν χρησιμοποιήσουμε την κωδικοποίηση τιμών θα πρέπει να δημιουργήσουμε και τους ανάλογους τελεστές διασταύρωσης και μετάλλαξης. Μερικά παραδείγματα κωδικοποίησης τιμών είναι τα εξής:

Χρωμόσωμα	12345678123901929399 49
Χρωμόσωμα	ABCDEFABGHKLMNWXP
Χρωμόσωμα	(πάνω) (δεξιά) (αριστερά)
Χρωμόσωμα	(1001) (001) (1110) (00101)
Χρωμόσωμα	6.578 13.558 22.142 4.567

Εικόνα 22 Παραδείγματα κωδικοποίησης τιμών

Σημείωση: Σε κάποια βιβλία όταν τα γονίδια της κωδικοποίησης τιμών αντιστοιχούν σε ακέραιους η κωδικοποίηση ονομάζεται κωδικοποίηση με ακέραιους αριθμούς. Αντίστοιχα όταν τα γονίδια αντιστοιχούν σε δεκαδικούς αριθμούς η κωδικοποίηση ονομάζεται κωδικοποίηση με αριθμούς κινητής υποδιαστολής.

3.7 Κωδικοποίηση με αριθμούς κινητής υποδιαστολής - Float / Real value encoding

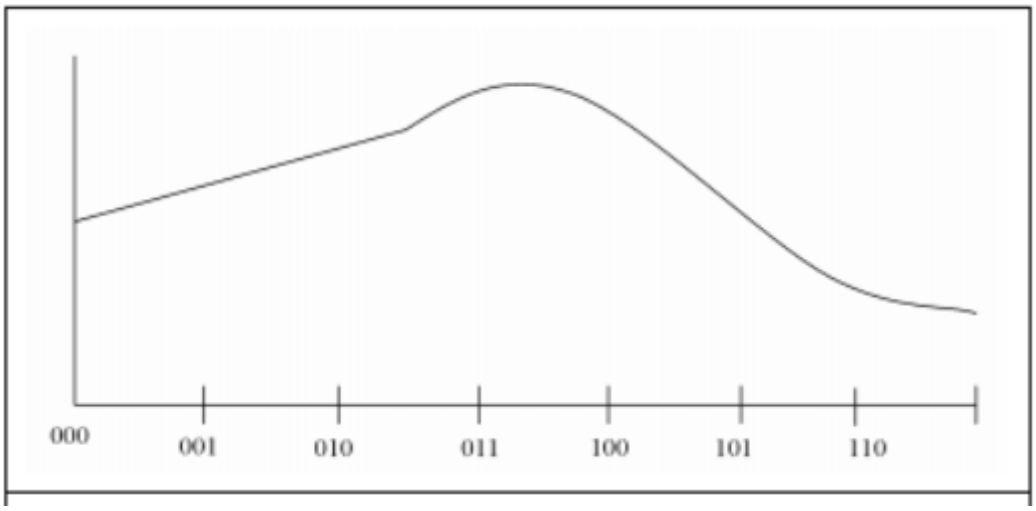
Υποπερίπτωση της κωδικοποίησης τιμών όπου τα γονίδια του χρωμοσώματος είναι δεκαδικοί αριθμοί. Η μέθοδος αυτή έχει πολύ καλά αποτελέσματα σε προβλήματα συνεχών (μη διακριτών) τιμών και γενικά σε προβλήματα όπου οι μεταβλητές των λύσεων είναι πραγματικοί αριθμοί (L. Budin, 2000) (Chuangyin Dang, 2007) (C. Janikow, 1991). Η ακρίβεια εξαρτάται από το μέγεθος του αριθμού κινητής υποδιαστολής που μπορεί να χειριστεί ο υπολογιστής.

Χρωμόσωμα	16.578 13.558 22.142 14.567

Εικόνα 23 Αναπαράσταση με αριθμούς κινητής υποδιαστολής

Οι λόγοι για τους οποίους αρκετές φορές προτιμάμε την αναπαράσταση με αριθμούς κινητής υποδιαστολής και γενικά μια κωδικοποίηση με μεγαλύτερο αριθμό συμβόλων από τη δυαδική είναι οι εξής (Goldberg D. E., Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking, 1991):

- Είναι πιο εύκολο για τον άνθρωπο να καταλάβει την αντιστοίχιση μιας μεταβλητής σε ένα γονίδιο. Για παράδειγμα όταν προσπαθούμε να βρούμε τα βάρη σε ένα νευρωνικό δίκτυο, είναι πιο εύκολο να αντιστοιχίσουμε κάθε βάρος σε ένα γονίδιο που θα έχει ως τιμή έναν πραγματικό αριθμό.
- Με τη χρήση της δυαδικής κωδικοποίησης προκύπτει το λεγόμενο Hamming cliff problem. Ως απόσταση Χάμινγκ (Hamming distance) μεταξύ δύο συμβολοσειρών ίσου μήκους ορίζεται ο αριθμός θέσεων στις οποίες τα αντίστοιχα σύμβολα είναι διαφορετικά. Το πρόβλημα Hamming cliff προκύπτει όταν δύο γειτονικές λύσεις π χ το 31 και το 32 που με δυαδική κωδικοποίηση γράφονται 01111 και 10000 αντίστοιχα, έχουν την μέγιστη απόσταση Χάμινγκ (στην περίπτωσή μας 5 καθώς όλα τα bit είναι διαφορετικά). Σε αυτή την περίπτωση ο αλγόριθμος δυσκολεύεται να πάει από την μια στην άλλη. Για παράδειγμα στο σχήμα που ακολουθεί παρατηρούμε ότι ο μέσος όρος της καταλληλότητας των λύσεων που αρχίζουν με 0 είναι μεγαλύτερος από τον μέσο όρο της καταλληλότητας των λύσεων που αρχίζουν με 1 $\overline{f}(0**) > \overline{f}(1*0)$. Επομένως είναι πολύ πιθανό ο αλγόριθμος αρχικά να συγκλίνει στην τιμή 011. Από εκεί είναι πολύ δύσκολο να πάει στο ολικό μέγιστο που είναι το 100 καθώς έχουν απόσταση Χάμινγκ 3. Ο αλγόριθμος θα εγκλωβιστεί στο τοπικό ακρότατο 011.



Εικόνα 24 Hamming cliff problem

- Μπορεί να αποδειχθεί θεωρητικά, αλλά και έχει αποδειχθεί και πρακτικά ότι τα αλφάβητα με μεγαλύτερο αριθμό συμβόλων, συγκλίνουν γρηγορότερα σε μια λύση. Για τους γενετικούς αλγορίθμους που χρησιμοποιούν αριθμούς κινητής υποδιαστολής αυτό μπορεί

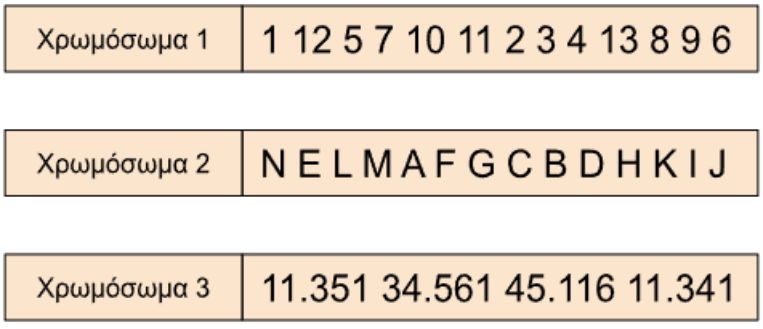
να θεωρηθεί ως πλεονέκτημα, αν στόχος μας είναι η ταχύτητα, ή μειονέκτημα λόγω της πιθανής πρόωρης σύγκλισης.

- Σε προβλήματα συνεχών (μη διακριτών) τιμών, η κωδικοποίηση με αριθμούς κινητής υποδιαστολής, έχει μεγαλύτερη ακρίβεια από τη δυαδική κωδικοποίηση.
- Δε χρειάζεται να κάνουμε αποκωδικοποίηση του χρωμοσώματος κάθε φορά που υπολογίζουμε την καταλληλότητά του.
- Η τυπική απόκλιση είναι μικρότερη σε σχέση με τη δυαδική κωδικοποίηση, δηλαδή σε κάθε εκτέλεση το αποτέλεσμα του γενετικού αλγορίθμου θα είναι παρόμοιο (C. Janikow, 1991).
- Μπορούν να χρησιμοποιηθούν όταν ο χώρος αναζήτησης είναι πολύ μεγάλος ή ακόμα και άγνωστος χωρίς να μειωθεί η ακρίβεια.
- Δίνεται η ευκαιρία στο σχεδιαστή να δημιουργήσει νέους αλγορίθμους διασταύρωσης και μετάλλαξης οι οποίοι θα κάνουν χρήση της γνώσης του προβλήματος.
- Μειώνεται η πιθανότητα να εμφανιστεί το φαινόμενο της “απάτης” (deception) (Goldberg D. E., Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction, 1989) (Goldberg D. E., Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis, 1989) και μειώνεται (ή ακόμα και μηδενίζεται) η πιθανότητα ο τελεστής της διασταύρωσης να διαταράξει την τιμή μιας μεταβλητής.

3.8 Κωδικοποίηση μετάθεσης - Permutation encoding - Path representation

Η συγκεκριμένη κωδικοποίηση χρησιμοποιείται σε προβλήματα διάταξης, δηλαδή σε προβλήματα στα οποία μας ενδιαφέρει η σειρά των μεταβλητών, όπως το πρόβλημα του πλανόδιου πωλητή ή προβλήματα χρονοδρομολόγησης διεργασιών. Το γονίδια του χρωμοσώματος μπορούν να είναι οτιδήποτε, από έναν ακέραιος αριθμός μέχρι μια περίπλοκη δομή. Προφανώς μας συμφέρει να αντιστοιχίσουμε τις μεταβλητές του προβλήματος σε έναν ακέραιο αριθμό. Αυτό που έχει σημασία στην κωδικοποίηση μετάθεσης δεν είναι το περιεχόμενο των γονιδίων, αλλά η σειρά που εμφανίζονται στο χρωμόσωμα.

Τα γονίδια θα πρέπει να εμφανίζονται στο χρωμόσωμα ακριβώς μια φορά, οπότε είναι προφανές ότι χρειάζονται ειδικοί τελεστές διασταύρωσης και μετάλλαξης οι οποίοι θα ικανοποιούν αυτόν τον περιορισμό. Οι τελεστές διασταύρωσης και μετάλλαξης δε θα πρέπει να αλλάζουν τα γονίδια. Δε θέλουμε να βρούμε νέες μεταβλητές αλλά να ταξινομήσουμε τις ήδη υπάρχουσες.



Εικόνα 25 Κωδικοποίηση μετάθεσης

Για παράδειγμα αν έχουμε n πόλεις ένα διάνυσμα της μορφής $n = (i_1, i_2, \dots, i_n)$ αντιστοιχεί στην διαδρομή η οποία ξεκινάει από την πόλη i_1, πηγαίνει στην πόλη i_{21}, … , στην πόλη i_n και επιστρέφει στην πόλη i_1. Αν το πλήθος των πόλεων είναι n χρειαζόμαστε n ακέραιους για την αναπαράσταση. Οι ειδικοί τελεστές διασταύρωσης και μετάλλαξης που έχουν δημιουργηθεί για την κωδικοποίηση μετάθεσης, φροντίζουν το παραγόμενο διάνυσμα να περιέχει τον κάθε ακέραιο ακριβώς μία φορά, χωρίς να παράγουν κάποιο ακέραιο που δεν ανήκει στο αρχικό διάνυσμα ακεραίων.

Στο πρόβλημα του πλανόδιου πωλητή κάθε διαδρομή μπορεί να αναπαρασταθεί με 2*n διαφορετικούς τρόπους αν χρησιμοποιήσουμε την κωδικοποίηση μετάθεσης και αν το TSP είναι συμμετρικό, όπου n ο αριθμός των πόλεων. n γιατί κάθε διαδρομή είναι ένας κύκλος hamilton οπότε δεν παίζει ρόλο από που ξεκινάμε, οι διαδρομές (5-4-1-2-3) και (1-2-3-5-4) έχουν το ίδιο μήκος και 2 γιατί αφού το TSP είναι συμμετρικό η απόσταση από την πόλη 1 στην πόλη 2 είναι ίδια με την απόσταση από την πόλη 2 στην πόλη 1. Οπότε η διαδρομή (5-4-1-2-3) έχει το ίδιο μήκος με τη διαδρομή (5-3-2-1-4).

Σημείωση: Ο συνολικός αριθμός των σημείων στο πρόβλημα του πλανόδιου πωλητή είναι n+1, όπου n ο αριθμός των πόλεων. Στον αριθμό αυτό προσθέτουμε τον πωλητή. Επειδή όμως στην αναζήτηση της βέλτιστης διαδρομής δεν παίζει ρόλο από πιο σημείο ξεκινάμε, όταν λέμε αναφερόμαστε σε ένα πρόβλημα TSP με n σημεία/πόλεις, σε αυτά θα συμπεριλαμβάνεται και ο πωλητής.

Συγκεκριμένα για το TSP έχουν προταθεί και άλλες τρεις μορφές αναπαράστασης. Οι δύο πρώτες (ordinal και adjacency representation) δεν δίνουν καλά αποτελέσματα (P. Larrañaga, 1999), ενώ η τρίτη (matrix representation) έχει πολλές παραλλαγές (οι δύο πιο γνωστές είναι από τους Fox and McMahon (McMahon, Genetic operators for sequencing problems, 1991) και Homaifar et al. (A. Homaifar, 1993)) και παρόλο που δίνει καλά αποτελέσματα, λόγω της πολυπλοκότητας της και της ανάγκης για δημιουργία διαφορετικών τελεστών διασταύρωσης και μετάλλαξης, δε θα αναλυθεί αλλά θα γίνει μια απλή αναφορά στις δύο γνωστές μεθόδους που αναφέραμε.

3.8.1 Ordinal representation

Η αναπαράσταση ordinal representation (J. Grefenstette, 1985) δημιουργήθηκε από κάποιους ερευνητές στην προσπάθειά τους να λύσουν το TSP με τη χρήση Γ.Α., χωρίς να χρειάζεται κάποιος ειδικός τελεστής διασταύρωσης. Ο κλασικός τελεστή διασταύρωσης one-point crossover δίνει πάντα έγκυρες διαδρομές.

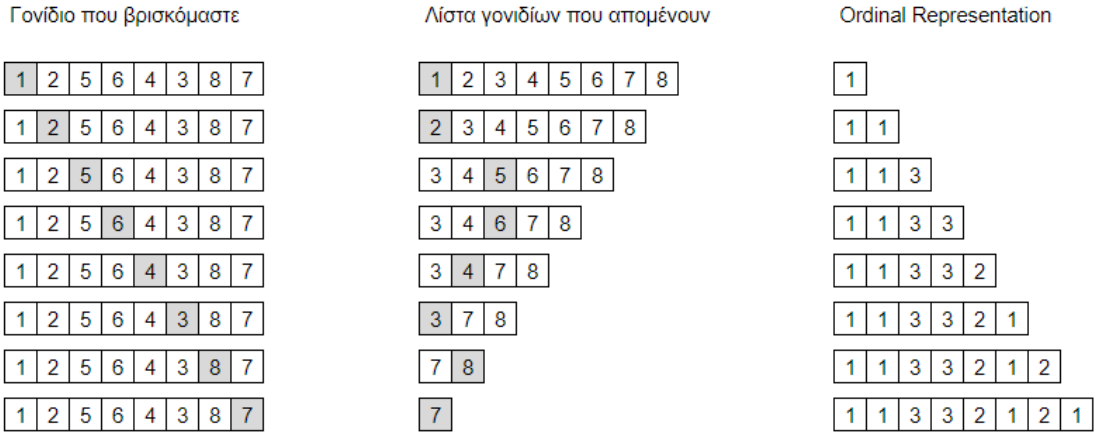
Η αναπαράσταση ordinal representation θα αναλυθεί μέσα από ένα παράδειγμα. Έστω ότι έχουμε το χρωμόσωμα με αναπαράσταση μετάθεσης την C=(1, 2, 5, 6, 4, 3, 8, 7). Για να πάρουμε την αντίστοιχη ordinal αναπαράσταση ordinal O ακολουθούμε τα εξής βήματα:

- Παίρνουμε την ταξινομημένη λίστα L που περιέχει όλα τα γονίδια σε αύξουσα σειρά δηλαδή L=(1, 2, 3, 4, 5, 6, 7, 8).
- Το πρώτο γονίδιο στο χρωμόσωμα C δηλαδή το γονίδιο 1 βρίσκεται στην πρώτη θέση στη λίστα L. Προσθέτουμε τη θέση που βρίσκεται στην λίστα L στην ordinal representation και αφαιρούμε το γονίδιο από την λίστα L. Άρα έχουμε L = (2, 3, 4, 5, 6, 7, 8) και O = (1).
- Το επόμενο γονίδιο στο χρωμόσωμα C είναι το 2. Ακολουθώντας τα ίδια βήματα έχουμε L = (3, 4, 5, 6, 7, 8) και O = (1, 1).
- Το επόμενο γονίδιο στο χρωμόσωμα C είναι το 5. Το 5 βρίσκεται στην τρίτη θέση στη λίστα L. Επομένως παίρνουμε L = (3, 4, 6, 7, 8) και O = (1, 1, 3).
- Το επόμενο γονίδιο στην C είναι το 6 που μας δίνει L = (3, 4, 7, 8) και O = (1, 1, 3, 3).
- Το επόμενο γονίδιο στην C είναι το 4 που μας δίνει L = (3, 7, 8) και O = (1, 1, 3, 3, 2).
- Το επόμενο γονίδιο στην C είναι το 3 που μας δίνει L = (7, 8) και O = (1, 1, 3, 3, 2, 1).
- Το επόμενο γονίδιο στην C είναι το 8 που μας δίνει L = (7) και O = (1, 1, 3, 3, 2, 1, 2).
- Το επόμενο γονίδιο στην C είναι το 7 που μας δίνει L = () και O = (1, 1, 3, 3, 2, 1, 2, 1).

Σε κάθε διαδρομή αντιστοιχεί μια μοναδική ordinal αναπαράσταση και είναι πολύ εύκολο να αποκωδικοποιήσουμε την ordinal αναπαράσταση ακολουθώντας τα βήματα που χρησιμοποιούμε για την κωδικοποίηση αντίστροφα. Για παράδειγμα έστω ότι έχουμε O = (1, 1, 3, 3, 2, 1, 2, 1) και L = (1, 2, 3, 4, 5, 6, 7, 8).

- Το πρώτο γονίδιο στην O είναι το 1. Διαγράφουμε το 1 από την L και έχουμε L = (2, 3, 4, 5, 6, 7, 8) και C = (1).
- Το επόμενο γονίδιο στην O είναι πάλι το 1 και μας δίνει L = (3, 4, 5, 6, 7, 8) και C = (1, 2).
- Το επόμενο γονίδιο στην O είναι το 3 και μας δίνει L = (3, 4, 6, 7, 8) και C = (1, 2, 5).
- Το επόμενο γονίδιο στην O είναι το 3 και μας δίνει L = (3, 4, 7, 8) και C = (1, 2, 5, 6).

- Το επόμενο γονίδιο στην Ο είναι το 2 και μας δίνει L = (3, 7, 8) και C = (1, 2, 5, 6, 4).
- Το επόμενο γονίδιο στην Ο είναι το 1 και μας δίνει L = (7, 8) και C = (1, 2, 5, 6, 4, 3).
- Το επόμενο γονίδιο στην Ο είναι το 2 και μας δίνει L = (7) και C = (1, 2, 5, 6, 4, 3, 8).
- Το επόμενο γονίδιο στην Ο είναι το 1 και μας δίνει L = () και C = (1, 2, 5, 6, 4, 3, 8, 7).



Εικόνα 26 Μετατροπή path σε ordinal representation

Το ενδιαφέρον αυτής της μεθόδου είναι ότι μπορεί να εφαρμοστεί ο κλασικός one point crossover. Για παράδειγμα έστω ότι έχουμε τους γονείς P1 = (1 2 | 5 2 1 2 1) και P2 = (4 1 | 4 2 1 2 1) σε ordinal αναπαράσταση. Η αναπαράσταση αυτή αντιστοιχεί στις έγκυρες διαδρομές P1 = (1-3-7-4-2-6-5) και P2 = (4-1-6-3-2-7-5). Μετά την εφαρμογή του τελεστή one point crossover, που θα δούμε αναλυτικά στην παράγραφο “Διασταύρωση”, προκύπτουν οι απόγονοι O1 = (1 2 4 2 1 2 1) και O2 = (4 1 5 2 1 2 1) που αντιστοιχούν στις έγκυρες διαδρομές O1 = (1-3-6-4-2-7-5) και O2 = (4-1-7-3-2-6-5).

3.8.2 Adjacency representation

Η δεύτερη αναπαράσταση που δημιουργήθηκε για το TSP είναι η adjacency representation (J. Grefenstette, 1985). Σε αυτή την αναπαράσταση κάθε πόλη j βρίσκεται στη θέση i του χρωμοσώματος αν στην διαδρομή πηγαίνουμε από την πόλη i στην πόλη j.

Για παράδειγμα έστω ότι έχουμε την διαδρομή 1-3-5-6-4-2-8-7. Από την πόλη 1 πηγαίνουμε στην πόλη 3 οπότε το πρώτο γονίδιο του χρωμοσώματος είναι το 3. Από την 2 πηγαίνουμε στην πόλη 8 οπότε το δεύτερο γονίδιο είναι το 8. Από την πόλη 3 πηγαίνουμε στην πόλη 5 οπότε το τρίτο γονίδιο του χρωμοσώματος είναι το 5. Επαναλαμβάνοντας την διαδικασία παίρνουμε το χρωμόσωμα (3, 8, 5, 2, 6, 4, 1, 7). Η διαδικασία φαίνεται αναλυτικά στο παρακάτω σχήμα.

Εικόνα 27 Path to adjacency representation

Στην κλασική αναπαράσταση μετάθεσης για κάθε διαδρομή έχουμε 2*n τρόπους για την αναπαράσταση μιας διαδρομής αν το TSP είναι συμμετρικό και n αν είναι ασύμμετρο. Στην adjacency αναπαράσταση κάθε διαδρομή μπορεί να αναπαρασταθεί με δύο τρόπους αν το TSP είναι συμμετρικό και έναν αν είναι ασύμμετρο.

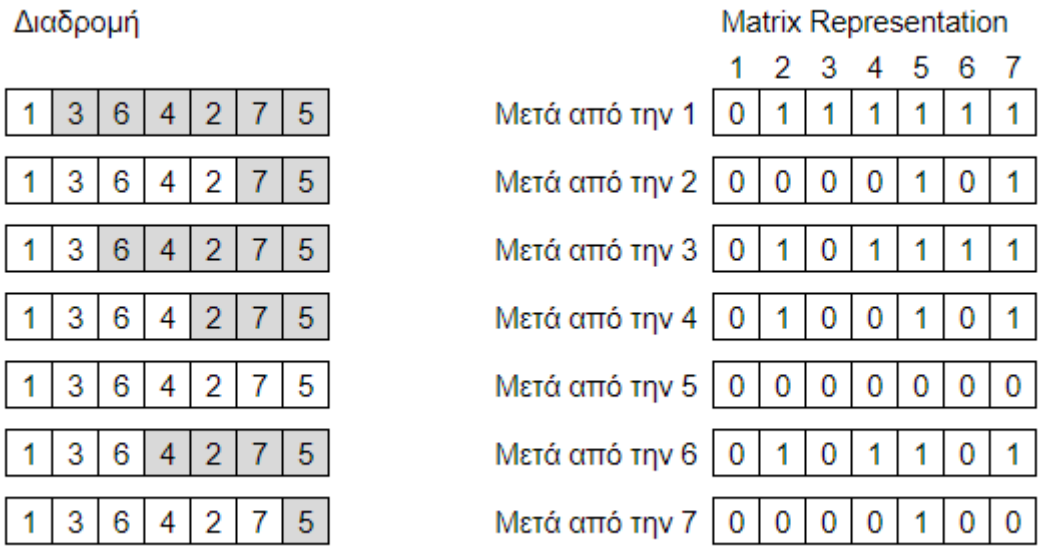
Για την αναπαράσταση adjacency δημιουργήθηκαν ειδικοί αλγόριθμοι διασταύρωσης (alternating edges, subtour chunks κτλ.) (Dr. Anantkumar J. Umbarkar, 2015) (P. Larrañaga, 1999), αλλά τελικά η συγκεκριμένη αναπαράσταση δεν είχε τα επιθυμητά αποτελέσματα.

3.8.3 Matrix representation

Η συγκεκριμένη μορφή αναπαράστασης έχει αρκετές παραλλαγές. Στη συνέχεια αυτής της παραγράφου θα αναλύσουμε τις δύο πιο διαδεδομένες. Η πρώτη παραλλαγή είναι από τους Fox and McMahon (McMahon, Genetic operators for sequencing problems, 1991). Το χρωμόσωμα κάθε διαδρομής αντιστοιχεί σε έναν δυαδικό πίνακα κατά τέτοιο τρόπο ώστε το στοιχείο Mij του πίνακα είναι 1 αν και μόνο αν η πόλη i βρίσκεται πριν από την πόλη j στη διαδρομή.

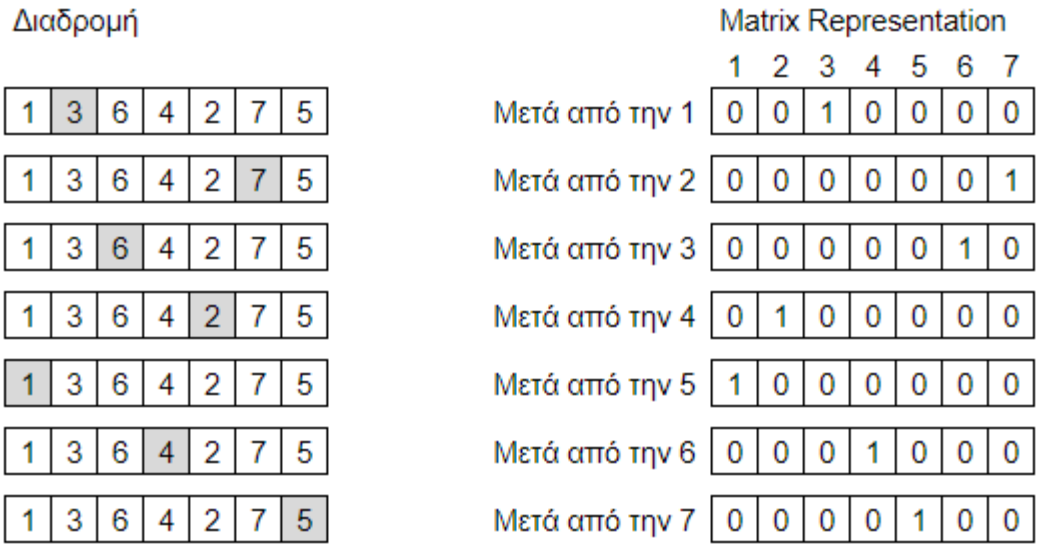
Για παράδειγμα έστω ότι έχουμε την διαδρομή (1-3-6-4-2-7-5).

- Η πρώτη γραμμή του πίνακα θα είναι η 0,1,1,1,1,1,1 γιατί όλες οι πόλεις είναι μετά την πόλη 1.
- Η δεύτερη γραμμή του πίνακα θα είναι η 0,0,0,0,1,0,1 γιατί οι πόλεις 7 και 5 είναι μετά την πόλη 2.
- Η τρίτη γραμμή θα είναι η 0,1,0,1,1,1,1 γιατί οι πόλεις 6,4,2,7,5 είναι μετά την πόλη 3.
- Η τέταρτη γραμμή θα είναι η 0,1,0,0,1,0,1 γιατί οι πόλεις 2,7,5 είναι μετά την πόλη 4.
- Η πέμπτη γραμμή θα είναι η 0,0,0,0,0,0,0 γιατί δεν υπάρχει καμία πόλη μετά την πόλη 5.
- Η έκτη γραμμή θα είναι η 0,1,0,1,1,0,1 γιατί οι πόλεις 4,2,7,5 είναι μετά την πόλη 6.
- Τέλος η έβδομη γραμμή θα είναι η 0,0,0,0,1,0,0 γιατί μόνο η πόλη 5 είναι μετά την πόλη 7.



Εικόνα 28 Matrix representation

Η δεύτερη εκδοχή αυτής της αναπαράστασης είναι από τους Homaifar et al. (A. Homaifar, 1993). Το χρωμόσωμα κάθε διαδρομής αντιστοιχεί σε έναν δυαδικό πίνακα κατά τέτοιο τρόπο ώστε το στοιχείο Mij του πίνακα είναι 1 αν και μόνο αν η επόμενη από την πόλη i είναι η πόλη j. Επομένως κάθε γραμμή του πίνακα θα έχει μόνο έναν άσσο. Για παράδειγμα για την διαδρομή (1-3-6-4-2-7-5) παίρνουμε τον ακόλουθο πίνακα:



Εικόνα 29 Matrix representation - alternative method

4 Αρχικοποίηση (Initialization)

Σε αυτό το στάδιο πρέπει να δημιουργήσουμε τον αρχικό πληθυσμό από δυνατές λύσεις του προς επίλυση προβλήματος. Το πως θα γίνει η αρχικοποίηση και το μέγεθος του αρχικού πληθυσμού αποτελούν αντικείμενο μελέτης. Για τη διαδικασία αρχικοποίησης έχουμε τις εξής επιλογές.

- **Τυχαία παραγωγή του αρχικού πληθυσμού**. Με διαφορά η πιο διαδεδομένη διαδικασία αρχικοποίησης καθώς δίνει την δυνατότητα στον Γ.Α. να ερευνήσει ένα μεγάλο αριθμό πιθανών λύσεων.
- **Χρήση ευρετικού αλγορίθμου**. Ο ευρετικός αλγόριθμος που θα χρησιμοποιηθεί εξαρτάται από το πρόβλημα. Για το TSP μπορεί να χρησιμοποιηθεί για παράδειγμα ο αλγόριθμος του πλησιέστερου γείτονα (nearest neighbor). Με αυτό τον τρόπο αυξάνουμε την ταχύτητα σύγκλισης του Γ.Α. αλλά επειδή μειώνουμε τον χώρο των πιθανών λύσεων, υπάρχει κίνδυνος ο αλγόριθμος να εγκλωβιστεί σε ένα τοπικό ακρότατο.
- **Υβριδική αρχικοποίηση**. Για να αυξήσουμε την ταχύτητα σύγκλισης, αλλά και να έχει ο γενετικός αλγόριθμος ένα αρκετά μεγάλο χώρο αναζήτησης λύσεων, ένα μέρος του αρχικού πληθυσμού παράγεται τυχαία και ένα μέρος με τη χρήση κάποιου ευρετικού αλγορίθμου.

Για το μέγεθος του αρχικού πληθυσμού από δυνατές λύσεις (pop_size) δεν υπάρχουν συγκεκριμένοι κανόνες. Συνήθως χρησιμοποιείται ένας αριθμός ανάμεσα στο 20 και στο 100 αλλά κάποια προβλήματα μπορεί να χρειάζονται ακόμα μεγαλύτερο αριθμό. Όσο μεγαλώνει το μέγεθος τόσο αυξάνεται και το κόστος. Μέχρι κάποιο σημείο (100 για τα περισσότερα προβλήματα) με την αύξηση του αρχικού πληθυσμού παίρνουμε καλύτερες λύσεις γρηγορότερα (σε λιγότερες γενιές). Από κάποιο σημείο και μετά η αύξηση του αρχικού πληθυσμού αυξάνει το κόστος χωρίς να παίρνουμε καλύτερες λύσεις (Olympia Roeva, 2013) (Ali Yadav Nikravesh, 2018).

Μερικοί από τους παράγοντες που επηρεάζουν το μέγεθος του πληθυσμού είναι οι εξής:

- **Διάσταση του προβλήματος**. Για μεγαλύτερα προβλήματα χρειαζόμαστε μεγαλύτερο μέγεθος πληθυσμού.
- **Μέθοδος επιλογής**. Αν η μέθοδος επιλογής έχει μεγάλη πίεση επιλογής, το μέγεθος του πληθυσμού θα πρέπει να είναι μεγαλύτερο, διαφορετικά ο πληθυσμός θα γεμίσει με αντίγραφα του ίδιου ατόμου και θα συγκλίνει πρόωρα.
- **Ακρίβεια / Κόστος**. Ανάλογα με το αποδεκτό για εμάς περιθώριο λάθους μπορεί να επιλέξουμε ένα μικρότερο μέγεθος πληθυσμού (παρόλο που με μεγαλύτερο ίσως υπάρχουν περιθώρια βελτίωσης) ώστε να μειώσουμε το κόστος.
- **Υλοποίηση του γενετικού αλγορίθμου**. Οι διάφορες μορφές κωδικοποίησης καθώς επίσης και οι αλγόριθμοι διασταύρωσης και μετάλλαξης, επηρεάζονται διαφορετικά από το μέγεθος του πληθυσμού.

Έχουν γίνει πολλές μελέτες για τον υπολογισμό του κατάλληλου μεγέθους πληθυσμού (n) (Pedro A. Diaz-Gomez, 2007) (S. Gotshall, 2002) (Jason Digalakis, 2001). Όμως όπως είπαμε καμία από αυτές δεν δίνει αποτέλεσμα για όλα τα προβλήματα. Ο μοναδικός τρόπος για να υπολογίσουμε το κατάλληλο n για το πρόβλημά μας είναι με δοκιμές. Να αρχίσουμε δηλαδή από ένα μικρό n το οποίο σταδιακά να αυξάνεται μέχρι είτε η αύξηση να μη μας δίνει καλύτερα αποτελέσματα είτε να μη θέλουμε να χρησιμοποιήσουμε περισσότερη υπολογιστική ισχύς.

Ένας καλός αρχικός πληθυσμός πρέπει να αποτρέπει την πρόωρη σύγκλιση. Αν το μέγεθος του αρχικού πληθυσμού είναι πολύ μικρό ο πληθυσμός δε θα έχει μεγάλη ποικιλομορφία. Έτσι σύντομα όλος ο πληθυσμός θα γεμίσει με αντίγραφα των ίδιων ατόμων (αυτό συμβαίνει λόγω των building blocks που θα δούμε στη συνέχεια) και έτσι ο αλγόριθμος θα εγκλωβιστεί σε ένα τοπικό ακρότατο.

Πρόωρη σύγκλιση μπορεί επίσης να υπάρξει, ανεξάρτητα από το μέγεθος του αρχικού πληθυσμού, αν χρησιμοποιηθεί κάποιος ευρετικός αλγόριθμος. Όμως λόγω των building blocks, υπάρχει ο κίνδυνος ο πληθυσμός να γεμίσει με αντίγραφα του ίδιου ατόμου.

Κάποιες μελέτες υποστηρίζουν (Schaffer, 1989) ότι το μέγεθος του αρχικού πληθυσμού επηρεάζεται από την πιθανότητα μετάλλαξης ή/και την πιθανότητα διασταύρωσης (όταν το μέγεθος του πληθυσμού είναι μικρό επηρεάζεται περισσότερο από την μετάλλαξη και λιγότερο από τη

διασταύρωση), ενώ κάποιες άλλες υποστηρίζουν ότι η επιρροή της πιθανότητας μετάλλαξης ή/και της πιθανότητας διασταύρωσης στο μέγεθος του αρχικού πληθυσμού είναι ασήμαντη (Kevin L. Mills, 2014).

Για το TSP έχουν προταθεί (Ahmad B. Hassanat, 2018) πάρα πολλοί τρόποι αρχικοποίησης του πληθυσμού. Οι περισσότεροι από αυτούς κάνουν χρήση κάποιου γνωστού ευρετικού αλγορίθμου για το TSP. Για παράδειγμα η επιλεκτική αρχικοποίηση (Yang, 1998) χρησιμοποιεί τον αλγόριθμο K πλησιέστερων γειτόνων. Επιλέγει τυχαία μια αρχική πόλη και τοποθετεί τυχαία τους K πλησιέστερους γείτονές της στη διαδρομή. Στη συνέχεια συμπληρώνει το χρωμόσωμα τοποθετώντας τυχαία τις N-K-1 πόλεις.

Μια ενδιαφέρουσα προσέγγιση είναι η αρχικοποίηση με ταξινόμηση του πληθυσμού. Αρχικά παράγουμε τυχαία ένα πολύ μεγάλο αριθμό από λύσεις, τις ταξινομούμε με βάση την απόδοση και χρησιμοποιούμε τις καλύτερες από αυτές για να δημιουργήσουμε τον αρχικό πληθυσμό (Olga Yugay, 2008). Στη συνέχεια θα δούμε τις δύο πιο διαδεδομένες μεθόδους αρχικοποίησης για το TSP.

4.1 Τυχαία επιλογή – Random

Ο πιο απλός τρόπος για να παράγουμε τον αρχικό πληθυσμό είναι η τυχαία επιλογή. Για παράδειγμα για το πρόβλημα του πλανόδιου πωλητή για 9 σημεία αν χρησιμοποιηθεί η αναπαράσταση μετάθεσης, αν θέλουμε το μέγεθος του πληθυσμού να είναι N τότε διαλέγουμε τυχαία N διαδρομές που ενώνουν αυτά τα 9 σημεία.

Η μέθοδος αυτή χρησιμοποιείται από τους περισσότερους γενετικούς αλγορίθμους, καθώς αυξάνει το χώρο αναζήτησης λύσεων, μειώνοντας έτσι τον κίνδυνο να εγκλωβιστεί ο Γ.Α. σε κάποιο τοπικό ακρότατο. Φυσικά αυξάνοντας τον χώρο αναζήτησης, ο γενετικός αλγόριθμος χρειάζεται περισσότερο χρόνο για να συγκλίνει.

1	5	9	8	2	3	7	6	4
5	2	3	8	9	1	6	7	4
2	8	9	7	6	5	4	1	3
7	6	5	8	9	4	3	2	1
1	5	3	8	4	9	7	6	2

Εικόνα 30 Τυχαίος αρχικός πληθυσμός για pop_size = 5

4.2 Ευρετικός αλγόριθμος πλησιέστερου γείτονα NN (Nearest neighbor)

Για να συγκλίνει πιο γρήγορα ο γενετικός αλγόριθμος μπορούμε να χρησιμοποιήσουμε έναν ευρετικό αλγόριθμο. Ο πιο απλός για το TSP είναι ο αλγόριθμος nearest neighbor τον οποίο αναλύσαμε στο πρώτο μέρος της εργασίας. Ο αλγόριθμος nearest neighbor έχει τα εξής πλεονεκτήματα σε σχέση με τους άλλους ευρετικούς αλγορίθμους:

- Είναι πολύ γρήγορος.
- Οι λύσεις που δίνει είναι κατά μέσο όρο 24% χειρότερες από τη βέλτιστη, αφήνοντας έτσι περιθώρια στον Γ.Α. να αναζητήσει περισσότερες λύσεις (Eneko Osaba. Roberto Carballedo, 2014).

Για να πάρουμε τα N άτομα που θα αποτελέσουν τον αρχικό μας πληθυσμό μπορούμε να ξεκινήσουμε από διαφορετική πόλη κάθε φορά. Αν το μέγεθος του πληθυσμού που θέλουμε είναι

μεγαλύτερο από τον αριθμό των κορυφών, τότε συμπληρώνουμε τον πληθυσμό με κάποιες τυχαίες λύσεις για να αυξήσουμε το χώρο αναζήτησης.

1	2	3	5	4	5	7	8	9
2	3	4	5	6	8	7	9	1
3	5	4	6	7	8	9	1	2
4	5	6	7	8	9	3	2	1
5	1	7	8	9	6	2	3	4

Εικόνα 31 Αρχικός πληθυσμός με τη χρήση ευρετικού αλγορίθμου

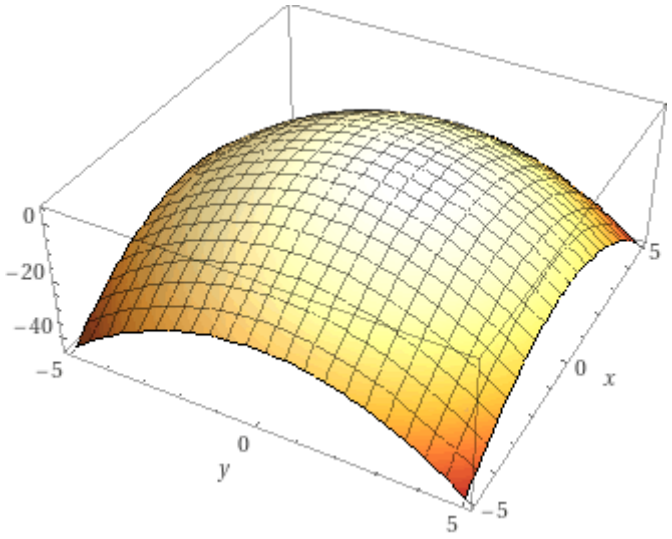
5 Αξιολόγηση (Evaluation)

Στη φύση οι πιο “κατάλληλοι” οργανισμοί, δηλαδή οι πιο δυνατοί, έξυπνοι, γρήγοροι κτλ. έχουν μεγαλύτερη πιθανότητα να επιβιώσουν και να κάνουν απογόνους. Αντίθετα οι πιο “αδύναμοι” έχουν μεγαλύτερη πιθανότητα να πεθάνουν από ασιτία, να καταλήξουν τροφή για άλλους οργανισμούς, να πάθουν κάποιο ατύχημα κτλ. Επομένως σύμφωνα με τη φυσική επιλογή, οι πιο “κατάλληλοι” οργανισμοί έχουν μεγαλύτερη πιθανότητα να κληρονομήσουν τα χαρακτηριστικά τους στη νέα γενιά. Με την πάροδο του χρόνου, η νέα γενιά θα γίνει πιο δυνατή, πιο γρήγορη, πιο έξυπνη.

Οι γενετικοί αλγόριθμοι με την αξιολόγηση προσπαθούν να μιμηθούν αυτή τη διαδικασία που συμβαίνει στη φύση. Σε κάθε γενιά τα άτομα (λύσεις) αξιολογούνται και αυτά με την καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιλεγούν για να κάνουν απογόνους. Η αξιολόγηση γίνεται με μία συνάρτηση η οποία ονομάζεται συνάρτηση αξιολόγησης, ή συνάρτηση καταλληλότητας, ή συνάρτηση ικανότητας, ή στα αγγλικά fitness function. Πολλές φορές θα την συναντήσουμε και ως αντικειμενική συνάρτηση. Σε προβλήματα βελτιστοποίησης η αντικειμενική συνάρτηση και η συνάρτηση ικανότητας ταυτίζονται αλλά αυτό δεν ισχύει για όλα τα προβλήματα.

Κατά την αξιολόγηση, η κωδικοποιημένη λύση (γονότυπος), αποκωδικοποιείται (φαινότυπος) και εισάγεται ως είσοδος στη συνάρτηση αξιολόγησης, η οποία επιστρέφει συνήθως ως τιμή ένα θετικό αριθμό. Όσο μεγαλύτερη είναι η τιμή τόσο πιο καλή είναι η λύση, άρα έχει μεγαλύτερη πιθανότητα να επιλεγεί για να παράγει απογόνους. Στόχος μας είναι τα καλά χαρακτηριστικά (γονίδια) που υπάρχουν στις λύσεις με υψηλή απόδοση, να διατηρηθούν και να περάσουν στην επόμενη γενιά. Όπως και στη φύση, έτσι και στους γενετικούς αλγορίθμους, τα πιο κατάλληλα άτομα δεν είναι σίγουρο ότι θα επιβιώσουν, αλλά έχουν μεγαλύτερη πιθανότητα από τα άτομα με μικρή απόδοση. Έτσι δίνεται η δυνατότητα στον γενετικό αλγόριθμο να εξερευνήσει σε βάθος το χώρο των λύσεων, χωρίς να εγκλωβίζεται σε τοπικά ακρότατα.

Σημείωση: Αν η συνάρτηση καταλληλότητας επιστρέφει αρνητικές τιμές, δεν μπορούμε να χρησιμοποιήσουμε κάποιες από τις μεθόδους επιλογής, όπως η επιλογή ρουλέτας. Σε αυτή την περίπτωση θα πρέπει ή να χρησιμοποιήσουμε κάποια άλλη μέθοδο επιλογής ή να γίνει μετασχηματισμός στη συνάρτηση καταλληλότητας ώστε να επιστρέφει μόνο θετικές τιμές. Για παράδειγμα η συνάρτηση καταλληλότητας $f(x,y) = 5 - (x^2 + y^2)$ $με$ $x, y \in [-5, 5]$ επιστρέφει και θετικές και αρνητικές τιμές όπως φαίνεται στο σχήμα που ακολουθεί:



Εικόνα 32 Η συνάρτηση καταλληλότητας επιστρέφει και αρνητικές τιμές

Με την επιλογή της κατάλληλης κωδικοποίησης οι λύσεις με παρόμοια απόδοση έχουν κάποια κοινά χαρακτηριστικά. Με την αξιολόγηση στοχεύουμε στη διατήρηση των κοινών χαρακτηριστικών που εμφανίζονται στα άτομα με υψηλή απόδοση από γενιά σε γενιά και με τη χρήση του τελεστή διασταύρωσης, τον εμπλουτισμό τους με καλά χαρακτηριστικά από άλλα άτομα, δημιουργώντας έτσι καλύτερες λύσεις.

Για παράδειγμα έστω ότι θέλουμε να βρούμε το μέγιστο της συνάρτησης f(x)=x^2 με το x να είναι ένας ακέραιος που παίρνει τιμές στο διάστημα [0,31]. Η συνάρτηση καταλληλότητας του προβλήματος είναι η ίδια η συνάρτηση. Έστω ότι έχουμε χρησιμοποιήσει δυαδική αναπαράσταση και ο πληθυσμός αποτελείται από τα εξής χρωμοσώματα.

Χρωμόσωμα	Απόδοση
(1 1 0 0 0)	576
(1 0 0 0 1)	289
(0 0 0 0 1)	1
(0 0 1 1 0)	36
(1 0 0 1 1)	361

Σύμφωνα με τα όσα είπαμε ως τώρα τα χρωμόσωμα (1 1 0 0 0) είναι το πιο κατάλληλο και το χρωμόσωμα (0 0 0 0 1) το λιγότερο κατάλληλο. Επομένως το χρωμόσωμα (1 1 0 0 0) έχει μεγαλύτερη πιθανότητα να επιβιώσει και να κάνει απογόνους, μεταφέροντας τα καλά του γονίδια (τους δύο άσσους στην αρχή του χρωμοσώματος) στον επόμενο πληθυσμό. Παρόλα αυτά την αν χρησιμοποιήσουμε διασταύρωση ενός σημείου μετά το δεύτερο γονίδιο, η καλύτερη λύση προκύπτει από τη διασταύρωση των χρωμοσωμάτων (1 1 | 0 0 0) και (0 0 | 1 1 0) τα οποία θα μας δώσουν τον απόγονο (1 1 | 1 1 0) με απόδοση 900. Μέσα από αυτό το απλό παράδειγμα καταλαβαίνουμε γιατί πρέπει να επιβιώνουν και κάποια από τα άτομα με χαμηλή απόδοση.

Η επιλογή της κατάλληλης συνάρτησης αξιολόγησης μπορεί να είναι προφανής για κάποια προβλήματα όπως τα προβλήματα βελτιστοποίησης ή το πρόβλημα του πλανόδιου πωλητή (Για το TSP η συνάρτηση καταλληλότητας είναι η f=1/L όπου L το μήκος της διαδρομής), ενώ για κάποια άλλα όπως τα προβλήματα πολλαπλών στόχων (multi-objective problems), δηλαδή προβλήματα με πολλές αντικειμενικές συναρτήσεις, αποτελεί ένα ξεχωριστό πρόβλημα. Οι γενετικοί αλγόριθμοι που επιλύουν αυτά τα προβλήματα ονομάζονται γενετικοί αλγόριθμοι πολλαπλών στόχων (multi-objective GAs - MOGAs).

Ο υπολογισμός της καταλληλότητας ενός ατόμου θα πρέπει να επιβαρύνει το γενετικό αλγόριθμο όσο γίνεται λιγότερο, γιατί γίνεται σε κάθε κύκλο του αλγορίθμου τόσες φορές όσες είναι και το μέγεθος του πληθυσμού. Όταν ο υπολογισμός της καταλληλότητας είναι πολύ χρονοβόρος, ή όταν η

συνάρτηση καταλληλότητας έχει “θόρυβο” ή όταν δεν υπάρχει συνάρτηση καταλληλότητας για το πρόβλημα, τότε καταφεύγουμε σε προσεγγιστικές μεθόδους υπολογισμού της καταλληλότητας.

Στον αλγόριθμο που υλοποιήσαμε μπορούμε να εισάγουμε τα δεδομένα (τα σημεία που βρίσκονται οι πόλεις) με δύο τρόπους. Ο πρώτος τρόπος είναι να θεωρήσουμε ότι τα σημεία βρίσκονται σε ένα καρτεσιανό σύστημα συντεταγμένων, οπότε κάθε σημείο ορίζεται ως Α(Χ,Υ) με το Χ να είναι η τετμημένη και το Υ η τεταγμένη του σημείου. Ο δεύτερος τρόπος είναι να θεωρήσουμε ότι τα σημεία βρίσκονται στη Γη, οπότε κάθε σημείο ορίζεται ως Α(Χ,Υ) με το Χ να είναι το γεωγραφικό πλάτος και το Υ το γεωγραφικό μήκος του σημείου (αφού μετατρέψουμε τις μοίρες σε δεκαδικούς αριθμούς - Decimal degrees).

Προφανώς για την καλύτερη απόδοση του συστήματος στην αρχή γίνεται ο υπολογισμός της απόστασης όλων το σημείων με όλα τα σημεία και το αποτέλεσμα αποθηκεύεται σε ένα δισδιάστατο array. Στην περίπτωσή μας η απόσταση d_{AB} είναι ίση με την απόσταση d_{BA} αλλά αν χρησιμοποιούσαμε πραγματικά δεδομένα μέσω του geocoder αυτό δεν ισχύει. Μπορεί στη διαδρομή να υπάρχουν μονόδρομοι, εμπόδια κτλ.

5.1 Καρτεσιανό σύστημα συντεταγμένων

Στο σύστημα αυτό οι συντεταγμένες (X_A, Y_A) ενός σημείου Α δείχνουν τη θέση του Α κατά την ορθή προβολή του στους άξονες των τετμημένων και τεταγμένων αντίστοιχα. Ως απόσταση d μεταξύ δύο σημείων Α και Β ορίζουμε τον αριθμό $d = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$ και το μήκος της συνολικής διαδρομής θα είναι το άθροισμα των αποστάσεων μεταξύ των σημείων που αποτελούν την διαδρομή.

5.2 Γεωγραφικές συντεταγμένες

Κάθε σημείο ορίζεται από 2 αριθμούς (Χ,Υ) οι οποίοι αποτελούν το γεωγραφικό μήκος και πλάτος του αντίστοιχα.

Γεωγραφικό πλάτος Χ (latitude) ενός σημείου που βρίσκεται στην επιφάνεια της Γης είναι η γωνία που σχηματίζει η κατακόρυφος του τόπου με το επίπεδο του ισημερινού και παίρνει τις τιμές 0° - 90° Β ή 0° - 90° Ν.

Γεωγραφικό μήκος Υ (longitude) ενός σημείου στην επιφάνεια της Γης είναι η στερεή γωνία που σχηματίζεται από το επίπεδο του μεσημβρινού που διέρχεται από το εν λόγω σημείο με το επίπεδο του πρώτου μεσημβρινού και παίρνει τις τιμές 0° - 180° Α ή 0° - 180° Δ.

Πάρα πολλά συστήματα γεωγραφικών συντεταγμένων μετατρέπουν τις συντεταγμένες σε δεκαδικούς αριθμούς με το γεωγραφικό πλάτος να παίρνει τιμές από -90 έως 90 και το γεωγραφικό μήκος από -180 έως 180 (Decimal degrees). Το ίδιο κάνουμε και εμείς στο σύστημά μας.

Για τον υπολογισμό της απόστασης χρησιμοποιούμε τον τύπο του Haversine ο οποίος δίνει την απόσταση μεταξύ δύο σημείων Α (φ1, λ1) και Β (φ2, λ2) πάνω σε μια σφαίρα (Γη) με γνωστή ακτίνα και δίνεται από τον τύπο:

$$d = 2 \cdot r \cdot arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos(\varphi_1)\, cos(\varphi_2)\, sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Εικόνα 33 Απόσταση 2 σημείων με τη χρήση γεωγραφικών συντεταγμένων

Σημείωση: Στον προηγούμενο τύπο r=6371 δηλαδή η ακτίνα της Γης σε χιλιόμετρα. Επειδή η Γη δεν είναι τέλεια σφαίρα η ακτίνα της Γης είναι κατά προσέγγιση.

6 Επιλογή - Selection – Reproduction

Αφού έχουμε αξιολογήσει όλα τα άτομα επιλέγουμε ποια από αυτά θα χρησιμοποιηθούν (αυτά που θα επιλεγούν ονομάζονται γονείς - parents) για να δημιουργήσουν απογόνους - offsprings.. Η επιλογή γίνεται τυχαία αλλά τα άτομα με καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιλεγούν. Μετά το τέλος της διαδικασίας δημιουργείται ένας νέος προσωρινός πληθυσμός ο οποίος χρησιμοποιείται για να γίνει η διασταύρωση και στη συνέχεια η μετάλλαξη ώστε να προκύψει ο νέος πληθυσμός.

Η επιλογή είναι μια διαδικασία η οποία μειώνει την ποικιλομορφία του πληθυσμού καθώς τα άτομα με καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιλεγούν δημιουργώντας έτσι αντίγραφα του εαυτού τους. Μετά την επιλογή η μέση απόδοση του πληθυσμού αυξάνεται αφού τα άτομα με χαμηλή απόδοση αντικαθίστανται από άτομα με υψηλότερη απόδοση.

Στην κλασική μορφή των γενετικών αλγορίθμων όλα τα νέα μέλη προκύπτουν από τη διαδικασία της επιλογής. Αυτό δεν συμβαίνει πάντα. Πολλές φορές κρατάμε το άτομο ή τα άτομα (λύσεις) με την καλύτερη απόδοση από την προηγούμενη γενιά διασφαλίζοντας έτσι ότι ο αλγόριθμος δε θα χάσει κάποιες καλές λύσεις. Βέβαια αν το κάνουμε αυτό για μεγάλο αριθμό ατόμων κινδυνεύουμε να περιορίσουμε το εύρος των λύσεων με αποτέλεσμα ο γενετικός αλγόριθμος να εγκλωβιστεί σε κάποιο τοπικό ακρότατο. Για αυτό το λόγο συνήθως μόνο ή λύση με τη μεγαλύτερη απόδοση μεταφέρεται στην επόμενη γενιά ή το πολύ ένα ποσοστό 10%. Όταν συμβαίνει αυτό τότε λέμε ότι ο γενετικός αλγόριθμος χρησιμοποιεί τη μέθοδο του ελιτισμού (elitism).

Σε κάποιες υλοποιήσεις Γ.Α. οι απόγονοι που προκύπτουν μετά το τέλος της διαδικασίας επιλογής/διασταύρωσης/μετάλλαξης, δεν αντικαθιστούν όλο τον αρχικό πληθυσμό. Αντίθετα γίνεται αξιολόγηση και τα άτομα με καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα να επιλεγούν. Αυτή η διαδικασία μπορεί να γίνεται συγκεντρωτικά ή και μεμονωμένα. Όταν γίνεται συγκεντρωτικά, δημιουργούνται όλοι οι απόγονοι και από το σύνολο των απογόνων και γονέων, επιλέγουμε τα N καταλληλότερα άτομα για να δημιουργήσουμε τη νέα γενιά. Η διαδικασία αυτή είναι γνωστή ως επιλογή επιζώντων (survivor selection) . Όταν γίνεται μεμονωμένα στην ουσία δεν υπάρχουν γενιές. Αν για παράδειγμα ο γενετικός αλγόριθμος δημιουργεί δύο απογόνους από δύο γονείς, με το που θα δημιουργηθούν οι δύο απόγονοι γίνεται επιλογή για το ποια άτομα του πληθυσμού θα αντικαταστήσουν. Σε αυτή τη μορφή του γενετικού αλγορίθμου, γονείς και απόγονοι συνυπάρχουν στην ίδια γενιά. Αυτή η διαδικασία είναι γνωστή ως επιλογή σταθερής κατάστασης (steady state selection).

Υπάρχουν πολλοί τρόποι με τους οποίους μπορεί να γίνει η επιλογή. Στη συνέχεια θα δούμε τους σημαντικότερους από αυτούς.

6.1 Αναλογική επιλογή καταλληλότητας - Fitness proportionate selection

Η μέθοδος “Fitness proportionate selection” αποτελεί την πιο διαδεδομένη μέθοδο επιλογής (Σημείωση: Κάποιοι ερευνητές θεωρούν την μέθοδο τουρνουά την πιο διαδεδομένη). Η πιθανότητα επιλογής ενός ατόμου i εξαρτάται από την απόδοσή του και δίνεται από τον τύπο $P_i = f_i / \sum_{j=1}^{N} f_j$, όπου f_i είναι η καταλληλότητα του ατόμου i και N ο συνολικός αριθμός των ατόμων. Πιο απλά η πιθανότητα επιλογής ενός ατόμου είναι το πηλίκο της απόδοσής του προς τη συνολική απόδοση του πληθυσμού. Επειδή θέλουμε να επιλέξουμε N άτομα, η διαδικασία πρέπει να επαναληφθεί N φορές.

Με την αναλογική επιλογή καταλληλότητας ένα άτομο μπορεί να επιλεγεί περισσότερες από μια φορές, αλλά μπορεί να μην επιλεγεί καμία φορά. Υπάρχει δηλαδή ο κίνδυνος το άτομο με την μεγαλύτερη απόδοση να χαθεί, ειδικά αν το μέγεθος του πληθυσμού είναι μικρό και ο κίνδυνος ο πληθυσμός να γεμίσει με αντίγραφα του ίδιου ατόμου, αν ένα άτομο έχει πολύ μεγαλύτερη απόδοση από τα υπόλοιπα.

Πλεονεκτήματα:

- Εύκολη στην κατανόηση και στην υλοποίηση.
- Δίνει την ευκαιρία σε όλα τα άτομα να επιλεγούν.
- Δίνει την ευκαιρία σε άτομα να επιλεγούν περισσότερες από μία φορά.

Μειονεκτήματα:

- Κίνδυνος πρόωρης σύγκλισης αν κάποιο χρωμόσωμα έχει πολύ μεγαλύτερη απόδοση από τα άλλα. Σε αυτή την περίπτωση μπορεί να αποκτήσουμε πολλά αντίγραφα του ίδιου ατόμου. Αν αυτή η λύση δεν είναι η βέλτιστη τότε ο αλγόριθμος εγκλωβίζεται σε ένα τοπικό ακρότατο αφού δεν θα εξετάσει όλο τον χώρο των πιθανών λύσεων.
- Αν όλα τα άτομα έχουν περίπου την ίδια απόδοση τότε έχουμε μια σχεδόν τυχαία επιλογή όπως θα δούμε πιο αναλυτικά στο κεφάλαιο "Πιθανολογική καθολική δειγματοληψία"
- Δεν μπορεί να χρησιμοποιηθεί αν η απόδοση μπορεί να πάρει αρνητικές τιμές.

Η μέθοδος "Fitness proportionate selection πολλές φορές αναφέρεται ως ρουλέτα επιλογής - roulette wheel selection, χωρίς όμως οι δύο έννοιες να ταυτίζονται. Η ρουλέτα επιλογής είναι ο αλγόριθμος, ο τρόπος δηλαδή με τον οποίο επιλέγεται ένα άτομο.. Επειδή ο Holland (Holland, 1975) στην έρευνά του χρησιμοποίησε τη ρουλέτα επιλογής, επικράτησε σαν όρος και πολλοί λένε ότι ο γενετικός αλγόριθμος που υλοποίησαν χρησιμοποιεί τη ρουλέτα επιλογής, παρόλο που η επιλογή των ατόμων γίνεται με κάποιο άλλο τρόπο όπως η δυαδική αναζήτηση. Για να αποφύγουμε τη σύγχυση που μπορεί να προκαλέσει ο διαχωρισμός τον δύο εννοιών σε κάποιον αναγνώστη, στη συνέχεια της εργασίας η ρουλέτα επιλογής και η αναλογική επιλογή καταλληλότητας θα σημαίνουν το ίδιο. Στη συνέχεια θα δούμε αναλυτικά όλους τους τρόπους με τους οποίους μπορούμε να επιλέξουμε ένα άτομο.

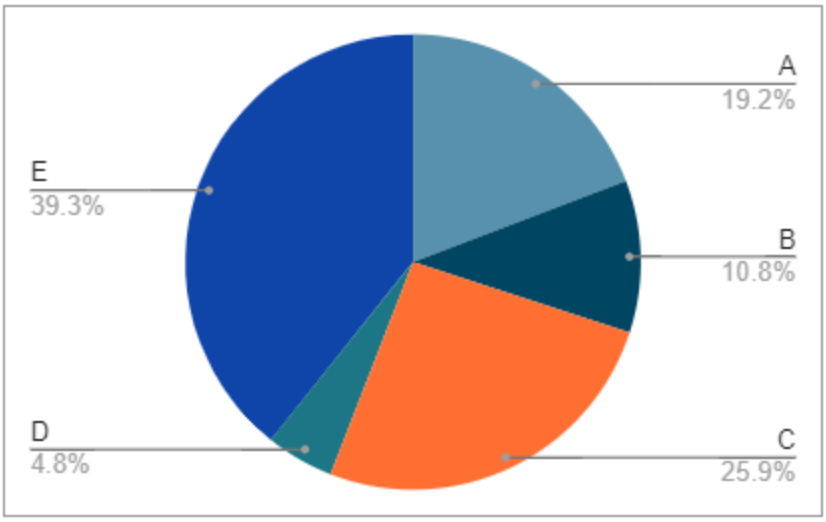
6.1.1 Ρουλέτα επιλογής - Roulette wheel selection

Σε αυτή τη μέθοδο μπορούμε να φανταστούμε ότι κάθε χρωμόσωμα καταλαμβάνει ένα ποσοστό μιας ρουλέτας ανάλογα με την απόδοσή του. Όσο μεγαλύτερη είναι η απόδοση τόσο μεγαλύτερο το μέρος της ρουλέτας που καταλαμβάνει. Για παράδειγμα έστω ότι έχουμε 5 χρωμοσώματα A B C D E με τις ακόλουθες αποδόσεις.

Χρωμόσωμα	Fitness f(x)	Percentage
A	13.5	19.20%
B	7.6	10.81%
C	18.2	25.89%
D	3.4	4.84%
E	27.6	39.26%

Εικόνα 34 Απόδοση χρωμοσωμάτων

Η ρουλέτα που προκύπτει φαίνεται στο ακόλουθο σχήμα.



Εικόνα 35 Ρουλέτα επιλογής

Κάθε φορά που θέλουμε να επιλέξουμε ένα άτομο περιστρέφουμε τη ρουλέτα. Αν θέλουμε να επιλέξουμε n άτομα περιστρέφουμε τη ρουλέτα n φορές. Η μέθοδος "ρουλέτα επιλογής" περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Υπολογίζουμε την συνολική απόδοση του πληθυσμού F δηλαδή το άθροισμα των αποδόσεων όλων των ατόμων.

Βήμα 2ο: Τοποθετούμε τα άτομα σε μια σειρά (πάνω στη ρουλέτα) και επιλέγουμε ένα τυχαίο αριθμό r στο διάστημα [0,F).

Βήμα 3ο: Θέτουμε S=0. Ξεκινώντας από την αρχή της σειράς που δημιουργήσαμε, επισκεπτόμαστε ένα ένα τα χρωμοσώματα προσθέτοντας στην S την τιμή της απόδοσής του. Όταν S>0 ο αλγόριθμος επιλέγει το χρωμόσωμα την απόδοση του οποίου πρόσθεσε τελευταία στην S.

Επαναλαμβάνουμε τη διαδικασία n φορές όπου n το μέγεθος του πληθυσμού (Αν έχουμε ελιτισμό αφαιρούμε από το n τον αριθμό των ατόμων που επιλέχθηκαν με ελιτισμό). Αυτή η μέθοδος είναι γνωστή και ως γραμμική αναζήτηση (linear search).

6.1.2 Δυαδική αναζήτηση

Το βασικό μειονέκτημα της ρουλέτας επιλογής είναι η ταχύτητα, καθώς στη χειρότερη περίπτωση έχει πολυπλοκότητα O(n) ενώ στη μέση περίπτωση O(n/2). Ένας εναλλακτικός τρόπος είναι να επιλέξουμε τα άτομα κάνοντας δυαδική αναζήτηση. Ο τρόπος αυτός έχει πολυπλοκότητα O(logn) και περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Τοποθετούμε τα άτομα σε μια σειρά και σε ένα πίνακα (array) L αποθηκεύουμε το άθροισμα της καταλληλότητας του ατόμου συν το άθροισμα της καταλληλότητας όλων των προηγούμενων ατόμων (CDF), με το πρώτο στοιχείο του array να είναι L[0]=0. Δηλαδή το L[1] θα περιέχει την απόδοση του πρώτου ατόμου, το L[2] το άθροισμα της απόδοσης του πρώτου και του δεύτερου ατόμου, και το τελευταίο στοιχείο του array θα περιέχει το άθροισμα των αποδόσεων όλων των ατόμων.

Βήμα 2ο: Διαιρούμε όλα τα στοιχεία του array L με το άθροισμα των αποδόσεων όλων των ατόμων.

Βήμα 3ο: Παράγουμε ένα τυχαίο αριθμό r στο διάστημα [0,1).

Βήμα 4ο: Κάνουμε δυαδική αναζήτηση για να βρούμε σε πιο στοιχείο του array αντιστοιχεί ο τυχαίος αριθμός r. Επαναλαμβάνουμε τη διαδικασία n φορές όπου n το μέγεθος του πληθυσμού.

6.1.3 Στοχαστική αποδοχή - Stochastic acceptance

Μια σχετικά πρόσφατη μέθοδος επιλογής ενός ατόμου είναι η μέθοδος της στοχαστικής αποδοχής (Adam Lipowski, 2011). Αν και στη θεωρία η επιλογή γίνεται σε χρόνο O(1), στην πράξη δεν είναι πάντα πιο γρήγορη από τη δυαδική αναζήτηση, αφού στην πραγματικότητα η πολυπλοκότητα είναι O(c). Η υλοποίηση του αλγορίθμου της στοχαστικής αποδοχής περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Βρίσκουμε τη μέγιστη απόδοση του πληθυσμού Fmax.

Βήμα 2ο: Παράγουμε έναν τυχαίο αριθμό r στο διάστημα [0,1).

Βήμα 3ο: Παίρνουμε τυχαία ένα άτομο X. Αν r<F(X)/Fmax το άτομο επιλέγεται, διαφορετικά επιστρέφουμε στο Βήμα 2. Επαναλαμβάνουμε τη διαδικασία n φορές όπου n το μέγεθος του πληθυσμού.

Όπως θα δούμε και στο κεφάλαιο ανάλυση μεθόδων, η στοχαστική αποδοχή είναι μια πολύ γρήγορη μέθοδος επιλογής, ειδικά στην περίπτωση κατά την οποία η επιλογή ενός ατόμου επηρεάζει την καταλληλότητα των υπολοίπων. Σε αυτή την περίπτωση αν χρησιμοποιήσουμε τη δυαδική αναζήτηση θα πρέπει να υπολογίσουμε ξανά την CDF, η ακόμα χειρότερα αν χρησιμοποιήσουμε τη μέθοδο alias να υπολογίσουμε το alias table.

Το μειονέκτημα της στοχαστικής αποδοχής είναι ότι για κάποιες κατανομές της καταλληλότητας, όπως η εκθετική κατανομή, μπορεί να είναι πιο αργή ακόμα και από τη μέθοδο γραμμικής αναζήτησης. Όταν η μέγιστη απόδοση του πληθυσμού είναι πολύ μεγαλύτερη από τις άλλες αποδόσεις, μπορεί να χρειαστεί να παραχθούν πάρα πολλοί τυχαία αριθμοί και η απόδοση του αλγορίθμου να μειωθεί.

6.1.4 Alias method

Οι προηγούμενες τρεις μέθοδοι που αναλύσαμε έχουν τα εξής προβλήματα:

- Η πολυπλοκότητά τους αυξάνεται όσο αυξάνεται το μέγεθος του πληθυσμού (linear search, binary search) με αποτέλεσμα να γίνονται πολύ αργές για μεγάλους πληθυσμούς.
- Δε δουλεύουν καλά για όλες τις κατανομές αποδόσεων (stochastic acceptance).

Τη λύση σε αυτά τα προβλήματα δίνει η alias method (Schwarz, 2011). Δε θα αναλύσουμε πως ακριβώς το κάνει καθώς είναι έξω από το αντικείμενο της μελέτης μας, αλλά θα δείξουμε μέσα από ένα παράδειγμα ότι η μέθοδος δουλεύει.

Στην αρχή (initialization stage) δημιουργούνται 2 array (alias και prob) με το prob να περιέχει πιθανότητες και το alias δείκτες στον prob. Αν υποθέσουμε ότι ο αρχικός μας πληθυσμός έχει 4 άτομα A B C και D με πιθανότητες επιλογής 10% 20% 30% και 40% probabilities = [0.1 , 0.2 , 0.3 , 0.4]. Μετά το initialization stage προκύπτουν οι εξής 2 πίνακες:

alias = [2 , 3 , 3]

prob = [0.4 , 0.8 , 0.6 , 1]

Παράγουμε ένα τυχαίο αριθμό i στο διάστημα [0,3] και έναν τυχαίο αριθμό j στο διάστημα [0,1). Συγκρίνουμε την τιμή prob[i] με τον j. Αν j < prob[i] τότε η συνάρτηση επιστρέφει i, διαφορετικά επιστρέφει alias[i].

Στο παράδειγμά μας:

- Για i=0 το prob[0] είναι 0.4 άρα η πιθανότητα να πάρουμε 0 (δηλαδή το A) είναι 0.4 και 0.6 να πάρουμε το alias[0]=2 (δηλαδή το C).
- Για i=1 το prob[1] είναι 0.8 άρα η πιθανότητα να πάρουμε 1 (δηλαδή το B) είναι 0.8 και 0.2 να πάρουμε το alias[1]=3 (δηλαδή το D).
- Για i=2 το prob[2] είναι 0.6 άρα η πιθανότητα να πάρουμε 2 (δηλαδή το C) είναι 0.6 και 0.4 να πάρουμε το alias[2]=3 (δηλαδή το D).
- Για i=3 το prob[3] είναι 1 άρα παίρνουμε πάντα 3 (δηλαδή το D).

Οπότε η πιθανότητα να πάρουμε το:

- Για i=0 το prob[0] είναι 0.4 άρα η πιθανότητα να πάρουμε 0 (δηλαδή το A) είναι 0.4 και 0.6 να πάρουμε το alias[0]=2 (δηλαδή το C).
- Για i=1 το prob[1] είναι 0.8 άρα η πιθανότητα να πάρουμε 1 (δηλαδή το B) είναι 0.8 και 0.2 να πάρουμε το alias[1]=3 (δηλαδή το D).
- Για i=2 το prob[2] είναι 0.6 άρα η πιθανότητα να πάρουμε 2 (δηλαδή το C) είναι 0.6 και 0.4 να πάρουμε το alias[2]=3 (δηλαδή το D).
- Για i=3 το prob[3] είναι 1 άρα παίρνουμε πάντα 3 (δηλαδή το D).

Η διαδικασία αρχικοποίησης έχει πολυπλοκότητα Θ(n), αλλά η κάθε αναζήτηση Θ(1). Όταν η απόδοση των ατόμων του πληθυσμού επηρεάζεται από την επιλογή ενός ατόμου, η μέθοδος alias πρέπει να κάνει initialization σε κάθε επιλογή με αποτέλεσμα να γίνεται πολύ αργή.

6.1.5 Σύγκριση μεθόδων

Σε αυτό το κεφάλαιο θα δείξουμε πρακτικά ότι η συχνότητα επιλογής των ατόμων και με τις τέσσερις μεθόδους είναι η αναμενόμενη. Επίσης θα δούμε πόσο χρόνο χρειάζονται στην πράξη και οι τέσσερις μέθοδοι για να επιλέξουν ένα πολύ μεγάλο αριθμό ατόμων.

Έστω ότι έχουμε 10 άτομα [0,1,2,3,4,5,6,7,8,9] με απόδοση [1,3,5,7,10,11,13,15,16,19] αντίστοιχα. Η συνολική απόδοση του πληθυσμού είναι 100 άρα η πιθανότητα επιλογής του κάθε ατόμου είναι [1%, 3%, 5%, 7%, 10%, 11%, 13%, 15%, 16%, 19%]. Θα επιλέξουμε 10.000.000 άτομα με κάθε μία από τις τέσσερις μεθόδους.

Χρωμοσώματα με linear search:

Πλήθος 0 = 99497 - Πιθανότητα 0.99497%
Πλήθος 1 = 300631 - Πιθανότητα 3.00631%
Πλήθος 2 = 500296 - Πιθανότητα 5.00296%
Πλήθος 3 = 700012 - Πιθανότητα 7.00012%
Πλήθος 4 = 1000297 - Πιθανότητα 10.00297%
Πλήθος 5 = 1100490 - Πιθανότητα 11.0049%
Πλήθος 6 = 1299092 - Πιθανότητα 12.99092%
Πλήθος 7 = 1499787 - Πιθανότητα 14.99787%
Πλήθος 8 = 1599928 - Πιθανότητα 15.99928%
Πλήθος 9 = 1899970 - Πιθανότητα 18.9997%

Εικόνα 36 Επιλογή 10.000.000 ατόμων με linear search

Χρωμοσώματα με binary search:

Πλήθος 0 = 99996 - Πιθανότητα 0.99996%
Πλήθος 1 = 299210 - Πιθανότητα 2.9921%
Πλήθος 2 = 499308 - Πιθανότητα 4.99308%
Πλήθος 3 = 699183 - Πιθανότητα 6.99183%
Πλήθος 4 = 1000106 - Πιθανότητα 10.00106%
Πλήθος 5 = 1099692 - Πιθανότητα 10.99692%
Πλήθος 6 = 1300265 - Πιθανότητα 13.00265%
Πλήθος 7 = 1500994 - Πιθανότητα 15.00994%
Πλήθος 8 = 1600574 - Πιθανότητα 16.00574%
Πλήθος 9 = 1900672 - Πιθανότητα 19.00672%

Εικόνα 37 Επιλογή 10.000.000 ατόμων με binary search

Χρωμοσώματα με stochastic acceptance:

Πλήθος 0 = 100421 - Πιθανότητα 1.00421%
Πλήθος 1 = 299286 - Πιθανότητα 2.99286%
Πλήθος 2 = 500114 - Πιθανότητα 5.00114%
Πλήθος 3 = 700455 - Πιθανότητα 7.00455%
Πλήθος 4 = 998898 - Πιθανότητα 9.98898%
Πλήθος 5 = 1098864 - Πιθανότητα 10.98864%
Πλήθος 6 = 1301663 - Πιθανότητα 13.01663%
Πλήθος 7 = 1500518 - Πιθανότητα 15.00518%
Πλήθος 8 = 1599728 - Πιθανότητα 15.99728%
Πλήθος 9 = 1900053 - Πιθανότητα 19.00053%

Εικόνα 38 Επιλογή 10.000.000 ατόμων με stochastic acceptance

Χρωμοσώματα με alias method:

Πλήθος 0 = 100096 - Πιθανότητα 1.00096%
Πλήθος 1 = 299338 - Πιθανότητα 2.99338%
Πλήθος 2 = 500167 - Πιθανότητα 5.00167%
Πλήθος 3 = 699515 - Πιθανότητα 6.99515%
Πλήθος 4 = 1000261 - Πιθανότητα 10.00261%
Πλήθος 5 = 1100551 - Πιθανότητα 11.00551%
Πλήθος 6 = 1298612 - Πιθανότητα 12.98612%
Πλήθος 7 = 1499641 - Πιθανότητα 14.99641%
Πλήθος 8 = 1600127 - Πιθανότητα 16.00127%
Πλήθος 9 = 1901692 - Πιθανότητα 19.01692%

Εικόνα 39 Επιλογή 10.000.000 ατόμων με alias method

Παρατηρούμε ότι και στις τέσσερις περιπτώσεις η συχνότητα εμφάνισης των ατόμων ήταν η αναμενόμενη. Η μικρή απόκλιση που υπάρχει είναι μέσα στα όρια του στατιστικού λάθους. Σειρά έχει η ανάλυση του χρόνου εκτέλεσης. Για την ανάλυση θα κάνουμε τα εξής πειράματα τα οποία θα επαναλάβουμε 10 φορές και θα πάρουμε το μέσο όρο εκτέλεσης.

- **Πείραμα Α**. Πληθυσμός 10 ατόμων. Η απόδοση των ατόμων αυξάνεται γραμμικά 1,2...9. Παράγουμε 50.000 τέτοια άτομα (Μελέτη του χρόνου για μικρό Ν, μικρό αριθμό επαναλήψεων και γραμμική αύξηση της απόδοσης).
- **Πείραμα Β**. Πληθυσμός 10 ατόμων. Η απόδοση των ατόμων αυξάνεται γραμμικά 1,2...9. Παράγουμε 5.000.000 τέτοια άτομα (Μελέτη του χρόνου για μικρό Ν, μεγάλο αριθμό επαναλήψεων και γραμμική αύξηση της απόδοσης).
- **Πείραμα C**. Πληθυσμός 10 ατόμων. Η απόδοση των ατόμων αυξάνεται γραμμικά 1,2...9. Παράγουμε 50.000 τέτοια άτομα. Η απόδοση των ατόμων δεν παραμένει σταθερή. (Μελέτη του χρόνου για μικρό Ν, μικρό αριθμό επαναλήψεων και γραμμική αύξηση της απόδοσης. Η απόδοση αλλάζει).
- **Πείραμα D**. Πληθυσμός 100 ατόμων. Η απόδοση των ατόμων αυξάνεται γραμμικά 1,2...98,99. Παράγουμε 5.000.000 τέτοια άτομα (Μελέτη του χρόνου για μεγάλο Ν, μικρό μεγάλο επαναλήψεων και γραμμική αύξηση της απόδοσης).
- **Πείραμα Ε**. Πληθυσμός 100 ατόμων. Η απόδοση των ατόμων αυξάνεται εκθετικά 1,2, ... ,298,299. Παράγουμε 5.000.000 τέτοια άτομα (Μελέτη του χρόνου για μεγάλο Ν, μεγάλο αριθμό επαναλήψεων και εκθετική αύξηση της απόδοσης).
- **Πείραμα F**. Πληθυσμός 100 ατόμων. Η απόδοση των ατόμων αυξάνεται γραμμικά 1,2...9. Παράγουμε 50.000 τέτοια άτομα. (Μελέτη του χρόνου για μεγάλο Ν, μικρό αριθμό επαναλήψεων και γραμμική αύξηση της απόδοσης).

Τα αποτελέσματα των πειραμάτων για τη μέθοδο binary search είναι:

- Πείραμα A binary search 0.03658787727 sec
- Πείραμα B binary search 3.424004521 sec
- Πείραμα C binary search 0.101601143 sec
- Πείραμα D binary search 4.747312164 sec
- Πείραμα E binary search 4.253465614 sec
- Πείραμα F binary search 0.05860632896 sec

Τα αποτελέσματα των πειραμάτων για την alias method είναι:

- Πείραμα A alias method 0.02288665771 sec
- Πείραμα B alias method 1.965926448 sec
- Πείραμα C alias method 0.3698747158 sec
- Πείραμα D alias method 1.965547681 sec
- Πείραμα E alias method 2.065620613 sec
- Πείραμα F alias method 0.02077710629 sec

Τα αποτελέσματα των πειραμάτων με τη μέθοδο linear search είναι:

- Πείραμα A linear search 0.02588333024 sec
- Πείραμα B linear search 2.44629788399 sec
- Πείραμα C linear search 0.0420620441437 sec
- Πείραμα D linear search 14.1896623 sec
- Πείραμα E linear search 20.4362750053 sec
- Πείραμα F linear search 0.15113384819 sec

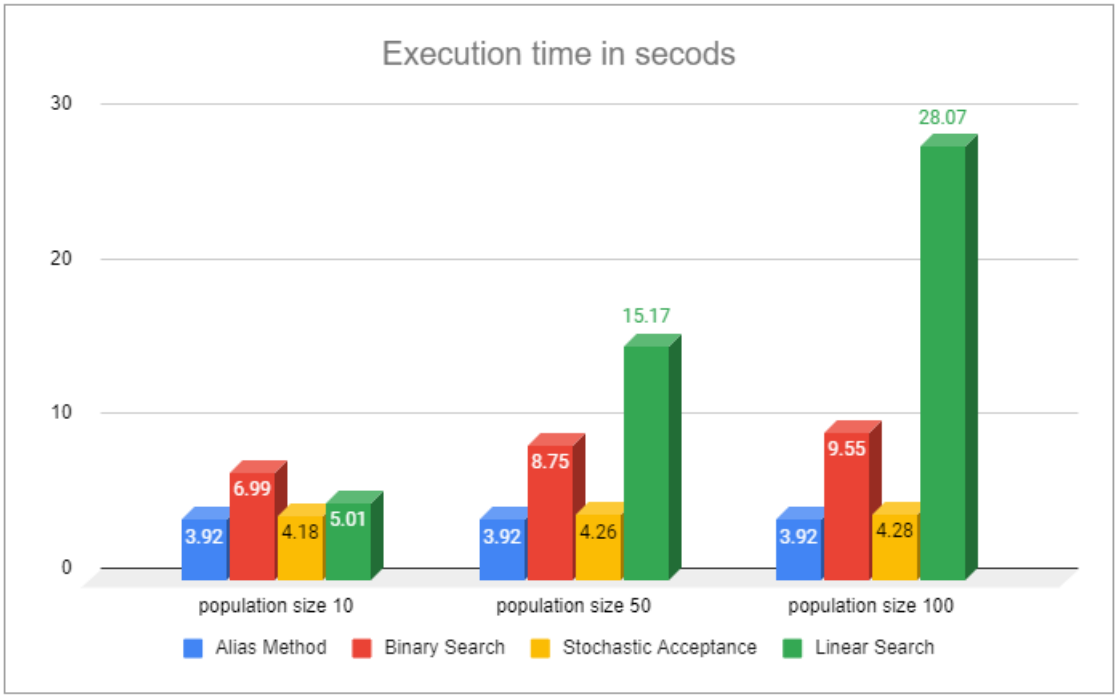
Τα αποτελέσματα των πειραμάτων με τη μέθοδο stochastic acceptanceείναι:

- Πείραμα A stochastic acceptance 0.0204758644104 sec
- Πείραμα B stochastic acceptance 2.00975298882 sec
- Πείραμα C stochastic acceptance 0.020925865638 sec
- Πείραμα D stochastic acceptance 2.09460186958 sec
- Πείραμα E stochastic acceptance 38.0461602211 sec
- Πείραμα F stochastic acceptance 0.0309820175171 sec

Τα συμπεράσματα που βγάζουμε από τη μελέτη είναι τα εξής:

- Αν η απόδοση των ατόμων δεν αλλάζει, οπότε δεν χρειάζεται να δημιουργούμε κάθε φορά το alias table, η μέθοδος alias αποτελεί την καλύτερη επιλογή εκτός και αν ο πληθυσμός είναι πάρα πολύ μικρός. Σε αυτή την περίπτωση η μέθοδος stochastic acceptance, μπορεί να είναι καλύτερη (εξαρτάται από την κατανομή των αποδόσεων)
- Η μέθοδος stochastic acceptance δεν επηρεάζεται από το μέγεθος του πληθυσμού, δεν έχει πρόβλημα όταν αλλάζει η απόδοση των ατόμων, δε δεσμεύει μνήμη και είναι πολύ απλή στην υλοποίηση και την κατανόηση. Το βασικό της μειονέκτημα είναι ότι είναι εξαιρετικά ευάλωτη στην κατανομή της απόδοσης. Αν η μέγιστη απόδοση του πληθυσμού είναι πολύ μεγαλύτερη από την απόδοση των άλλων ατόμων τότε ο αλγόριθμος πρέπει να παράγει ένα πολύ μεγάλο αριθμό τυχαίων αριθμών, με αποτέλεσμα να καθυστερεί πολύ.
- Η μέθοδος linear search επηρεάζεται πολύ από το μέγεθος του πληθυσμού αφού έχει πολυπλοκότητα O(n). Επίσης αρνητική επίδραση στην απόδοση της μεθόδου έχει και η εκθετική κατανομή των αποδόσεων. Γενικά δε θα πρέπει να χρησιμοποιείται παρά μόνο στην περίπτωση πολύ μικρών πληθυσμών (μέχρι 10-15).
- Η μέθοδος binary search είναι η μόνη μέθοδος που ευνοείται από την εκθετική κατανομή των αποδόσεων. Δεν επηρεάζεται τόσο από το μέγεθος του πληθυσμού όσο η μέθοδος linear search καθώς η πολυπλοκότητά της είναι O(logn), αλλά δεν υπάρχει κανένας λόγος να τη χρησιμοποιήσουμε όταν μπορούμε να χρησιμοποιήσουμε τη μέθοδο alias.

Στο σχήμα που ακολουθεί φαίνεται η απόδοση των τεσσάρων μεθόδων στην επιλογή 10.000.000 ατόμων, για πληθυσμούς 10, 50 και 100 ατόμων, με την απόδοση των ατόμων να δίνεται από τον τύπο $F_{I+1} = F_I + 2$ και $F_I = 1$.



Εικόνα 40 Σύγκριση linear search, binary search stochastic acceptance και alias method

- Η υλοποίηση του αλγορίθμου σε PHP για την alias method βρίσκεται εδώ.
- Η υλοποίηση του αλγορίθμου σε PHP με binary search βρίσκεται εδώ.
- Η υλοποίηση του αλγορίθμου σε PHP με linear search βρίσκεται εδώ.
- Η υλοποίηση του αλγορίθμου σε PHP με stochastic acceptance βρίσκεται εδώ.

6.2 Πιθανολογική καθολική δειγματοληψία - Stochastic universal sampling (SUS)

Η μέθοδος επιλογής stochastic universal sampling (Baker J. E., 1987) μοιάζει πολύ με τη μέθοδο fitness proportionate selection, αλλά όπως θα δούμε στη συνέχεια, είναι πιο γρήγορη καθώς χρειαζόμαστε μόνο ένα τυχαίο αριθμό για να επιλέξουμε όλα τα άτομα και πιο δίκαιη. Η επιλογή των ατόμων γίνεται με βάση την απόδοσή τους.

Πιο συγκεκριμένα στην αρχή υπολογίζουμε την απόδοση όλων των ατόμων (χρωμοσωμάτων) του πληθυσμού και την κανονικοποιούμε, δηλαδή διαιρούμε την κάθε απόδοση με τη συνολική απόδοση του πληθυσμού, ώστε το άθροισμα να είναι ίσο με 1, παίρνοντας έτσι την πιθανότητα επιλογής του κάθε χρωμοσώματος. Στη συνέχεια υπολογίζουμε την αθροιστική πιθανότητα του κάθε ατόμου, δηλαδή το άθροισμα των πιθανοτήτων των ατόμων πριν από αυτό και παράγουμε ένα τυχαίο αριθμό στο διάστημα [0,1/N], όπου N ο αριθμός των ατόμων που θέλουμε να επιλέξουμε. Αυτός ο αριθμός είναι ο δείκτης του πρώτου ατόμου που θα επιλεγεί, δηλαδή αν ο τυχαίος αριθμός είναι ο Α επιλέγουμε το χρωμόσωμα του οποίου η αθροιστική πιθανότητα είναι μικρότερη από το Α και το Α δεν είναι μεγαλύτερο από την αθροιστική πιθανότητα κάποιου άλλου ατόμου. Επαναλαμβάνουμε την διαδικασία N-1 φορές, προσθέτοντας κάθε φορά τον αριθμό 1/N στον Α, επιλέγοντας έτσι Ν άτομα.

Για παράδειγμα έστω ότι έχουμε 5 χρωμοσώματα A B C D E με τις ακόλουθες αποδόσεις.

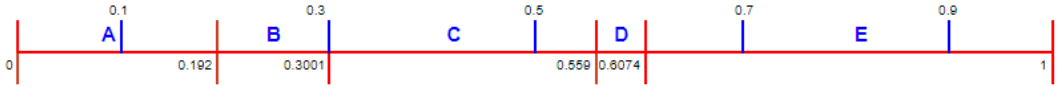
Χρωμόσωμα	Fitness f(x)	Percentage
A	13.5	19.20%
B	7.6	10.81%
C	18.2	25.89%
D	3.4	4.84%
E	27.6	39.26%

Εικόνα 41 Αποδόσεις χρωμοσωμάτων

Τοποθετούμε τα άτομα πάνω σε μια ευθεία γραμμή με βάση την αθροιστική πιθανότητα κατά τέτοιο τρόπο ώστε το μήκος του τμήματος που αντιστοιχεί σε κάθε άτομο να είναι ανάλογο της απόδοσής του.

Έστω ότι θέλουμε να επιλέξουμε 5 άτομα (N=5). Παράγουμε ένα τυχαίο αριθμό στο διάστημα [0,1/N] δηλαδή στο διάστημα [0,0.2]. Έστω ότι ο τυχαίο αριθμός είναι το 0.1. Αυτός είναι ο πρώτος δείκτης και μας δίνει το άτομο A.

Τα υπόλοιπα χρωμοσώματα επιλέγονται προσθέτοντας κάθε φορά τον αριθμό 1/N=0.2 στον τυχαίο αριθμό 0.1. Έτσι προκύπτουν οι δείκτες 0.3 0.5 0.7 0.9 οι οποίοι μας δίνουν τα άτομα B, C, E, E. Επομένως τα 5 άτομα που επιλέγονται είναι τα A, B, C, E, E.



Εικόνα 42 Stochastic universal sampling

Μπορεί η μέθοδος SUS να φαίνεται ίδια με τη μέθοδο FPS, αλλά αυτό δεν ισχύει. Η πιθανότητα επιλογής των ατόμων και με τις δύο μεθόδους είναι η ίδια, δηλαδή αν επιλέξουμε άπειρα άτομα το αποτέλεσμα των δύο μεθόδων θα είναι το ίδιο, όμως με τη μέθοδο FPS μπορεί:

- Κάποια άτομα με υψηλή απόδοση να επιλέγουν πολλές φορές με αποτέλεσμα να γεμίσουν τον πληθυσμό με αντίγραφά τους και να οδηγήσουν τον γενετικό αλγόριθμο σε ένα τοπικό ακρότατο.
- Κάποια άτομα με υψηλή απόδοση να μην επιλεγούν καμία φορά και τη θέση τους να πάρουν άτομα με χαμηλή απόδοση, με αποτέλεσμα να καθυστερήσει η σύγκλιση.

Για να γίνει πιο κατανοητό. Έστω ότι έχουμε τα 5 χρωμοσώματα A, B, C, D και E του προηγούμενου παραδείγματος. Χρησιμοποιώντας τη διωνυμική κατανομή $C_k^n p^k (1-p)^{n-k}$ θα βρούμε τις εξής πιθανότητες.

- Και τα 5 να είναι E. $C_5^5 0.3926^5 (1-0.3926)^{5-5} = 0.9327\%$
- Κανένα να μην είναι E. $C_{05}^5 0.3926^0 (1-0.3926)^{5-0} = 8.2675\%$

Παρατηρούμε ότι παρόλο που το E έχει 39.26% πιθανότητα να επιλεγεί, αρκετές φορές (8.2675%) ο πληθυσμός που θα επιλέξουμε δεν θα περιέχει κανένα αντίγραφο από το καλύτερο άτομο και κάποιες φορές (0.9327%) θα περιέχει 5 φορές το ίδιο άτομο. Επίσης με μια μικρή πιθανότητα μπορεί να επιλέξουμε 3,4 ή ακόμα και 5 φορές το ίδιο άτομο. Με τη μέθοδο SUS δεν έχουμε αυτό το πρόβλημα. Το D δεν θα επιλεγεί ποτέ περισσότερες από μία φορές αφού τα διαστήματα 1/N είναι μεγαλύτερα από το μήκος του D. Το C δε θα επιλεγεί ποτέ λιγότερες από μία φορές και περισσότερες από δύο φορές. Ο αλγόριθμος SUS περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Υπολογίζουμε την συνολική απόδοση του πληθυσμού δηλαδή το άθροισμα των αποδόσεων όλων των ατόμων.

Βήμα 2ο: Υπολογίζουμε την πιθανότητα επιλογής του κάθε ατόμου δηλαδή το κλάσμα απόδοση του ατόμου προς συνολική απόδοση.

Βήμα 3ο: Υπολογίζουμε την αθροιστική (cumulative) πιθανότητα κάθε ατόμου.

Βήμα 4ο: Παράγουμε ένα τυχαίο αριθμό r ανάμεσα στο 0 και το 1.

Βήμα 5ο: Συγκρίνουμε τον r με τις αθροιστικές πιθανότητες των ατόμων και επιλέγουμε το πρώτο άτομο του οποίου η αθροιστική πιθανότητα είναι μεγαλύτερη από τον r.

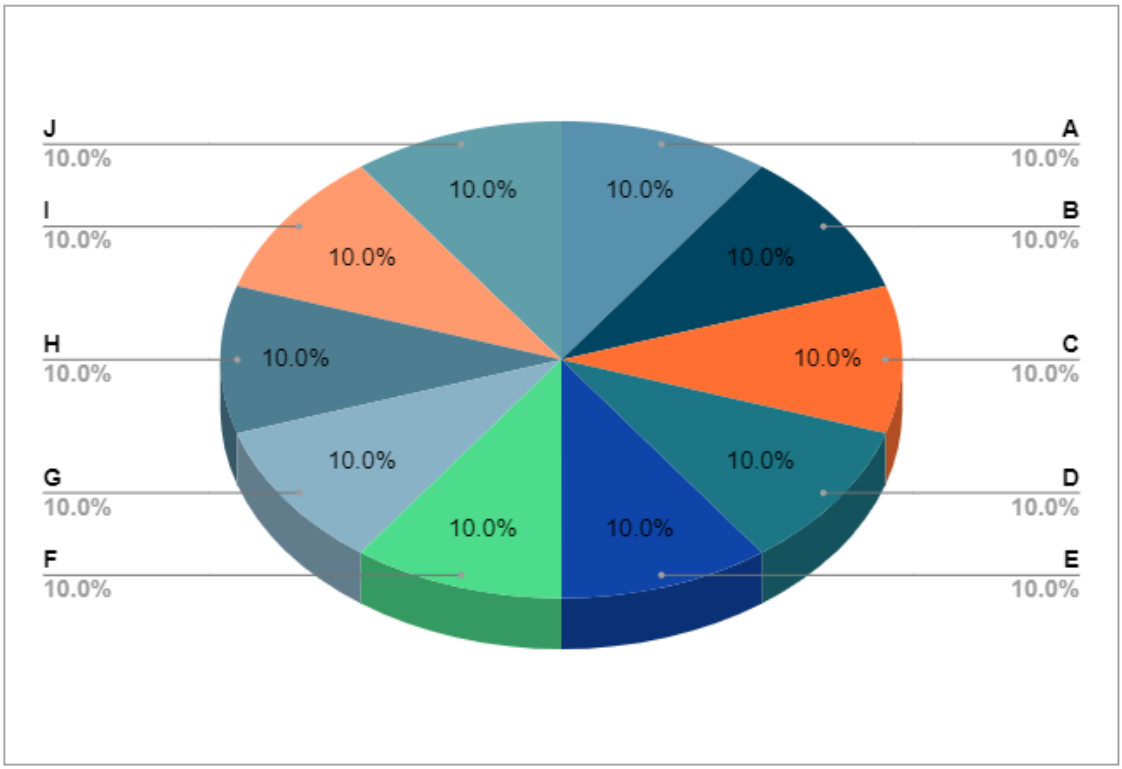
Βήμα 6ο: Προσθέτουμε στον r τον αριθμό 1/N όπου N ο αριθμός των ατόμων που θέλουμε να επιλέξουμε δηλαδή r=r+1/N και επιστρέφουμε στο Βήμα 5 επαναλαμβάνοντας τη διαδικασία N-1 φορές.

Η υλοποίηση του αλγορίθμου σε PHP βρίσκεται εδώ.

6.3 Επιλογή κατάταξης - Rank Selection

Οι μέθοδοι επιλογής που βασίζονται στην απόδοση των ατόμων έχουν πρόβλημα όταν οι αποδόσεις των ατόμων είναι σχεδόν ίδιες, ή όταν ένα άτομο έχει πολύ μεγαλύτερη απόδοση από τα υπόλοιπα. Για να λύσει αυτά τα προβλήματα ο Baker (Baker J. E., 1985) δημιούργησε την επιλογή κατάταξης. Πριν αναλύσουμε τη μέθοδο επιλογής κατάταξης θα δούμε αναλυτικά τα δύο προβλήματα που μπορεί να προκύψουν όταν η επιλογή γίνει με τη μέθοδο της ρουλέτας.

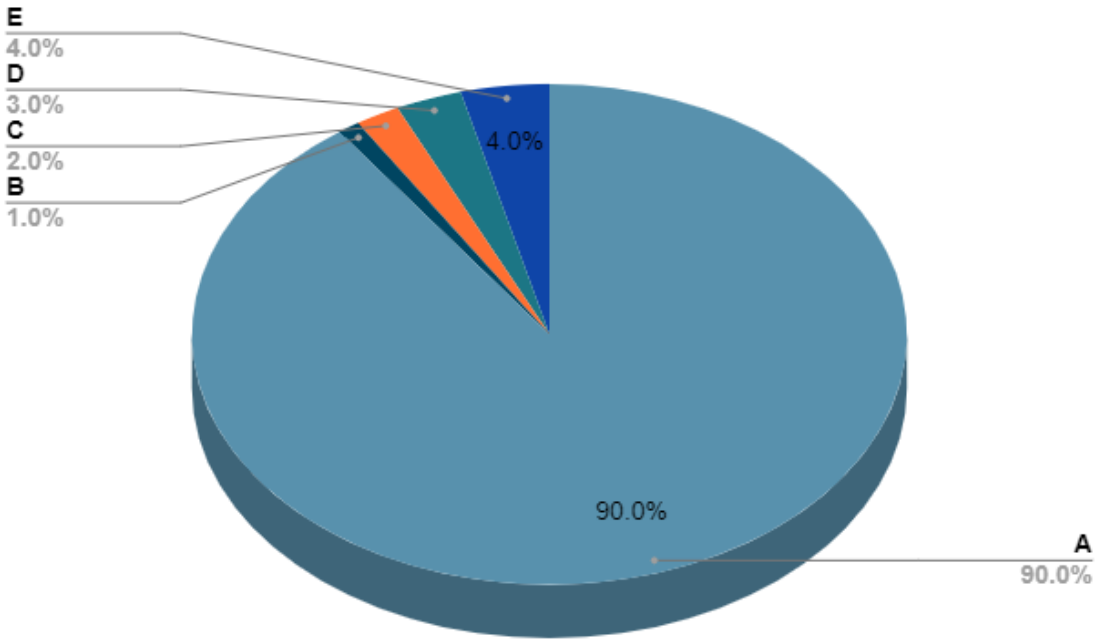
Έστω ότι έχουμε 10 χρωμοσώματα με αποδόσεις: 99.99 100.01 99.98 100.02 99.97 100.03 99.96 100.04 99.95 και 100.05. Η συνολική απόδοση του πληθυσμού είναι 1000. Σε κάθε περιστροφή της ρουλέτας το χρωμόσωμα με την χειρότερη απόδοση (99.95) έχει πιθανότητα 9.995% να επιλεγεί ενώ αυτό με την καλύτερη (100.05) 10.005%. Εύκολα γίνεται αντιληπτό ότι πλέον μιλάμε για σχεδόν τυχαία επιλογή. Στο σχήμα που ακολουθεί φαίνεται πως θα ήταν η ρουλέτα με τις αποδόσεις.



Εικόνα 43 Ρουλέτα επιλογής όταν η καταλληλότητα των ατόμων είναι σχεδόν ίδια

Το δεύτερο πρόβλημα προκύπτει όταν κάποιο από τα άτομα έχει πολύ μεγαλύτερη απόδοση από τα άλλα με αποτέλεσμα να επιλέγεται συνέχεια. Έτσι μπορεί ο αλγόριθμος να εγκλωβιστεί σε ένα τοπικό ακρότατο αν ο πληθυσμός γεμίσει με αντίγραφα αυτού του ατόμου. Για παράδειγμα έστω ότι έχουμε 5 άτομα με αποδόσεις: 90,1, 2, 3 και 4. Η συνολική απόδοση είναι 100, η πιθανότητα να επιλεγεί

το άτομο με απόδοση 90 είναι 90% ενώ τα άλλα έχουν πιθανότητα από 1% έως 4%. Στο σχήμα που ακολουθεί φαίνεται πως θα ήταν η ρουλέτα με τις αποδόσεις.



Εικόνα 44 Ρουλέτα όταν ένα άτομο έχει πολύ μεγαλύτερη απόδοση

Σε αυτή την περίπτωση, ειδικά αν το μέγεθος του πληθυσμού είναι μικρό, ο Γ.Α. μπορεί πολύ εύκολα να γεμίσει με αντίγραφα του ατόμου Α. Τη λύση σε αυτά τα προβλήματα έχετε να δώσει η επιλογή κατάταξης. Αρχικά υπολογίζουμε την απόδοση του κάθε ατόμου και στη συνέχεια ταξινομούμε τα άτομα με βάση την απόδοση. Στη συνέχεια χρησιμοποιούμε την επιλογή ρουλέτας, μόνο που για "απόδοση" χρησιμοποιούμε τη θέση (ranking) που βρίσκεται το κάθε άτομο στην ταξινομημένη λίστα με τις αποδόσεις. Για τον υπολογισμό της "απόδοσης" των ατόμων με βάση την κατάταξη έχουμε πολλές επιλογές αλλά οι δύο πιο διαδεδομένες είναι οι εξής.

6.3.1 Γραμμική επιλογή κατάταξης

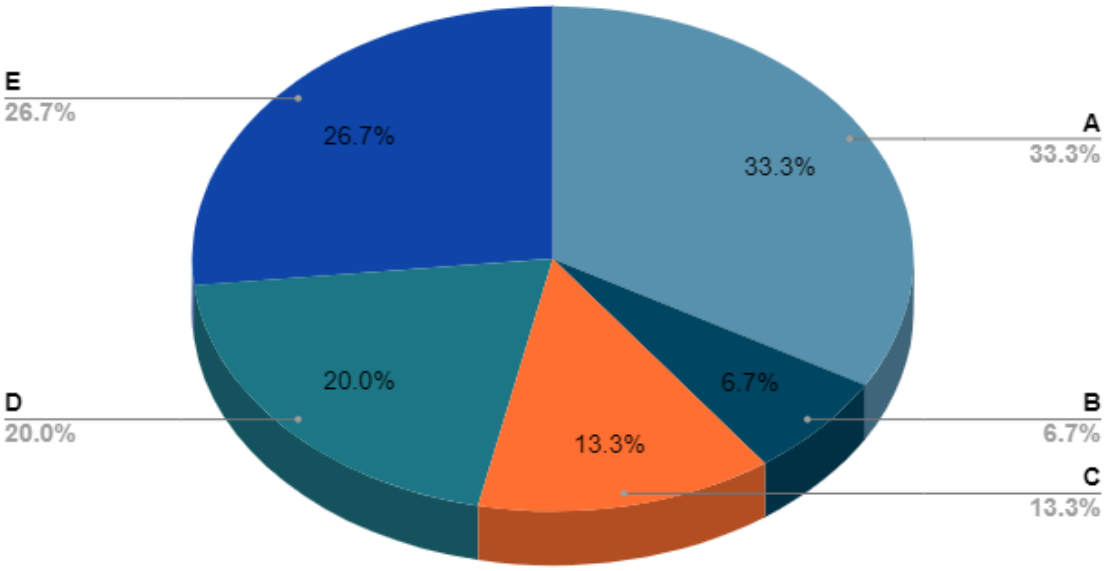
Στη γραμμική επιλογή κατάταξης (Tobias Blickle, 1995) (Anupriya Shukla, 2015) [(Mandira Chakraborty, 1999) η πιθανότητα επιλογής των ατόμων με βάση την κατάταξη δίνεται από τον τύπο $P(i) = \frac{1}{N}(min + (max - min) * \frac{i-1}{N-1})$ (1), όπου N το μέγεθος του πληθυσμού, i η κατάταξη του ατόμου στην ταξινομημένη λίστα αποδόσεων και max, min σταθερές που καθορίζουν την πίεση της επιλογής με max + min=2 και $1 \leq max \leq 2$.

Σημείωση: Σε πολλές έρευνες η πιθανότητα επιλογής των ατόμων στη γραμμική επιλογή κατάταξης δίνεται από τον τύπο $P(i) = i/\sum_{j=1}^{N} j$ (2), δηλαδή το άτομο που βρίσκεται στη χειρότερη θέση στην ταξινομημένη λίστα των αποδόσεων έχει απόδοση 1, αυτό που βρίσκεται στη δεύτερη χειρότερη θέση έχει απόδοση 2 και αυτό που βρίσκεται στην καλύτερη έχει απόδοση N.

Στο σχήμα που ακολουθεί φαίνεται η πιθανότητα επιλογής των ατόμων για max=2 και max=5/3, όταν ένα άτομο έχει πολύ μεγαλύτερη απόδοση. Για max=5/3 από 90% που ήταν η πιθανότητα να επιλέξουμε το χρωμόσωμα Α, έπεσε στο 33.33% ενώ για το χρωμόσωμα Β με τη χειρότερη απόδοση η πιθανότητα ανέβηκε από 1% σε 6.67%. Έτσι μειώνεται ο κίνδυνος να γεμίσει ο πληθυσμός με αντίγραφα του ατόμου Α.

Χρωμόσωμα	Fitness	Ranking	max=2	max=5/3
A	90	5	40.00%	33.33%
B	1	1	0.00%	6.67%
C	2	2	10.00%	13.33%
D	3	3	20.00%	20.00%
E	4	4	30.00%	26.67%

Εικόνα 45 Γραμμική επιλογή κατάταξης όταν ένα άτομο έχει πολύ μεγαλύτερη απόδοση

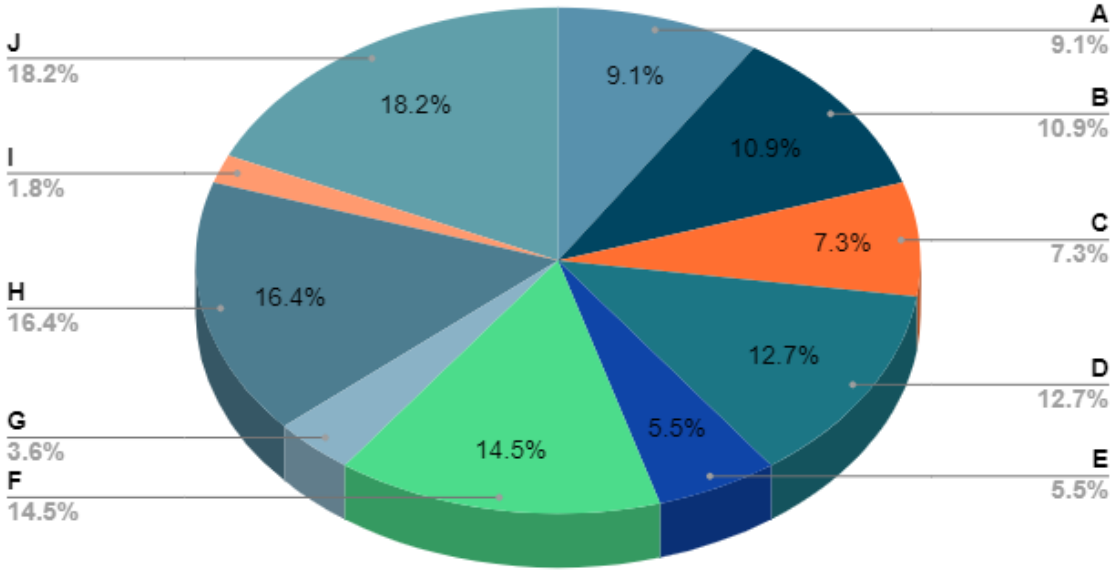


Εικόνα 46 Γραμμική επιλογή κατάταξης όταν ένα άτομο έχει πολύ μεγαλύτερη απόδοση για max=5/3

Στο σχήμα που ακολουθεί φαίνεται η πιθανότητα επιλογής των ατόμων για max=2 και max=20/11, όταν όλα τα άτομα έχουν περίπου την ίδια απόδοση. Για max=20/11 από περίπου 10% που ήταν η πιθανότητα να επιλέξουμε όλα τα άτομα, το άτομο με την καλύτερη απόδοση έχει πιθανότητα 18.18% και αυτό με την χειρότερη 1.82%.

Χρωμόσωμα	Fitness	Ranking	max=20/11	max=2
A	99.99	5	9.09%	8.89%
B	100.01	6	10.91%	11.11%
C	99.98	4	7.27%	6.67%
D	100.02	7	12.73%	13.33%
E	99.97	3	5.45%	4.44%
F	100.03	8	14.55%	15.56%
G	99.96	2	3.64%	2.22%
H	100.04	9	16.36%	17.78%
I	99.95	1	1.82%	0.00%
J	100.05	10	18.18%	20.00%

Εικόνα 47 Γραμμική επιλογή κατάταξης όταν όλα τα άτομα έχουν περίπου την ίδια απόδοση



Εικόνα 48 Γραμμική επιλογή κατάταξης όταν όλα τα άτομα έχουν περίπου την ίδια απόδοση max=20/11.

Ο αλγόριθμος της γραμμικής επιλογής κατάταξης περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Υπολογίζουμε την απόδοση όλων των ατόμων του πληθυσμού και αποθηκεύουμε τα αποτελέσματα σε μία λίστα.

Βήμα 2ο: Ταξινομούμε τη λίστα με βάση την απόδοση.

Βήμα 3ο: Υπολογίζουμε την πιθανότητα επιλογής των ατόμων με βάση τον τύπο (1).

Βήμα 4ο: Χρησιμοποιούμε τη ρουλέτα επιλογής, χρησιμοποιώντας για αποδόσεις τις πιθανότητες επιλογής των ατόμων που υπολογίσαμε στο Βήμα 3.

6.3.2 Εκθετική επιλογή κατάταξης

Στην εκθετική επιλογή κατάταξης (Tobias Blickle, 1995) η πιθανότητα επιλογής των ατόμων με βάση την κατάταξη δίνεται από τον τύπο $P(i) = \frac{C-1}{C^N-1} C^{N-1}$ (3), όπου N το μέγεθος του πληθυσμού και C σταθερά με C(0,1). Η μόνη διαφορά του αλγορίθμου εκθετικής επιλογής κατάταξης από τον αλγόριθμο της γραμμικής επιλογής κατάταξης, είναι ότι στο Βήμα 3 χρησιμοποιούμε τη σχέση (3) αντί για τη σχέση (1).

Στο σχήμα που ακολουθεί φαίνεται η πιθανότητα επιλογής των ατόμων για C=0.5 και C=0.7 όταν ένα άτομο έχει πολύ μεγαλύτερη πιθανότητα επιλογής.

Χρωμόσωμα	Fitness	Ranking	C=0.7	C=0.5
A	90	5	36.06%	51.61%
B	1	1	8.66%	3.23%
C	2	2	12.37%	6.45%
D	3	3	17.67%	12.90%
E	4	4	25.24%	25.81%

Εικόνα 49 Εκθετική επιλογή κατάταξης όταν ένα άτομο έχει πολύ μεγαλύτερη πιθανότητα επιλογής

Στο σχήμα που ακολουθεί φαίνεται η πιθανότητα επιλογής των ατόμων για C=0.5 και C=0.7 όταν όλα τα άτομα έχουν περίπου την ίδια πιθανότητα επιλογής.

Χρωμόσωμα	Fitness	Ranking	C=0.7	C=0.5
A	99.99	5	5.19%	1.56%
B	100.01	6	7.41%	3.13%
C	99.98	4	3.63%	0.78%
D	100.02	7	10.59%	6.26%
E	99.97	3	2.54%	0.39%
F	100.03	8	15.13%	12.51%
G	99.96	2	1.78%	0.20%
H	100.04	9	21.61%	25.02%
I	99.95	1	1.25%	0.10%
J	100.05	10	30.87%	50.05%

Εικόνα 50 Εκθετική επιλογή κατάταξης όταν όλα τα άτομα έχουν περίπου την ίδια πιθανότητα επιλογής

Πλεονεκτήματα:

- Αν εφαρμοστεί όταν τα ένα άτομο έχει πολύ μεγαλύτερη πιθανότητα επιλογής, αποτρέπει τον αλγόριθμο από το να κολλήσει σε ένα τοπικό ακρότατο, δίνοντας σε όλα τα άτομα την ευκαιρία να επιλεγούν αυξάνοντας έτσι το εύρος των λύσεων που θα αναζητήσει ο αλγόριθμος.
- Αν εφαρμοστεί όταν τα άτομα έχουν περίπου ίδια πιθανότητα επιλογής, δίνει στον Γ.Α. την ευκαιρία να επιλέξει τα καλύτερα άτομα από το να κάνει στην ουσία μια τυχαία επιλογή.

Μειονεκτήματα:

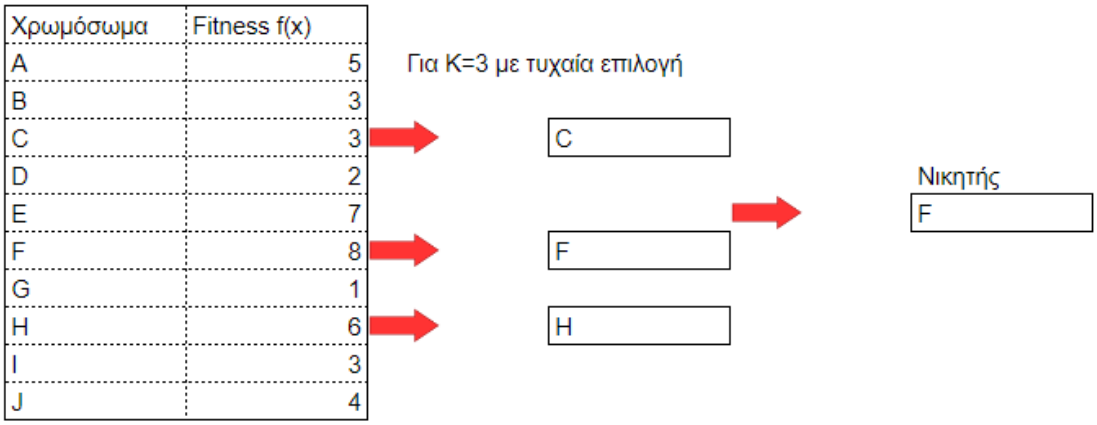
- Απαιτεί μεγαλύτερη υπολογιστική ισχύ καθώς σε κάθε γενιά πρέπει να ταξινομήσουμε τον πληθυσμό με βάση την απόδοσή του.

- Αν εφαρμοστεί όταν τα ένα άτομο έχει πολύ μεγαλύτερη πιθανότητα επιλογής έχουμε πιο αργή σύγκλιση καθώς τα καλύτερα άτομα δε θα επιλέγονται τόσο συχνά.
- Αν εφαρμοστεί όταν τα άτομα έχουν περίπου ίδια πιθανότητα επιλογής υπάρχει ο κίνδυνος να χάσουμε κάποιες καλές λύσεις καθώς τα άτομα με μικρή απόδοση δε θα επιλέγονται τόσο συχνά.

Η υλοποίηση του αλγορίθμου σε PHP με την χρήση της alias method βρίσκεται εδώ.

6.4 Επιλογή τουρνουά - Tournament Selection

Σε αυτή τη μέθοδο δημιουργούμε n, όπου n το μέγεθος του πληθυσμού, τουρνουά επιλέγοντας τυχαία K, όπου K σταθερός αριθμός, άτομα από τον πληθυσμό και επιλέγουμε τον νικητή, δηλαδή το άτομο με την καλύτερη απόδοση, του κάθε τουρνουά. Στο σχήμα που ακολουθεί φαίνεται ένα τουρνουά τριών ατόμων.



Εικόνα 51 Tournament selection για τουρνουά τριών ατόμων

Ο αλγόριθμος της επιλογής τουρνουά περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Με τη συνάρτηση αξιολόγησης υπολογίζουμε την απόδοση των ατόμων του πληθυσμού.

Βήμα 2ο: Επιλέγουμε τυχαία K άτομα από τον πληθυσμό.

Βήμα 3ο: Επιλέγουμε το άτομο με την μεγαλύτερη απόδοση από τα K και επαναλαμβάνουμε τη διαδικασία N φορές όπου N το μέγεθος του πληθυσμού.

Όπως οι περισσότεροι παράμετροι στους γενετικούς αλγορίθμους έτσι και η τιμή του K εξαρτάται από πολλούς παράγοντες και μόνο με δοκιμές μπορούμε να δούμε ποια είναι κατάλληλη για το πρόβλημά μας.

Αν το K είναι πολύ μικρό τότε αυξάνουμε το πεδίο των λύσεων αλλά μειώνουμε την ταχύτητα σύγκλισης. Για K=1 έχουμε μια εντελώς τυχαία επιλογή. Αν το K είναι πολύ μεγάλο τότε τα άτομα με μικρή απόδοση έχουν πολύ μικρή πιθανότητα να επιλεγούν επομένως ο γενετικός αλγόριθμος μπορεί να εγκλωβιστεί σε ένα τοπικό ακρότατο. Αν για παράδειγμα ο πληθυσμός είναι 40 και K=40, θα επιλέξουμε 40 φορές το ίδιο άτομο (στην περίπτωση που η επιλογή γίνεται χωρίς επανατοποθέτηση).

Το K συνήθως παίρνει τιμές από δύο (binary tournaments που είναι και τα πιο συνηθισμένα) μέχρι 20% του population size.

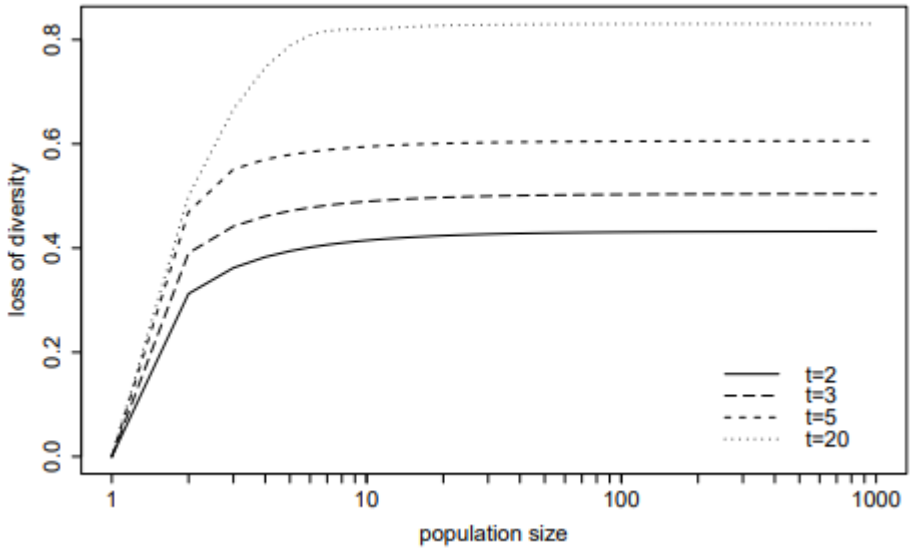
Η πιο απλή μορφή τουρνουά είναι αυτή της τυχαίας επιλογής των μελών του τουρνουά με επανατοποθέτηση.

Η μείωση της ποικιλομορφίας, δηλαδή το ποσοστό των ατόμων του αρχικού πληθυσμού που δεν υπάρχει καθόλου στο νέο μετά την επιλογή, από την τυχαία επιλογή τουρνουά με επανατοποθέτηση υπολογίστηκε από τον Motoki (T., 2002) και δίνεται από τον τύπο:

$$D(t,N) = \frac{1}{N}\sum_{k=1}^{N}\left(1 - \frac{k^t - (k-1)^t}{N^t}\right)^N$$

Εικόνα 52 Τύπος Motoki

Στον προηγούμενο τύπο N είναι το μέγεθος του πληθυσμού και t το μέγεθος του τουρνουά. Στο σχήμα που ακολουθεί βλέπουμε πως το μέγεθος του πληθυσμού και του τουρνουά επηρεάζουν την ποικιλομορφία. Για παράδειγμα για pop size = 10 και tournament size = 2 έχουμε απώλεια περίπου 40% του πληθυσμού. Για pop size = 10 και tournament size = 20 η απώλεια είναι 80% του πληθυσμού.



Εικόνα 53 Μείωση ποικιλομορφίας πληθυσμού

Σύμφωνα με τον Poli (Poli, 2005) δύο είναι οι παράγοντες που οδηγούν στη μείωση της ποικιλομορφίας όταν χρησιμοποιηθεί η τυχαία επιλογή τουρνουά με επανατοποθέτηση:

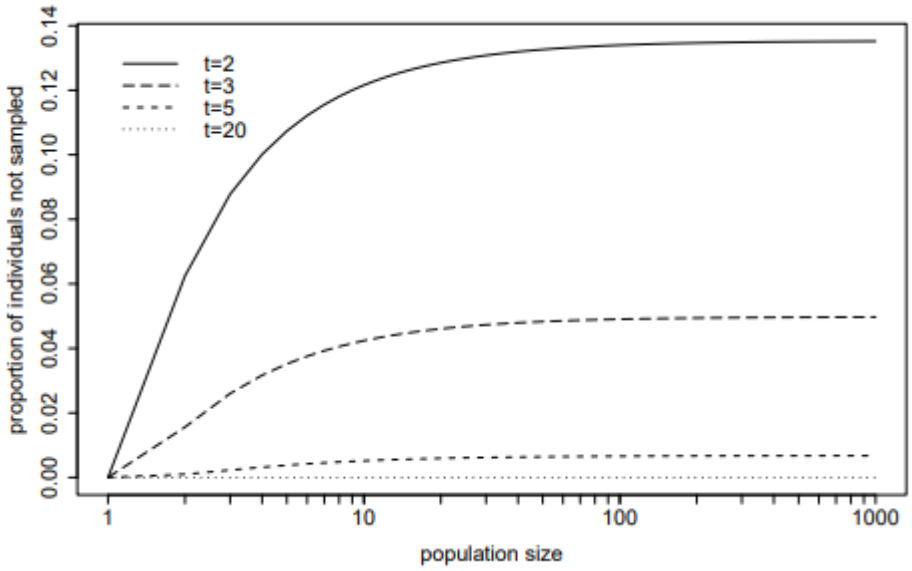
- Το να μην επιλεχθεί κάποιο άτομο σε κανένα τουρνουά (Not-Sampled Issue).
- Το να μην επιλεχθεί κάποιο άτομο επειδή δεν κέρδισε κανένα τουρνουά.

Το ποσοστό των ατόμων που δε θα επιλεγούν σε κανένα τουρνουά (Not-Sampled Issue) υπολογίστηκε από τον Poli (Poli, 2005) και δίνεται από τον τύπο:

$$NotSelected(N,t) = N\left(1 - \frac{1}{N}\right)^{tN}$$

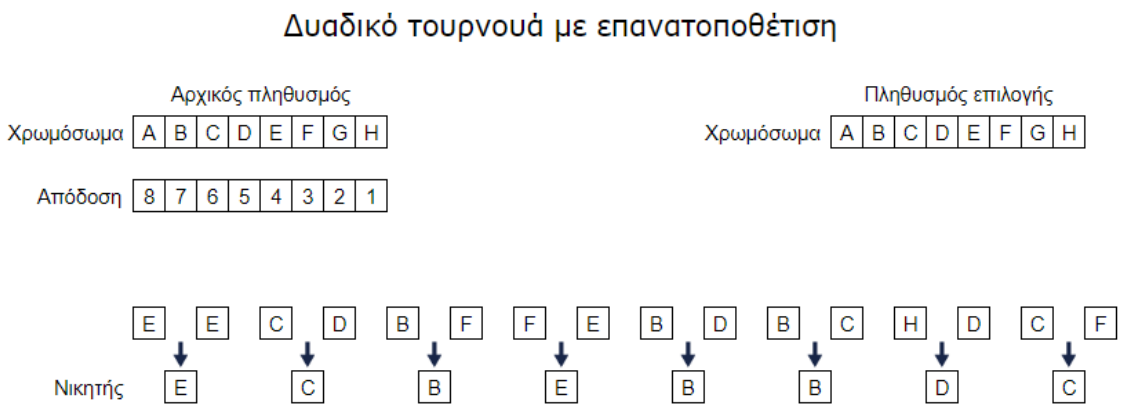
Εικόνα 54 Τύπος Poli

Στο παρακάτω σχήμα μπορούμε να δούμε ότι για t=2 και N=10 περίπου 13% του πληθυσμού δεν θα επιλεγεί σε κάποιο τουρνουά.



Εικόνα 55 Ποσοστό πληθυσμού που δε θα λάβει μέρος σε κανένα τουρνουά όταν έχουμε επανατοποθέτηση

Όπως φαίνεται και στο παράδειγμα του σχήματος που ακολουθεί, τα άτομα A και G δεν επιλέχθηκαν σε κανένα τουρνουά (παρόλο που το A έχει την καλύτερη απόδοση καθώς η επιλογή γίνεται τυχαία) και τα άτομα F και H δεν κέρδισαν κανένα τουρνουά (Το άτομο H που έχει την χειρότερη απόδοση μπορεί να κερδίσει μόνο αν επιλεγεί με τον εαυτό του).



Εικόνα 56 Παράδειγμα δυαδικού τουρνουά με επανατοποθέτηση

Το φαινόμενο του να επιλεγεί ένα άτομο περισσότερες από μια φορές αναλύθηκε από τους Xie et al. (Xie, An analysis of multi-sampled issue and no-replacement tournament selection, 2008) (το ονόμασαν multi-sampled issue) και απέδειξαν ότι η διαφορά στην απόδοση των δύο μεθόδων (με και χωρίς επανατοποθέτηση) είναι πολύ μικρή (όταν έχουμε φυσιολογικά μεγέθη πληθυσμού και τουρνουά) για να τη λάβουμε υπόψη σε σχέση με τους πόρους που χρειαζόμαστε για να την υλοποιήσουμε.

Στην επιλογή τουρνουά η πίεση της επιλογής (selection pressure) εξαρτάται από το μέγεθος του τουρνουά με αποτέλεσμα η χαμηλότερη τιμή που μπορεί να πάρει να είναι το 2 (Για k=1 δεν έχουμε τουρνουά αλλά τυχαία επιλογή). Οι Goldberg and Deb (Goldberg D. D., 1991) προσπάθησαν να λύσουν αυτό το πρόβλημα (δηλαδή να αυξήσουν την πίεση της επιλογής) εισάγοντας μια ακόμα μεταβλητή την p. Ο νικητής του τουρνουά έχει πιθανότητα p με p να ανήκει στο [0.5 , 1] ενώ ο χαμένος 1-p (ισχύει στην περίπτωση του binary tournament αλλά μπορεί να επεκταθεί). Οι Hingee and Hutter (Hingee, 2008)

απέδειξαν ότι κάθε επιλογή τουρνουά που χρησιμοποιεί πιθανότητα μπορεί να μετατραπεί σε επιλογή κατάταξης.

Για το ότι κάποια άτομα δε λαμβάνουν μέρος σε κανένα τουρνουά (Not-Sampled Issue) οι σημαντικότερες λύσεις που έχουν προταθεί είναι οι εξής.

6.4.1 Round-replacement tournament selection

Ο αλγόριθμος δημιουργήθηκε από τους Xie et al. (Xie, Is the not-sampled issue in tournament selection critical?, 2008) και περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Δημιουργούμε μια κενή λίστα L και θέτουμε i=0.

Βήμα 2ο: Επιλέγουμε K άτομα χωρίς επανατοποθέτηση από τον πληθυσμό P όπου K το μέγεθος του τουρνουά. Αν Psize < K δηλαδή αν ο πληθυσμός έχει πλέον λιγότερα μέλη από το μέγεθος του τουρνουά μεταφέρουμε τα μέλη της λίστας L στον πληθυσμό P.

Βήμα 3ο: Επιλέγουμε τον νικητή του τουρνουά

Βήμα 4ο: Οι χαμένοι του τουρνουά μεταφέρονται στην λίστα L.

Βήμα 5ο: Αυξάνουμε το i κατά 1. i=i+1. Αν i=N όπου N ο αριθμός των ατόμων που θέλουμε να πάρουμε από την επιλογή ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.

Με αυτό τον αλγόριθμο διασφαλίζουμε ότι κάθε άτομο θα λάβει μέρος ακριβώς K φορές σε ένα τουρνουά. Το άτομο με την καλύτερη απόδοση θα επιλεγεί K φορές (αφού θα κερδίσει και τα K τουρνουά, το άτομο με την χειρότερη απόδοση δε θα επιλεγεί (αφού θα χάσει όλα τα τουρνουά) και αυτό με τη μέση απόδοση θα επιλεγεί κατά μέσο όρο K/2 φορές. Οι Xie et al. (Xie, Is the not-sampled issue in tournament selection critical?, 2008) παρατήρησαν ότι αυτή η μέθοδος δίνει καλά αποτελέσματα μόνο στα binary tournaments.

6.4.2 Unbiased Tournament Selection

Ο αλγόριθμος δημιουργήθηκε από τους Sokolov and Whitley (Sokolov, 2005) και περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Δημιουργούμε μια μετάθεση M_K του αρχικού πληθυσμού P K-1 φορές όπου K το μέγεθος του τουρνουά κατά τέτοιο τρόπο ώστε $M_1[\iota] \neq M_2[\iota] \neq \ldots \neq M_{K-1}[\iota] \neq P[\iota]$

Βήμα 2ο: Για κάθε i επιλέγουμε το άτομο με την καλύτερη απόδοση.

Με αυτό τον αλγόριθμο διασφαλίζουμε ότι κάθε άτομο θα λάβει μέρος ακριβώς K φορές σε ένα τουρνουά. Το άτομο με την καλύτερη απόδοση θα επιλεγεί K φορές (αφού θα κερδίσει και τα K τουρνουά, το άτομο με την χειρότερη απόδοση δε θα επιλεγεί (αφού θα χάσει όλα τα τουρνουά) και αυτό με τη μέση απόδοση θα επιλεγεί κατά μέσο όρο K/2 φορές.

Το πρόβλημα είναι ότι επειδή τα τουρνουά δεν είναι ανεξάρτητα και όλος ο πληθυσμός από την διαδικασία της επιλογής προκύπτει σε ένα βήμα, δεν μπορούμε να χρησιμοποιήσουμε παράλληλο προγραμματισμό. Για αυτό το λόγο οι Sokolov and Whitley (Sokolov, 2005) πρότειναν μια εναλλακτική εκδοχή του unbiased tournament selection.

6.4.3 Parallel Unbiased Tournament Selection

Ο αλγόριθμος δημιουργήθηκε από τους Sokolov and Whitley (Sokolov, 2005) και περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Θέτουμε i=1.

Βήμα 2ο: Επιλέγουμε K-1άτομα από τον αρχικό πληθυσμό και το P[i].

Βήμα 3ο: Αυξάνουμε το i κατά 1 και επιστρέφουμε στο Βήμα 2.

Με αυτό τον τρόπο έχουμε δεν έχουμε πλέον το πρόβλημα ότι κάποιο άτομο δε θα λάβει μέρος σε κανένα τουρνουά, αφού θα το κάνει τουλάχιστον μία φορά και μπορούμε να χρησιμοποιήσουμε παράλληλο προγραμματισμό.

6.4.4 Σύγκριση μεθόδων

Στο ερώτημα ποια μέθοδος είναι καλύτερη είναι δύσκολο να απαντήσουμε. Το ότι μια μέθοδος είναι καλύτερη σε ένα συγκεκριμένο πρόβλημα με ένα συγκεκριμένο τελεστή διασταύρωσης και με ένα συγκεκριμένο τελεστής μετάλλαξης δε σημαίνει ότι είναι καλύτερη σε όλες. Μπορεί για παράδειγμα η τυχαία μέθοδος χωρίς επανατοποθέτηση να χάνει κάποια καλά χρωμοσώματα αλλά κερδίζει σε ποικιλομορφία που για κάποιες μεθόδους διασταύρωσης ίσως είναι πιο σημαντική. Αν ενδιαφερόμαστε περισσότερο για την ταχύτητα τότε σίγουρα η τυχαία μέθοδος επιλογής τουρνουά χωρίς επανατοποθέτηση, είναι η καλύτερη.

Στην έρευνά τους οι Xie et al. (Xie, An analysis of multi-sampled issue and no-replacement tournament selection, 2008) παρουσιάζουν αναλυτικά γραφήματα για το πως επηρεάζει το μέγεθος του πληθυσμού και του τουρνουά την ποικιλομορφία του πληθυσμού στις περιπτώσεις της τυχαίας επιλογής τουρνουά με και χωρίς επανατοποθέτηση.

Εμείς θα μελετήσουμε τα αποτελέσματα και των τεσσάρων μεθόδων που αναλύσαμε (random tournament selection χωρίς επανατοποθέτηση, random tournament selection με επανατοποθέτηση, unbiased tournament selection και parallel unbiased tournament selection) για population size 100, population size 25, tournament size 2 και tournament size 7, θα συγκρίνουμε τα αποτελέσματα μας με τις έρευνες των Xie et al. (Xie, An analysis of multi-sampled issue and no-replacement tournament selection, 2008) και Sokolov and Whitley (Sokolov, 2005), δίνοντας βάρος στο χρόνο που χρειάζεται η κάθε μέθοδος.

Για το πείραμα δημιουργήσαμε 10.000.000 άτομα δηλαδή 400.000 πληθυσμούς για population size 25 10 φορές, ενώ για population size 100 δημιουργήσαμε 1.000.000 άτομα δηλαδή 40.000 πληθυσμούς 10 φορές. Το άτομο 1 έχει το χειρότερο fitness value και το άτομο 100 το καλύτερο. Δε μας ενδιαφέρει η τιμή της απόδοσης καθώς δεν παίζει ρόλο στην επιλογή τουρνουά. Ένας πληθυσμός που τα 5 άτομα έχουν αποδόσεις 1, 2, 3, 4, 5 θα μας δώσει τα ίδια αποτελέσματα με έναν πληθυσμό που τα άτομα έχουν αποδόσεις 1, 2, 3, 4, 50. Τα αναλυτικά αποτελέσματα του πειράματος βρίσκονται εδώ και εδώ.

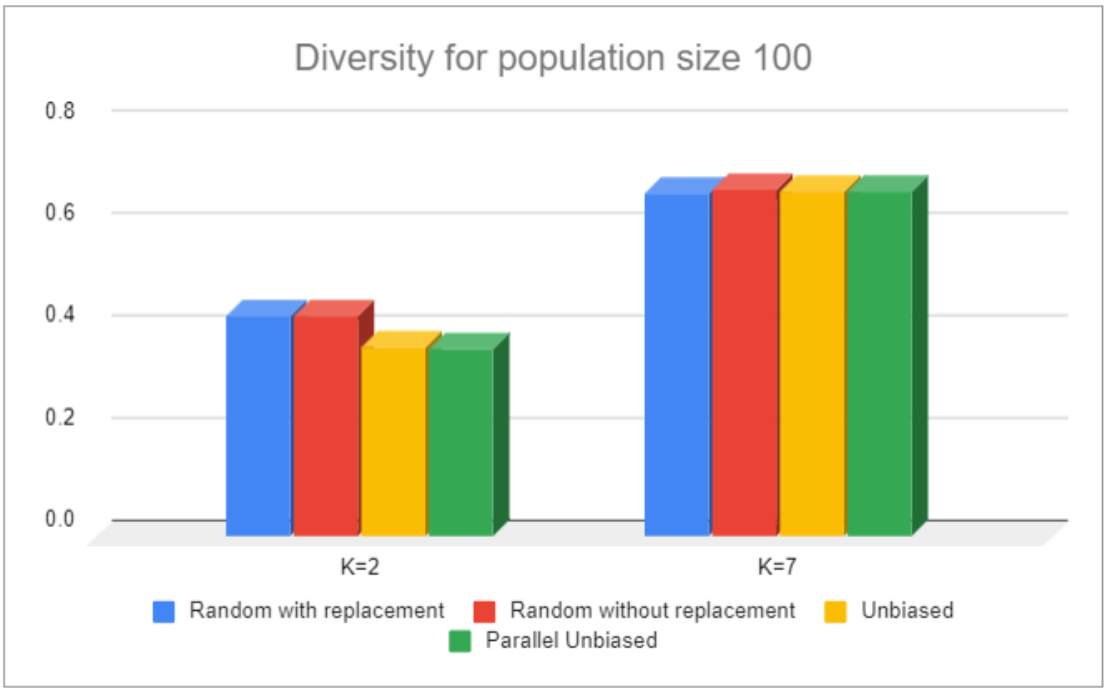
Παρατηρήσεις:

- Αν έχουμε population size 100 και tournament size 2 ανάμεσα στην επιλογή με και χωρίς επανατοποθέτηση δεν υπάρχει ουσιαστική διαφορά. Τα 36 άτομα με χειρότερη απόδοση έχουν μια μικρή ποσοστιαία αύξηση στην επιλογή τους, όταν έχουμε επανατοποθέτηση, της τάξεως του 0.01%. Τα άτομα με μεγάλη απόδοση επιλέγονται λιγότερο συχνά αλλά όχι τόσο ώστε να δικαιολογείται η αύξηση σε πόρους καθώς με επανατοποθέτηση χρειάστηκαν 0.7604949951 δευτερόλεπτα ενώ χωρίς 3.464523697 τι στιγμή που το άτομο με την καλύτερη απόδοση επιλέχθηκε κατά μέσο όρο 19938.1 φορές στην πρώτη περίπτωση και 19941.8 στη δεύτερη.
- Όταν δεν έχουμε επανατοποθέτηση τα K-1 άτομα με τη χειρότερη απόδοση δεν θα επιλεγούν ποτέ κάτι αναμενόμενο. Το άτομο με τη χειρότερη απόδοση μπορεί να κερδίσει ένα binary tournament μόνο όταν είναι αντιμέτωπο με τον εαυτό του. Το άτομο με την K-1 χειρότερη απόδοση πρέπει να είναι σε ένα τουρνουά με K-1 άτομα που έχουν χειρότερη απόδοση από αυτό για να επιλεγεί, πράγμα αδύνατον αφού δεν έχουμε επανατοποθέτηση.
- Όσο αυξάνεται το tournament size τόσο αυξάνεται και ο χρόνος εκτέλεσης. Αυτός είναι ένας επιπλέον λόγος που τα binary tournament είναι τα πιο διαδεδομένα και σχεδόν ποτέ δε χρησιμοποιούνται τουρνουά με tournament size > 7.
- Αν αυξήσουμε το tournament size αρχίζουν να παρατηρούνται διαφορές ανάμεσα στην μέθοδο random tournament με και χωρίς επανατοποθέτηση κάτι που συμφωνεί με τις παρατηρήσεις των Xie et al. (Xie, An analysis of multi-sampled issue and no-replacement

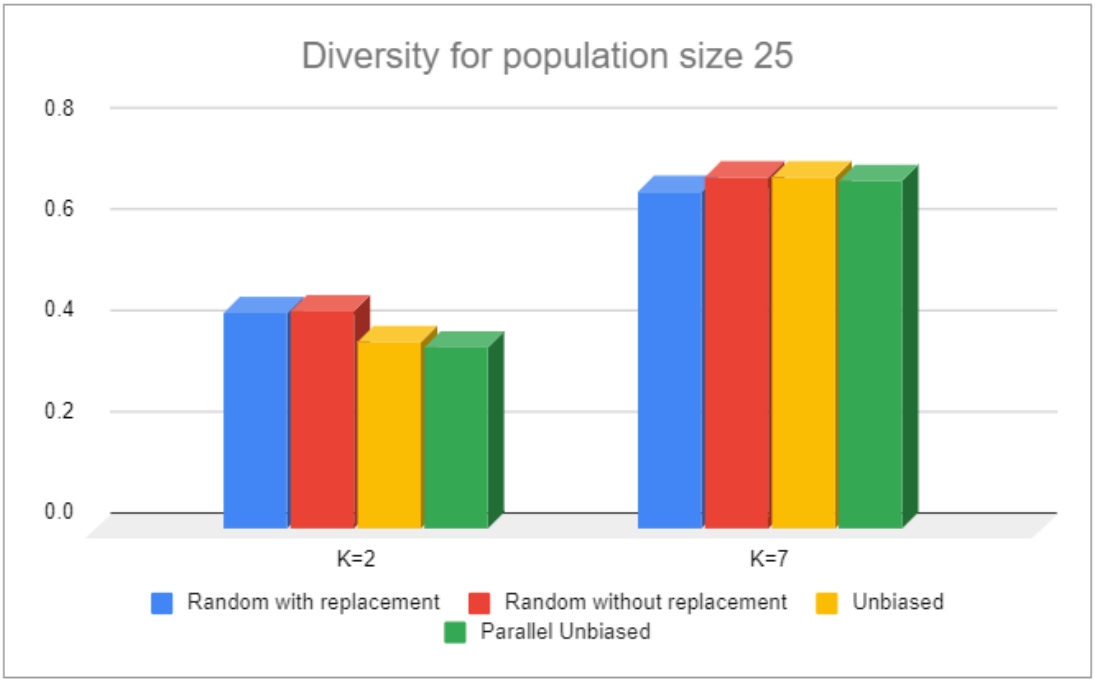
tournament selection, 2008). Για παράδειγμα για population size 100 και tournament size 7 το άτομο με την καλύτερη απόδοση επιλέγεται κατά μέσο όρο 67955.2 φορές ή ποσοστό 6.80% όταν έχουμε επανατοποθέτηση, ενώ χωρίς επανατοποθέτηση επιλέγεται 70164.5 φορές ή ποσοστό 7.02%. Προφανώς οι διαφορές είναι ακόμα μεγαλύτερες όταν το population size είναι 25 με τα αντίστοιχα ποσοστά να είναι 24.86% και 28% (Για ένα τόσο μικρό population size υπάρχουν διαφορές ακόμα και για tournament size = 2).

- Η μέθοδος unbiased tournament selection είναι πολύ χρονοβόρα και δεν μπορεί να χρησιμοποιηθεί παράλληλος προγραμματισμός. Για population size 100 και tournament size 2 η unbiased tournament selection χρειάζεται 4.254783368 δευτερόλεπτα ενώ η random tournament selection με επανατοποθέτηση μόλις 0.7604949951.
- Όπως παρατηρούμε η unbiased tournament selection δεν έχει σημαντικές διαφορές με τις άλλες μεθόδους κάτι που συμφωνεί και με την έρευνα των Yongsheng Fang, Jun Li (Yongsheng Fang, 2010) δίνοντας σχεδόν ίδια αποτελέσματα με τη μέθοδο random tournament χωρίς επανατοποθέτηση σε λιγότερο χρόνο.
- Η μέθοδος parallel unbiased tournament selection είναι πιο και γρήγορη από την μέθοδο unbiased tournament selection και από την μέθοδο random tournament selection χωρίς επανατοποθέτηση και μπορεί να χρησιμοποιηθεί παράλληλος προγραμματισμός.
- Αν το tournament size είναι μεγαλύτερο από 2 δεν υπάρχει διαφορά στην ποικιλομορφία του πληθυσμού . Για population size 100 και tournament size 7 οι 4 μέθοδοι δώσανε τις εξής απώλειες πληθυσμού 0.670102, 0.676686, 0.675231, 0.673136. Περίπου το 67% του πληθυσμού δεν επιλέγεται και στις 4 περιπτώσεις. Αντίθετα για tournament size 2 για τις μεθόδους random tournament selection with replacement, random tournament selection without replacement, unbiased tournament selection και parallel unbiased tournament selection έχουμε τις εξής απώλειες πληθυσμού 0.430485, 0.431723, 0.368362 0.366523 αντίστοιχα. Άρα υπάρχει διαφορά για K=2 κάτι που συμφωνεί με τις παρατηρήσεις των Xie et al. (Xie, Is the not-sampled issue in tournament selection critical?, 2008).

Στα δύο σχήματα που ακολουθούν βλέπουμε το ποσοστό του πληθυσμού που δεν επιλέγεται για δύο διαφορετικά μεγέθη πληθυσμού και δύο διαφορετικά μεγέθη τουρνουά και με τις 4 μεθόδους.



Εικόνα 57 Ποσοστό πληθυσμού που δεν επιλέγεται για pop-size 100



Εικόνα 58 Ποσοστό πληθυσμού που δεν επιλέγεται για pop-size 25

Πλεονεκτήματα:

- Μπορεί να χρησιμοποιηθεί παράλληλος προγραμματισμός. Κάθε τουρνουά είναι ανεξάρτητο οπότε μπορούμε να εκτελούμε κάθε τουρνουά παράλληλα με τα άλλα αυξάνοντας την ταχύτητα του αλγόριθμου.
- Δεν αντιμετωπίζει πρόβλημα αν κάποιο άτομο έχει πολύ μεγάλη απόδοση ή αν όλα έχουν περίπου την ίδια όταν το μέγεθος του τουρνουά είναι μικρό καθώς η αρχική επιλογή γίνεται τυχαία.
- Είναι γρήγορη (αφού δεν απαιτεί ταξινόμηση) και εύκολη στην υλοποίηση.
- Μπορεί να χρησιμοποιηθεί όταν η απόδοση παίρνει αρνητικές τιμές.
- Η πίεση επιλογής μπορεί εύκολα να τροποποιηθεί αυξάνοντας ή μειώνοντας το μέγεθος του τουρνουά.

Μειονεκτήματα:

- Για μεγάλο μέγεθος τουρνουά τα άτομα με χαμηλή απόδοση δε θα επιλέγονται καθόλου μειώνοντας έτσι το εύρος αναζήτησης λύσεων.
- Κάποια από τα άτομα του αρχικού πληθυσμού δε θα συμμετέχουν σε κανένα τουρνουά (όπως είδαμε όσο μικρότερο είναι το μέγεθος του τουρνουά και όσο μεγαλύτερο είναι το μέγεθος του πληθυσμού, τόσο αυξάνεται αυτό το ποσοστό και μπορεί να φτάσει μέχρι περίπου 14%).

Η υλοποίηση του random tournament selection with replacement σε php βρίσκεται εδώ.

Η υλοποίηση του random tournament selection without replacement σε php βρίσκεται εδώ.

Η υλοποίηση του unbiased tournament selection σε php βρίσκεται εδώ.

Η υλοποίηση του parallel unbiased tournament selection σε php βρίσκεται εδώ.

6.5 Επιλογή με αποκοπή - Truncation Selection - Top Mate Selection

Σε αυτή τη μέθοδο επιλογής αρχικά ταξινομούμε όλα τα άτομα με βάση την απόδοσή τους και επιλέγουμε τα N καλύτερα. Το N το έχουμε καθορίσει στην αρχή και συνήθως παίρνει τιμές ανάμεσα στο 10% και 50% του πληθυσμού.

Επειδή στους περισσότερους γενετικούς αλγορίθμους το μέγεθος του πληθυσμού παραμένει σταθερό από γενιά σε γενιά, αν έχουμε επιλογή με αποκοπή δημιουργούμε αντίγραφα των ατόμων που επιλέξαμε. Για παράδειγμα αν το μέγεθος του πληθυσμού είναι 100 και το N είναι 20% τότε θα ο Γ.Α. επιλέγει τα 20 άτομα με την καλύτερη απόδοση και θα παράγει 5 αντίγραφα από το καθένα.

Η μέθοδος αυτή έχει ένα σημαντικό μειονέκτημα. Η συνάρτηση κόστους μπορεί να μας δίνει την πληροφορία του πόσο καλό είναι ένα χρωμόσωμα, αλλά αυτό δε σημαίνει ότι δεν υπάρχουν καλά χαρακτηριστικά και στα χρωμοσώματα με χαμηλή απόδοση, ειδικά στις πρώτες γενιές του γενετικού αλγορίθμου. Αυτά τα καλά χαρακτηριστικά κινδυνεύουν να χαθούν για πάντα αν υπάρχουν.

Όποια μέθοδο επιλογής και να χρησιμοποιήσουμε θα έχουμε μείωση της ποικιλομορφίας του πληθυσμού, δηλαδή κάποια άτομα του αρχικού πληθυσμού δε θα υπάρχουν καθόλου στον νέο (συνήθως αυτά με μικρή απόδοση) και θα έχουν αντικατασταθεί από αντίγραφα ατόμων με καλύτερη απόδοση. Αυτός είναι άλλωστε και ο στόχος της επιλογής, να μας δώσει πληθυσμό με μεγαλύτερη κατά μέσο όρο απόδοση.

Το πρόβλημα είναι ότι στη μέθοδο με αποκοπή αυτό συμβαίνει σε ένα μεγάλο μέρος του πληθυσμού από την πρώτη γενιά, πριν δηλαδή ακόμα πάρουμε κάποιο καλό γενετικό υλικό που μπορεί να υπάρχει σε αυτά τα άτομα. Για αυτό το λόγο η επιλογή με αποκοπή δεν χρησιμοποιείται τόσο συχνά παρά μόνο όταν ο πληθυσμός είναι πολύ μεγάλος.

Υπάρχουν και παραλλαγές αυτής της μεθόδου, στις οποίες επιλέγουμε και ένα ποσοστό ατόμων με μέση και χαμηλή απόδοση.

Στο σχήμα που ακολουθεί φαίνεται η μέθοδος της επιλογής με αποκοπή, όταν επιλέγεται το 50% των ατόμων με την καλύτερη απόδοση.

Χρωμόσωμα	Fitness Value
A	4
B	7
C	12
D	8
E	5
F	1
G	10
H	12
I	-3
J	25
K	111
L	45

Ταξινόμηση

Χρωμόσωμα	Fitness Value
K	111
L	45
J	25
C	12
H	12
G	10
D	8
B	7
E	5
A	4
F	1
I	-3

Επιλογή 50%

Χρωμόσωμα	Fitness Value
K	111
L	45
J	25
C	12
H	12
G	10
K	111
L	45
J	25
C	12
H	12
G	10

Εικόνα 59 Επιλογή με αποκοπή

Πλεονεκτήματα:

- Η υλοποίηση είναι πολύ εύκολη. Το μόνο που πρέπει να κάνουμε είναι η ταξινόμηση του πληθυσμού με βάση την απόδοση.
- Ταχύτητα. Μετά την ταξινόμηση παίρνουμε ένα μέρος του αρχικού πληθυσμού χωρίς να χρειάζεται παραγωγή τυχαίων αριθμών ή αναζήτηση.
- Η απόδοση των ατόμων μπορεί να πάρει αρνητικές τιμές.

Μειονεκτήματα:

- Πρόωρη σύγκλιση. Το γενετικό υλικό του πληθυσμού με μικρή απόδοση, χάνεται από την αρχή και μπορεί να εμφανιστεί μόνο τυχαία μέσω της μετάλλαξης με αποτέλεσμα να υπάρχει μεγαλύτερη πιθανότητα ο γενετικός αλγόριθμος να συγκλίνει σε ένα τοπικό ακρότατο.

Η υλοποίηση του truncation selection σε php βρίσκεται εδώ.

6.6 Επιλογή Boltzmann - Boltzmann selection

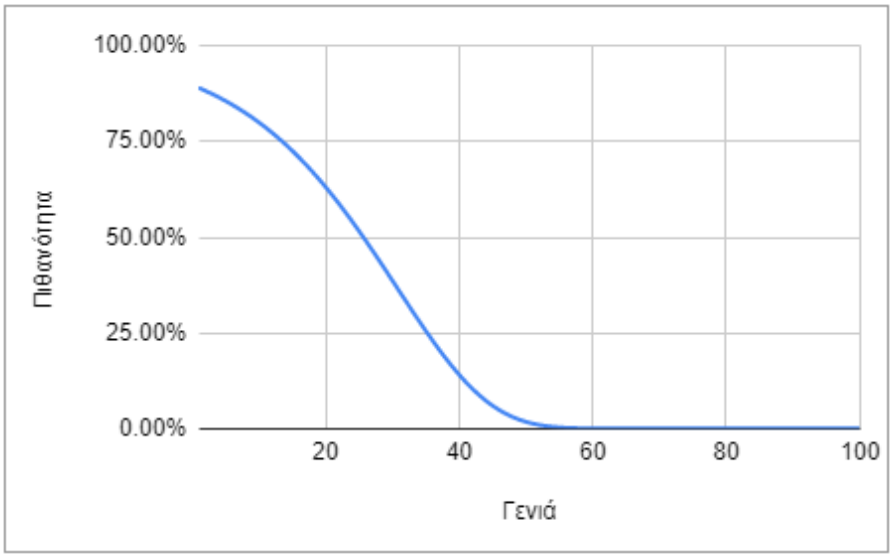
Η επιλογή Boltzmann δεν είναι ακριβώς μια διαφορετική διαδικασία επιλογής, αλλά ένας τρόπος να ελέγξουμε την πίεση της επιλογής. Μια συνεχώς μεταβαλλόμενη θερμοκρασία ελέγχει το ρυθμό επιλογής. Στις πρώτες γενιές η θερμοκρασία είναι υψηλή άρα η πίεση επιλογής είναι μικρή, με αποτέλεσμα τα άτομα με χαμηλή απόδοση να έχουν μεγαλύτερη πιθανότητα να επιλεγούν από ότι θα έπρεπε με βάση την απόδοσή τους. Αυτό επιτρέπει στον αλγόριθμο να κάνει αναζήτηση σε πολλές κατευθύνσεις. Όσο πλησιάζουμε προς το τέλος η θερμοκρασία μειώνεται, η πίεση επιλογής μεγαλώνει και τα άτομα με μεγαλύτερη απόδοση επιλέγονται πιο συχνά, ώστε ο αλγόριθμος να μπορέσει να συγκλίνει σε κάποιο ακρότατο.

Στην στατιστική μηχανική, η κατανομή Boltzmann (ή κατανομή Gibbs) μας δίνει την πιθανότητα ένα σύστημα να βρεθεί σε μία κατάσταση i. Η πιθανότητα δίνεται από τον τύπο $P_i = e^{-\varepsilon_i/kT}$, όπου ε_i η ενέργεια της κατάστασης, k η σταθερά Boltzmann και T η απόλυτη θερμοκρασία. Όσο μειώνεται η θερμοκρασία T τόσο μειώνεται και η πιθανότητα P_i.

Όπως αναφέραμε στην αρχή του κεφαλαίου η επιλογή Boltzmann είναι ένας τρόπος να δώσουμε στα άτομα με χαμηλή απόδοση, την ευκαιρία να επιλεγούν στις πρώτες γενιές του γενετικού αλγορίθμου. Ο τρόπος με τον οποίο θα εισάγουμε την κατανομή Boltzmann στον τελεστή επιλογής ενός γενετικού αλγορίθμου, αποτελεί αντικείμενο μελέτης. Στη συνέχεια του κεφαλαίου θα δούμε τρεις από αυτούς τους τρόπους.

- Συγκρίνουμε ένα άτομο A με το άτομο B που έχει την ως τώρα καλύτερη απόδοση. Αν η απόδοση του ατόμου A είναι μεγαλύτερη από την απόδοση του B τότε επιλέγεται το A. Αν όχι τότε το A επιλέγεται με πιθανότητα $P = e^{-(f(B)-f(A))/T}$, όπου $T = T_0(1-a)^k$, $k = 1 + 100 * g/G$, g η τρέχουσα γενιά, G ο μέγιστος αριθμός γενεών, a σταθερά που παίρνει τιμές στο διάστημα (0,1) και T_0 σταθερά που παίρνει τιμές στο διάστημα [5,100][(S.N.Deepa, 2008) σελ. 49]. Τον ίδιο τρόπο εξερεύνησης χρησιμοποιεί και ο μεταυρετικός αλγόριθμος simulated annealing. Στις πρώτες γενιές η θερμοκρασία είναι υψηλή, οπότε τα άτομα με χαμηλή απόδοση έχουν πολύ μεγάλη πιθανότητα να επιλεγούν. Για παράδειγμα αν υποθέσουμε ότι η μέγιστη απόδοση είναι f(B)=10, ένα άτομο A με απόδοση f(A)=0, έχει πιθανότητα $p \cong 90\%$ να επιλεγεί αν η αρχική θερμοκρασία T είναι 100. Αν η θερμοκρασία πέσει στο 10 η πιθανότητα να επιλεγεί είναι $p \cong 37\%$ ενώ αν η θερμοκρασία πέσει στο 1 η

πιθανότητα να επιλεγεί είναι $p \cong 0.0045\%$. Χωρίς τη χρήση ελιτισμού, με τη συγκεκριμένη μέθοδο υπάρχει η πιθανότητα το καλύτερο άτομο μιας γενιάς να μην επιλεγεί.



Εικόνα 60 Πιθανότητα επιλογής ενός ατόμου με f(A)=0, G=100, T0=100, a=0.07 και f(B)=10

- Δημιουργούμε τουρνουά και ο νικητής καθορίζεται με τη χρήση της κατανομής Boltzmann (Goldberg D. E., A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing, 1990). Τη μέθοδο αυτή τη συναντάμε στη βιβλιογραφία με το όνομα τουρνουά Boltzmann (Boltzmann tournament selection).
- Μετασχηματίζουμε τις αποδόσεις των ατόμων του πληθυσμού με τη χρήση της κατανομής Boltzmann και χρησιμοποιούμε την επιλογή ρουλέτες στις αποδόσεις που προέκυψαν (Jun LI, 2001). Οι αποδόσεις των ατόμων μετασχηματίζονται σύμφωνα με την σχέση g(X)=f(X)/T, όπου f(X) η απόδοση του ατόμου και T η θερμοκρασία. Η θερμοκρασία υπολογίζεται σύμφωνα με τον τύπο $T = \max\left[\left(1 - e^{-\left(1-\frac{k}{K+1}\right)*\lambda}\right) * T_0, \delta\right]$, όπου k η τρέχουσα γενιά, K ο μέγιστος αριθμός γενεών, λ μια σταθερά, T_0 η αρχική θερμοκρασία όπου συνήθως παίρνει την τιμή της μέγιστης απόδοσης του αρχικού πληθυσμού επί 10 και δ ένας μικρός θετικός αριθμός π χ 0.01.

Παράδειγμα. Έστω ότι η μέγιστη απόδοση του αρχικού πληθυσμού είναι 10 και ο αρχικός πληθυσμός αποτελείται από 4 άτομα με αποδόσεις f(A)=2, f(B)=3, f(C)=5 και f(D)=10. Αν η επιλογή γίνει με τη μέθοδο της ρουλέτας οι πιθανότητες επιλογής των ατόμων είναι: P(A)=10%, P(B)=15%, P(C)=25% και P(D)=50%.

Για λ=2.5 μετασχηματίζουμε τις αποδόσεις των ατόμων g(A)=1.022, g(B)=1.033,g(C)=1.056 , g(D)=1.115. Αν χρησιμοποιήσουμε την επιλογή ρουλέτας στις νέες αποδόσεις για k=1 οι πιθανότητες επιλογής των ατόμων είναι: P(A)=24.18%, P(B)=24.44%, P(C)=24.99% και P(D)=26.39%.

Στη γενιά 99 g(A)=1.513, g(B)=1.861, g(C)=2.816 και g(D)=7.928 οπότε με την επιλογή ρουλέτας οι πιθανότητες επιλογής των ατόμων είναι: P(A)=10.72%, P(B)=13.18%, P(C)=19.94% και P(D)=56.16%.

Έστω ότι κατά την εκτέλεση του γενετικού αλγορίθμου δημιουργήθηκε ένα άτομο E με απόδοση 23 το οποίο αντικατέστησε το άτομο B. Χωρίς μετασχηματισμό τα άτομα έχουν τις εξής πιθανότητες να επιλεγούν με την επιλογή ρουλέτας P(A)=5%, P(C)=12.5%, P(D)=25%, P(E)=57.5%.

Στη γενιά 2 g(A)=1.022, g(C)=1.056, g(D)=1.116 και g(E)=1.245, οπότε η πιθανότητα επιλογής με την επιλογή ρουλέτας είναι P(A)=23%, P(C)=23.79%, P(D)=25.14% και P(E)=28.07%.

Στη γενιά 99 g(A)=1.513, g(C)=2.816, g(D)=7.928 και g(E)=62.855, οπότε η πιθανότητα επιλογής με την επιλογή ρουλέτας είναι P(A)=2.01%, P(C)=3.75%, P(D)=10.55% και P(E)=83.69%.

Παρατηρούμε ότι στις πρώτες γενιές τα άτομα έχουν περίπου την ίδια πιθανότητα να επιλεγούν. Ακόμα και το άτομο Ε με απόδοση 23 δηλαδή 11.5 φορές μεγαλύτερη από την απόδοση του ατόμου Α έχει πιθανότητα επιλογής P(E)=28.07% όταν το άτομο Α έχει πιθανότητα P(A)=23%. Αυτό γίνεται για να κάνει ο γενετικός αλγόριθμος αναζήτηση σε περισσότερες κατευθύνσεις.

Όταν βρεθούμε μια γενιά πριν τον τερματισμό του αλγορίθμου, το άτομο Ε που έχει μεγαλύτερη απόδοση από την μέγιστη απόδοση του αρχικού πληθυσμού, ευνοείται και αποκτά πιθανότητα επιλογής 83.69%.

6.7 Επιλογή σταθερής κατάστασης - Steady state selection

Η "επιλογή" σταθερής κατάστασης (Darrell Whitley, 1988) (Syswerda G. , 1989) δεν είναι μια μέθοδος επιλογής ατόμων, αλλά μια διαφορετική μορφή γενετικού αλγορίθμου στην οποία δεν υπάρχουν γενιές. Ο αλγόριθμος περιγράφεται αναλυτικά από τα παρακάτω βήματα:

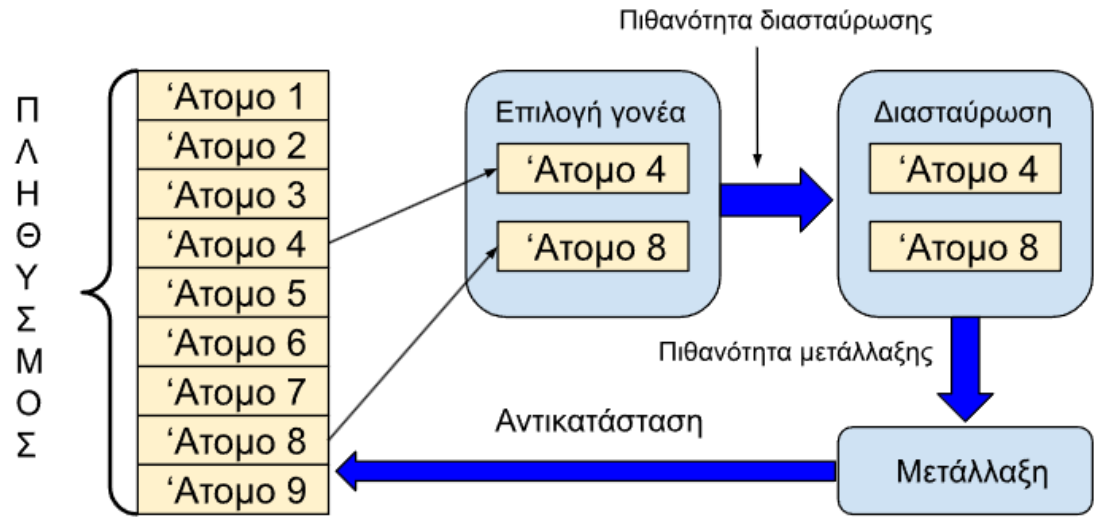
Βήμα 1ο: Επιλέγονται δύο γονείς με βάση την απόδοσή τους με κάποιον από τους γνωστούς τρόπους επιλογής. Προφανώς η επιλογή δεν μπορεί να γίνει με τη μέθοδο της πιθανολογικής καθολικής δειγματοληψίας.

Βήμα 2ο: Τα δύο άτομα που επιλέξαμε στο Βήμα 1 διασταυρώνονται. Το πόσο συχνά θα γίνει η διασταύρωση εξαρτάται από την πιθανότητα διασταύρωσης.

Βήμα 3ο: Τα δύο (ή το ένα ανάλογα με τον τελεστή διασταύρωσης που χρησιμοποιήσαμε), άτομα που προέκυψαν από το Βήμα 2 μεταλλάσσονται. Το πόσο συχνά θα γίνει η μετάλλαξη εξαρτάται από την πιθανότητα μετάλλαξης.

Βήμα 4ο: Οι δύο απόγονοι (ή ο απόγονος) που προέκυψαν από το Βήμα 3 αντικαθιστούν δύο άτομα από τον αρχικό πληθυσμό. Η αντικατάσταση στην αυθεντική έκδοση του αλγορίθμου γίνεται με τη μέθοδο κατάταξης, αλλά μπορεί να γίνει με οποιονδήποτε από τους γνωστούς τρόπους επιλογής (Luke, 2016).

Σημείωση: Αν στο Βήμα 4 επιλέξουμε για αντικατάσταση τα χρωμοσώματα με την χειρότερη απόδοση τότε ο Γ.Α. θα συγκλίνει πρόωρα. Η μέθοδος αυτή ονομάζεται GENITOR (Darrell Whitley, 1988).



Εικόνα 61 Steady state selection

6.7.1 Επιλογή επιζώντων - Survivor selection

Η διαδικασία της επιλογής σε έναν γενετικό αλγόριθμο μπορεί να εφαρμοστεί σε δύο σημεία. Το πρώτο σημείο είναι στην αρχή κάθε γενιάς όπου με την επιλογή δημιουργούμε έναν νέο προσωρινό πληθυσμό (επιλογή γονέα - parent selection). Το δεύτερο σημείο είναι στο τέλος της γενιάς όπου από τους

απογόνους και τους γονείς, επιλέγονται τα καταλληλότερα άτομα για να δημιουργήσουν το νέο πληθυσμό (επιλογή επιζώντων - survivor selection). Η επιλογή επιζώντων αποτελεί γενίκευση της επιλογής σταθερής κατάστασης.

Η επιλογή των ατόμων που θα σχηματίσουν τον νέο πληθυσμό μπορεί να γίνει με οποιαδήποτε από τις γνωστές τεχνικές επιλογής που χρησιμοποιούμε στην επιλογή γονέα.

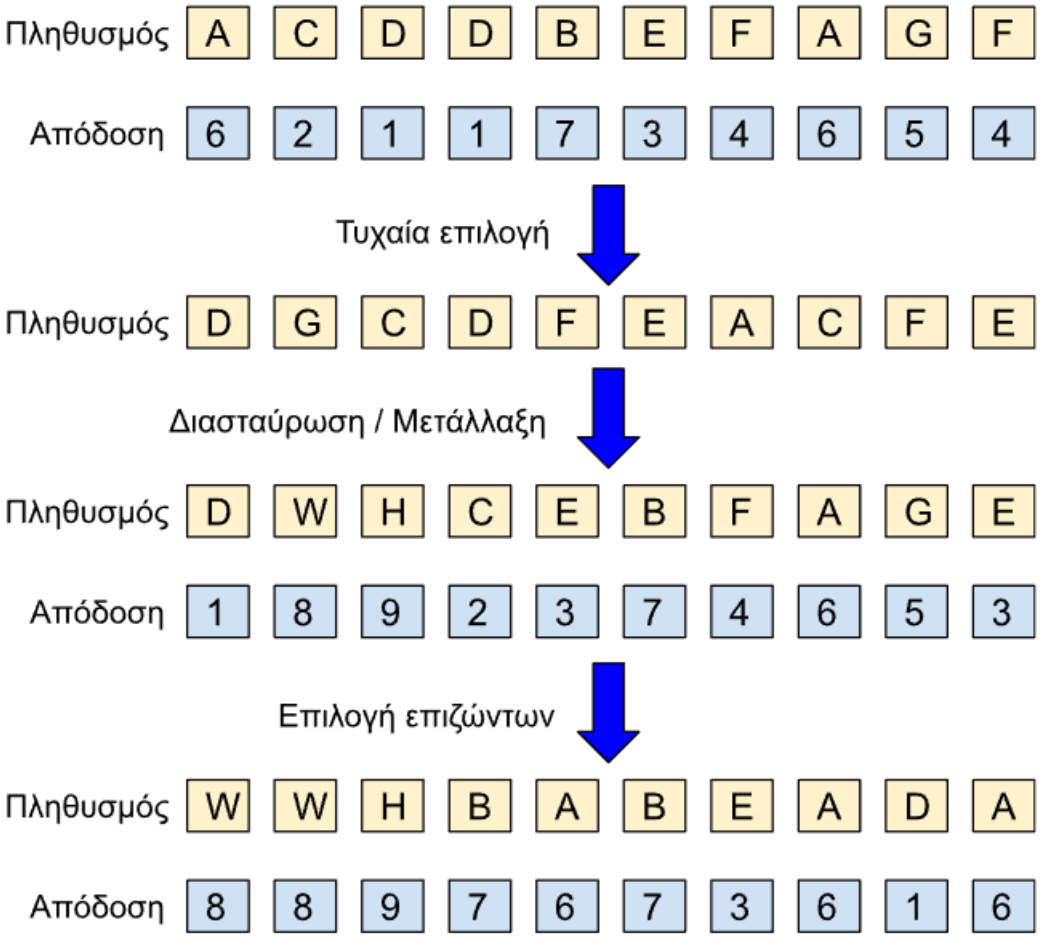
Εικόνα 62 Fitness based. Τα άτομα με τη χειρότερη απόδοση αντικαθίστανται(GENITOR)

Μια διαφορετική τεχνική είναι αγνοήσουμε τελείως την απόδοση των ατόμων κατά την επιλογή επιζώντων και η επιλογή να γίνει με βάση την "ηλικία" των ατόμων (age based).

Εικόνα 63 Age based

6.8 Τυχαία επιλογή - Uniform / Random Selection

Με αυτό τον τρόπο επιλογής όλα τα άτομα έχουν την ίδια πιθανότητα P=1/N , όπου N ο αριθμός των ατόμων του πληθυσμού, να επιλεγούν κατά την επιλογή του γονέα. Δεν υπάρχει η πίεση της επιλογής επομένως ο γενετικός αλγόριθμος μετατρέπεται σε έναν αλγόριθμο τυχαίας αναζήτησης αν δεν υπάρχει η επιλογή επιζώντων.



Εικόνα 64 Παράδειγμα τυχαίας επιλογής

6.9 Μετασχηματισμός απόδοσης

Ένας καλός γενετικός αλγόριθμος πρέπει στα πρώτα του στάδια να διατηρεί μια ποικιλία ατόμων, ώστε να κάνει αναζήτηση λύσεων σε διαφορετικές κατευθύνσεις. Πολλές φορές όμως όταν υπάρχει μεγάλη διαφορά στην απόδοση των ατόμων, τα άτομα με μικρή απόδοση δεν επιλέγονται καμιά φορά, με αποτέλεσμα ο αλγόριθμος να γεμίζει με αντίγραφα των ατόμων με καλή απόδοση και να εγκλωβίζεται σε ένα τοπικό ακρότατο. Για να δώσουμε την ευκαιρία σε κάποια από τα άτομα με χαμηλή απόδοση να περάσουν τα καλά τους γονίδια στις επόμενες γενιές, μερικές φορές πρέπει να γίνει ένας μετασχηματισμός στη συνάρτηση καταλληλότητας.

Υπάρχουν πολλοί τρόποι με τους οποίους μπορεί να γίνει ο μετασχηματισμός. Η επιλογή κατάταξης που αναλύσαμε σε προηγούμενη παράγραφο αποτελεί μια μορφή μετασχηματισμού. Μερικές ακόμα γνωστές μορφές μετασχηματισμού είναι οι εξής.

- **Γραμμική κλιμάκωση - Linear scaling**. Η νέα απόδοση δίνεται από τον τύπο $F_{new} = a * F_{old} + b$ με a, b σταθερές, οι οποίες υπολογίζονται πειραματικά ανάλογα με το πρόβλημα.
- **Δυναμική κλιμάκωση - Power scaling.** Η νέα απόδοση δίνεται από τον τύπο$F_{new} = F_{old}^{C}$, με C σταθερά, η οποία υπολογίζεται πειραματικά ανάλογα με το πρόβλημα.
- **Κλιμάκωση σίγμα - Sigma scaling.** Για την κλιμάκωση σίγμα υπάρχουν δύο προτάσεις. Η πρώτη πρόταση είναι από τον Forrest, S. (1985) (Forrest, 1985). Η νέα απόδοση δίνεται από

τον τύπο $F_{new} = F_{old} - (F_{avg} - c * \sigma)$ με c σταθερά, η οποία συνήθως παίρνει τιμές από 1 έως 3 και σ η τυπική απόκλιση. Η δεύτερη πρόταση έγινε από τον Tanese, R. (1989) (Tanese, 1989). Η νέα απόδοση δίνεται από τον τύπο $F_{new} = 1 + \frac{F_{old} - F_{avg}}{2\sigma}$, όπου σ η τυπική απόκλιση.

Σημείωση: Ο μετασχηματισμός μπορεί να χρησιμοποιηθεί και στην περίπτωση κατά την οποία η συνάρτηση καταλληλότητας επιστρέφει αρνητικές τιμές και θέλουμε να χρησιμοποιήσουμε μια μέθοδο επιλογής η οποία λειτουργεί μόνο με θετικές τιμές.

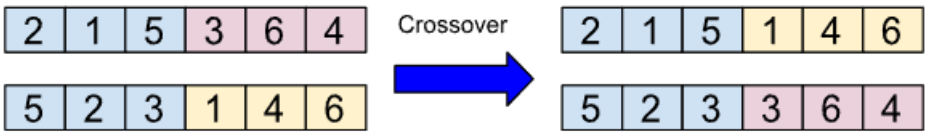
7 Διασταύρωση - Ανασυνδυασμός - Crossover – Recombination

Ανασυνδυασμός (recombination) ονομάζεται η διαδικασία κατά την οποία 2 γονείς (parents) συνδυάζονται για να παράγουν απογόνους (offsprings). Ο ορισμός αυτός αφορά την κλασική μορφή ενός γενετικού αλγορίθμου όπως διατυπώθηκε από τον Holland (Holland, 1975). Τον ανασυνδυασμό τις περισσότερες φορές θα τον δούμε να αναφέρεται ως διασταύρωση, γιατί στην κλασική μορφή ενός γενετικού αλγορίθμου οι δύο γονείς διασταυρώνονται σε κάποια σημεία για να παράγουν δύο απογόνους.

Στην πορεία δημιουργήθηκαν παρά πολλές μορφές ανασυνδυασμού. Μπορεί στη φύση να συνδυάζονται δύο άτομα για να παράγουν απογόνους, αλλά αυτό δεν συμβαίνει πάντα στους γενετικούς αλγορίθμους (A. E. Eiben Vrije, 1994) (Ting, 2005). Η πλειοψηφία των τελεστών διασταύρωσης είναι στοχαστικοί, αλλά υπάρχουν εξαιρέσεις, όπως για παράδειγμα η διασταύρωση sequential constructive (Ahmed Z. H., 2010) για το πρόβλημα του πλανόδιου πωλητή. Ο αριθμός των απογόνων που παράγεται από κάθε διασταύρωση είναι συνήθως δύο ή ένα.

Θα δώσουμε ένα πιο γενικό ορισμό για τον ανασυνδυασμό όπως αυτός εμφανίζεται σε πιο εξελιγμένους γενετικούς αλγορίθμους. Ανασυνδυασμός είναι η διαδικασία της ανταλλαγής γενετικού υλικού (γονιδίων) μεταξύ δύο ή και περισσότερων υποψηφίων λύσεων (χρωμοσωμάτων) που έχει ως στόχο τη δημιουργία μιας ή και περισσότερων καλύτερων λύσεων.

Η επιλογή του κατάλληλου τελεστή διασταύρωσης, είναι το σημαντικότερο κομμάτι στην επιτυχία ενός γενετικού αλγορίθμου. Η επιλογή εξαρτάται από το πρόβλημα και τον τρόπο με τον οποίο γίνεται η αναπαράσταση των λύσεων. Για παράδειγμα η διασταύρωση ενός σημείου δεν μπορεί να εφαρμοστεί στην κωδικοποίηση μετάθεσης, γιατί θα προκύψουν μη έγκυρες λύσεις. Τα τελευταία χρόνια έχουν γίνει υλοποιήσεις Γ.Α. οι οποίες χρησιμοποιούν περισσότερους από έναν αλγορίθμους διασταύρωσης (Ahmad B. A. Hassanat, 2017) (Stjepan Picek D. J., 2013). Κάθε φορά είτε επιλέγεται τυχαία ένας από τους διαθέσιμους αλγορίθμους διασταύρωσης, είτε χρησιμοποιούνται όλοι και κρατάμε τους καλύτερους απογόνους.

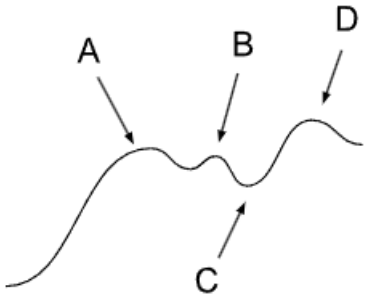


Εικόνα 65 Path representation και διασταύρωση ενός σημείου μας δίνουν μη έγκυρες λύσεις

Διασταύρωση δεν γίνεται σε όλα τα άτομα που προέκυψαν από την επιλογή, αλλά κάθε άτομο έχει μια πιθανότητα να διασταυρωθεί η οποία ονομάζεται πιθανότητα διασταύρωσης (crossover rate). Η κατάλληλη πιθανότητα διασταύρωσης δεν μπορεί να υπολογιστεί από κάποιο μαθηματικό τύπο καθώς εξαρτάται από πάρα πολλούς παράγοντες όπως, τη φύση του προβλήματος, την κωδικοποίηση που χρησιμοποιήθηκε, το μέγεθος του αρχικού πληθυσμού και τον τρόπο με τον οποίο γίνεται η διασταύρωση. Οι περισσότερες έρευνες προτείνουν μια πιθανότητα γύρω στο 70%, αλλά αυτός ο αριθμός αφορά την κλασική μορφή των γενετικών αλγορίθμων (Dr. Anantkumar J. Umbarkar, 2015) (D.D.(2015, 2015). Κάποιοι ερευνητές προτείνουν η πιθανότητα διασταύρωσης (όπως και η πιθανότητα μετάλλαξης και το μέγεθος του πληθυσμού), να αλλάζουν κατά τη διάρκεια εκτέλεσης του γενετικού

αλγορίθμου (S.N.Deepa, 2008) (Janikow). Ο μοναδικός τρόπος να βρούμε την "ιδανική" πιθανότητα διασταύρωσης για το γενετικό αλγόριθμο που υλοποιήσαμε, είναι να πειραματιστούμε με διάφορες τιμές.

Στόχος της διασταύρωσης είναι να παράγει απογόνους με καλύτερη απόδοση από τους γονείς, συνδυάζοντας τα καλά γονίδια των γονέων. Στην πράξη μπορεί να προκύψουν απόγονοι με χειρότερη απόδοση από τους γονείς, χωρίς αυτό να υποδηλώνει την κακή λειτουργία του γενετικού αλγορίθμου. Τα άτομα αυτά θα έχουν μικρή πιθανότητα επιλογής οπότε δε θα παράγουν αντίγραφα του εαυτού τους και κάποιο από αυτά ίσως μας οδηγήσει σε μια καλύτερη λύση. Για παράδειγμα μπορεί να μην υπάρχει συνδυασμός διασταύρωσης που να μας δίνει τη λύση D. Αν όμως οι λύσεις A και B διασταυρωθούν και μας δώσουν τη λύση C, αυτή μπορεί σε συνδυασμό με κάποια άλλη λύση, να μας δώσει τη λύση D.

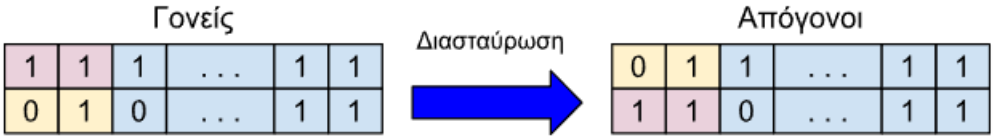


Εικόνα 66 Η διασταύρωση μπορεί να οδηγήσει σε χειρότερους απογόνους

Υπάρχουν πάρα πολλοί αλγόριθμοι διασταύρωσης. Επειδή στην αρχή οι γενετικοί αλγόριθμοι χρησιμοποιούσαν κυρίως τη δυαδική κωδικοποίηση οι περισσότεροι αλγόριθμοι θα πρέπει να τροποποιηθούν αν χρησιμοποιήσουμε άλλη μορφή κωδικοποίησης. Για την κωδικοποίηση μετάθεσης επειδή δεν παίζει ρόλο το γονίδιο, αλλά η σειρά με την οποία τα γονίδια βρίσκονται στο χρωμόσωμα έχουν δημιουργηθεί ειδικοί αλγόριθμοι διασταύρωσης.

7.1 Διασταύρωση ενός σημείου - One/Single point crossover

Πρόκειται για την πιο απλή μορφή διασταύρωσης και παρουσιάστηκε για πρώτη φορά από τον John Holland (1975) (Holland, 1975). Στη διασταύρωση ενός σημείου δύο γονείς παράγουν δύο απογόνους. Οι δύο γονείς χωρίζονται τυχαία σε δύο μέρη. Στη συνέχεια ενώνουμε το πρώτο μέρος του πρώτου γονέα με το δεύτερο μέρος του δεύτερου γονέα και το πρώτο μέρος του δεύτερου γονέα με το δεύτερο μέρος του πρώτου γονέα για να παράγουμε τους δύο απογόνους.



Εικόνα 67 Διασταύρωση ενός σημείου

Το βασικό μειονέκτημα αυτής της μεθόδου είναι ότι δημιουργεί μια συσχέτιση μεταξύ γειτονικών γονιδίων καθώς όλα τα γονίδια από ή μέχρι το σημείο διασταύρωσης μεταφέρονται σε κάποιον απόγονο. Αν ο γονέας έχει τα καλά του γονίδια στην αρχή και στο τέλος, η πιθανότητα να συνυπάρχουν στον ίδιο απόγονο είναι πολύ μικρή (Το φαινόμενο αυτό ονομάζεται positional bias (Eshelman L. C., 1989). Για αυτό το λόγο, όπως θα δούμε στη συνέχεια, δημιουργήθηκαν αλγόριθμοι διασταύρωσης 2 ή και K σημείων που περιορίζουν το πρόβλημα αυτό.

Η διασταύρωση ενός σημείου δημιουργήθηκε για την δυαδική κωδικοποίηση αλλά με ή χωρίς κάποιες τροποποιήσεις μπορεί να χρησιμοποιηθεί για όλες τις μορφές κωδικοποίησης. Αν χρησιμοποιηθεί για την κωδικοποίηση μετάθεσης πρέπει να δημιουργηθεί ένας διορθωτικός

αλγόριθμος, ώστε τα χρωμοσώματα που παράγονται να περιέχουν τα γονίδια των γονέων ακριβώς μια φορά.

Ο αλγόριθμος διασταύρωσης ενός σημείου αν χρησιμοποιηθεί σε γενετικούς αλγορίθμους που χρησιμοποιούν την κωδικοποίηση μετάθεσης ονομάζεται modified crossover (Davis L. , Applying Adaptive Algorithms toEpistatic Domains, 1985). Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Παράγουμε ένα τυχαίο αριθμό r στο διάστημα [1,N-1] όπου N ο αριθμός των γονιδίων.

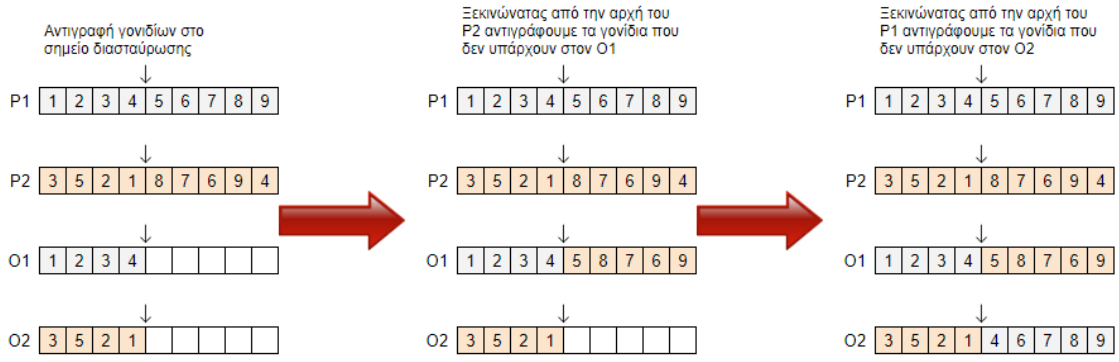
Βήμα 2ο: Αντιγράφουμε τα γονίδια που βρίσκονται στις θέσεις 1 έως r του πρώτου γονέα στον πρώτο απόγονο και αυτά του δεύτερου γονέα στον δεύτερο απόγονο.

Βήμα 3ο: Ξεκινώντας από την αρχή του δεύτερου γονέα αντιγράφουμε με τη σειρά τα γονίδια που δεν υπάρχουν ήδη στον πρώτο απόγονο στις θέσεις r+1 έως N.

Βήμα 4ο: Ξεκινώντας από την αρχή του πρώτου γονέα αντιγράφουμε με τη σειρά τα γονίδια που δεν υπάρχουν ήδη στον δεύτερο απόγονο στις θέσεις r+1 έως N.

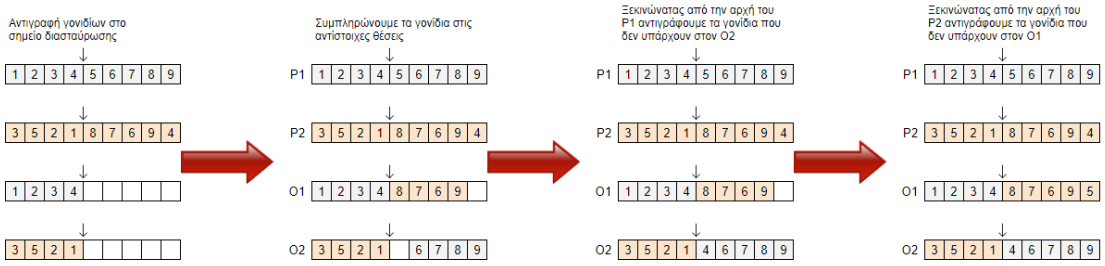
Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [1, 2, 3, 4, 5, 6, 7, 8, 9] και P2 = [3, 5, 2, 1, 8, 7, 6, 9, 4] με σημείο διασταύρωσης το 4. Τα γονίδια 1 , 2 , 3 και 4 του πρώτου γονέα τοποθετούνται στον πρώτο απόγονο O1 = [1 , 2 , 3 , 4 , _ , _ , _ , _ , _] και τα γονίδια 3 , 5 , 2 , 1 του δεύτερου γονέα στον δεύτερο απόγονο O2 = [3 , 5 , 2 , 1 , _ , _ , _ , _ , _].

Ξεκινάμε από την αρχή του δεύτερου γονέα P2. Το γονίδιο 3 υπάρχει ήδη στον O1. Πηγαίνουμε στον επόμενο γονίδιο του P2 που είναι το 5. Το 5 δεν υπάρχει στον O1 οπότε το τοποθετούμε O1 = [1 , 2 , 3 , 4 , 5 , _ , _ , _ , _]. Το επόμενο γονίδιο του P2 που δεν υπάρχει στον O1 είναι το 8 O1 = [1 , 2 , 3 , 4 , 5 , 8 , _ , _ , _]. Συνεχίζοντας και κάνοντας την ίδια διαδικασία και για τον δεύτερο απόγονο O2 παίρνουμε τους δύο απογόνους O1 = [1 , 2 , 3 , 4 , 5 , 8 , 7 , 6 , 9] και O2 = [3 , 5 , 2 , 1 , 4 , 6 , 7 , 8 , 9].



Εικόνα 68 Διασταύρωση ενός σημείου για το TSP

Σημείωση: Κάποιοι ερευνητές προτείνουν έναν διαφορετικό τρόπο διόρθωσης. Πρώτα συμπληρώνονται τα γονίδια τον απογόνων από τους γονείς στις αντίστοιχες θέσεις. Το αποτέλεσμα αυτής της υλοποίησης φαίνεται στο σχήμα που ακολουθεί.



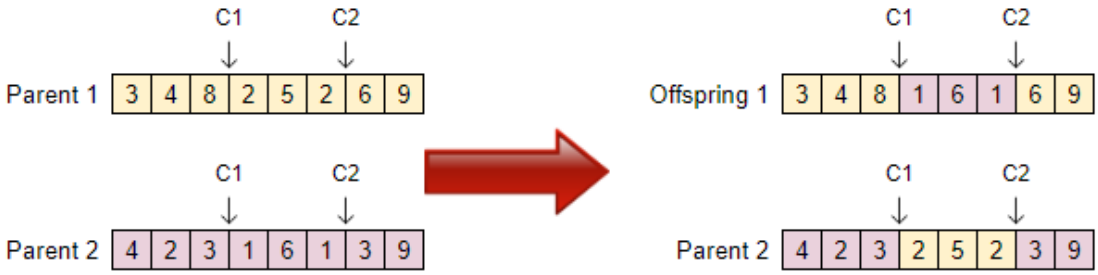
Εικόνα 69 Διασταύρωση ενός σημείου για το TSP με τοποθέτηση γονιδίων στις αντίστοιχες θέσεις

Η υλοποίηση του αλγορίθμου σε php για την κωδικοποίηση μετάθεσης βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου σε php για την δυαδική κωδικοποίηση βρίσκεται εδώ.

7.2 Διασταύρωση δύο σημείων - Two point crossover - Order Crossover (OX2)

Ο αλγόριθμος Order crossover προτάθηκε από τον Lawrence Davis (Davis L. , 1991). Όπως και στη διασταύρωση ενός σημείου δύο γονείς παράγουν δύο απογόνους. Για να γίνει η διασταύρωση επιλέγονται τυχαία δύο σημεία C1 και C2. Τα γονίδια που βρίσκονται αριστερά του C1 και δεξιά του C2 στον γονέα ένα αντιγράφονται στον απόγονο ένα. Αντίστοιχα τα γονίδια που βρίσκονται αριστερά του C1 και δεξιά του C2 στον γονέα δύο αντιγράφονται στον απόγονο δύο. Τα γονίδια που βρίσκονται ανάμεσα στα σημεία C1 και C2 του γονέα ένα αντιγράφονται στο απόγονο δύο και τα γονίδια που βρίσκονται ανάμεσα στα σημεία C1 και C2 του γονέα δύο αντιγράφονται στο απόγονο ένα.



Εικόνα 70 2 point crossover για κωδικοποίηση ακεραίων

Όπως και στην περίπτωση της διασταύρωσης ενός σημείου, αν χρησιμοποιήσουμε την διασταύρωση δύο σημείων για το TSP πρέπει να ελέγχουμε αν το γονίδιο υπάρχει ήδη στον απόγονο.

Ο αλγόριθμος διασταύρωσης δύο σημείων αν χρησιμοποιηθεί η κωδικοποίηση μετάθεσης περιγράφεται στα παρακάτω βήματα.

Βήμα 1: Παράγουμε 2 τυχαίους αριθμούς C1 και C2 με C1 να ανήκει στο [1,N-1], C2 να ανήκει στο [1,N] και C1<C2 όπου N ο αριθμός των γονιδίων.

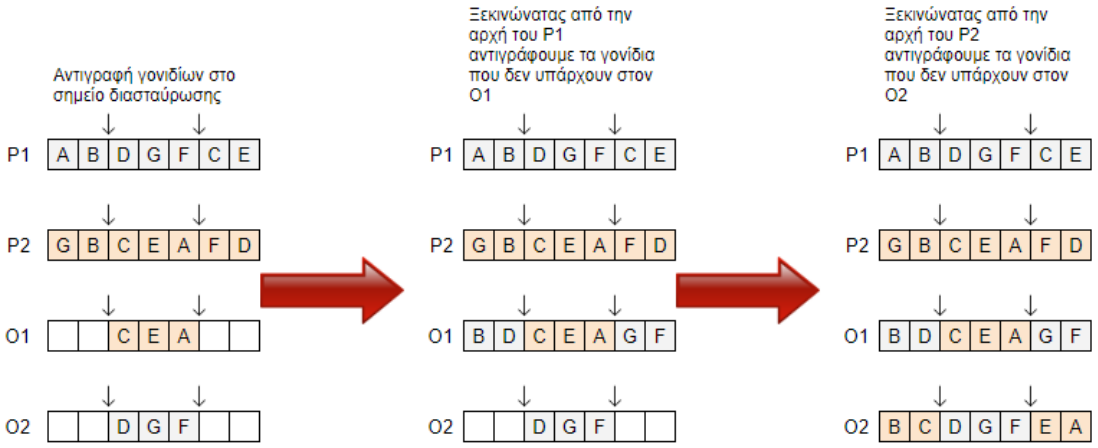
Βήμα 2: Τοποθετούμε τα γονίδια που βρίσκονται στον πρώτο γονέα P1 από τη θέση C1 μέχρι τη θέση C2 στον απόγονο O2 στις αντίστοιχες θέσεις. Τοποθετούμε τα γονίδια που βρίσκονται στο δεύτερο γονέα P2 από τη θέση C1 μέχρι τη θέση C2 στον απόγονο O1 στις αντίστοιχες θέσεις. Θέτουμε i=1.

Βήμα 3: Ελέγχουμε αν το γονίδιο i του γονέα P1 υπάρχει ήδη στον απόγονο O1. Αν δεν υπάρχει το τοποθετούμε στην 1η ελεύθερη θέση του απογόνου O1. Ελέγχουμε αν το γονίδιο i του γονέα P2 υπάρχει ήδη στον απόγονο O2. Αν δεν υπάρχει την τοποθετούμε στην 1η ελεύθερη θέση του απογόνου O2. Αυξάνουμε το i κατά 1, i=i+1.

Βήμα 4: Αν i=N+1 ο αλγόριθμος τερματίζει. Αν όχι επιστρέφουμε στο Βήμα 3.

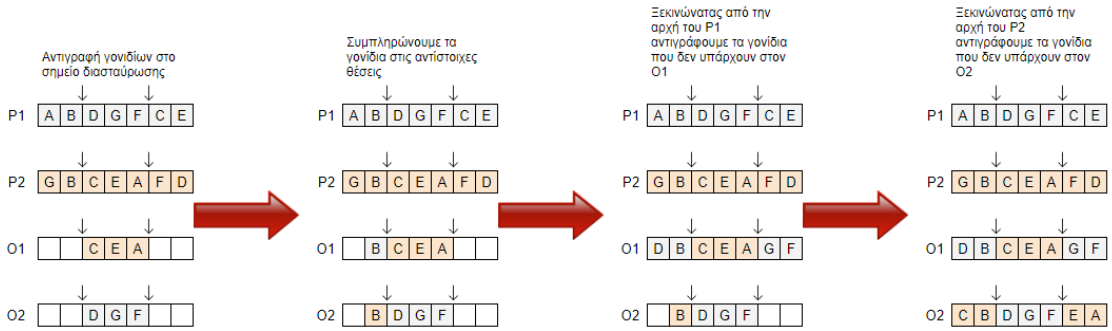
Παράδειγμα: Έστω ότι έχουμε τους γονείς P1= [A, B, D, G, F, C, E] και P2= [G, B, C, E, A, F, D]. Οι δύο τυχαίοι αριθμοί είναι οι C1=3 και C2=5. Σε αυτό το διάστημα υπάρχουν στον γονέα P1 τα γονίδια D,G,F και στον γονέα P2 τα γονίδια C,E,A. Τοποθετούμε στις αντίστοιχες θέσεις του απογόνου O2 τα γονίδια D,G,F O2=[_,_,D,G,F,_,_] και του απογόνου O1 τα γονίδια C,E,A O1=[_,_,C,E,A,_,_].

Ξεκινάμε από την αρχή του γονέα P1. Το γονίδιο A υπάρχει ήδη στον απόγονο O1 άρα πηγαίνουμε στο επόμενο. Το γονίδιο B δεν υπάρχει οπότε το τοποθετούμε O1= [B,_,C,E,A,_,_]. Επαναλαμβάνοντας τα ίδια βήματα παίρνουμε τους δύο απογόνους O1 = [B,D,C,E,A,G,F] και O2 = [B,C,D,G,F,E,A]. Η διαδικασία φαίνεται στο σχήμα που ακολουθεί.



Εικόνα 71 Διασταύρωση δύο σημείων για κωδικοποίηση μετάθεσης

Όπως και στη διασταύρωση ενός σημείου ο διορθωτικός αλγόριθμος μπορεί να τοποθετεί πρώτα τα γονίδια από τους γονείς στις αντίστοιχες ελεύθερες θέσεις των απογόνων.



Εικόνα 72 Διασταύρωση δύο σημείων για κωδικοποίηση μετάθεση συμπληρώνοντας πρώτα τα γονίδια στις αντίστοιχες ελεύθερες θέσεις

Σημείωση: Ο αλγόριθμος διασταύρωσης 2 σημείων μπορεί να επεκταθεί σε Κ σημεία διασταύρωσης αυξάνοντας έτσι το πεδίο λύσεων αλλά μπορεί να έχουμε χειρότερη απόδοση λόγω της διατάραξης των building blocks (Lobo, 2000). Σε μερικά βιβλία ο αλγόριθμος διασταύρωσης δύο σημείων αναφέρεται ως Order Crossover 2 (OX2).

Σημείωση: Οι αλγόριθμοι διασταύρωσης Κ σημείων δεν έχουν καλή απόδοση σε γενετικούς αλγορίθμους που χρησιμοποιούν την κωδικοποίηση μετάθεσης. Όπως θα δούμε και στη συνέχεια για αυτούς τους γενετικούς αλγορίθμους υπάρχουν ειδικοί αλγόριθμοι διασταύρωσης.

Η υλοποίηση του αλγορίθμου σε php για την κωδικοποίηση μετάθεσης βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου σε php για τη δυαδική κωδικοποίηση βρίσκεται εδώ.

7.3 Ομοιόμορφη διασταύρωση - Uniform crossover (UX)

Η ομοιόμορφη διασταύρωση προτάθηκε το 1989 από τον Syswerda (Syswerda G. , 1989). Σε αυτή τη μορφή διασταύρωσης ο πρώτος απόγονος κληρονομεί κάθε γονίδιο με πιθανότητα P από τον πρώτο γονέα και με πιθανότητα 1-Ραπό τον δεύτερο. Ο δεύτερος απόγονος κληρονομεί κάθε γονίδιο με πιθανότητα P από τον δεύτερο γονέα και με πιθανότητα 1-Ραπό τον πρώτο.

Στην αρχική έκδοση του αλγορίθμου η πιθανότητα P είναι 0.5 αλλά στη συνέχεια έγιναν υλοποιήσεις με διαφορετική πιθανότητα, δίνοντας έτσι την ευκαιρία στους απογόνους να κρατήσουν

περισσότερα γονίδια από τον ένα γονέα. Επειδή η ομοιόμορφη διασταύρωση γίνεται σε ένα γονίδιο κάποιοι ερευνητές χρησιμοποιούν ως P την πιθανότητα διασταύρωσης Pc.

Αν P=0.5 l/2 γονίδια κατά μέσο όρο θα ανταλλάξουν θέσεις όπου l ο αριθμός των γονιδίων του χρωμοσώματος. Στην πράξη θα συμβούν πολύ λιγότερες ανταλλαγές γονιδίων γιατί κάποια γονίδια θα έχουν την ίδια τιμή. Η πιθανότητα δύο γονίδια να έχουν την ίδια τιμή είναι 1/C όπου Co αριθμός των γραμμάτων του αλφαβήτου που έγινε η κωδικοποίηση, άρα η πιθανότητα τα γονίδια να είναι διαφορετικά είναι 1-1/C. Επειδή όμως στους γενετικούς αλγορίθμους τα γονίδια δεν έχουν πάρει τυχαίες τιμές, αλλά οι καλές λύσεις έχουν ομοιότητες, η πιθανότητα είναι πολύ μικρότερη.

Είναι προφανές ότι ο συγκεκριμένος αλγόριθμος δημιουργήθηκε για την δυαδική κωδικοποίηση και δεν μπορεί να χρησιμοποιηθεί στην κωδικοποίηση μετάθεσης. Η απόδοσή του στους Γ.Α. που χρησιμοποιούν δυαδική κωδικοποίηση φαίνεται να είναι καλύτερη σε σύγκριση με άλλους γνωστούς αλγορίθμους διασταύρωσης, ειδικά σε υλοποιήσεις με μικρό μέγεθος πληθυσμού (Stjepan Picek M. G., 2020). Τα βήματα του αλγόριθμου ομοιόμορφης διασταύρωσης είναι τα εξής:

Βήμα 1ο: Έστω P1, P2 οι δύο γονείς και O1, O2 οι δύο απόγονοι. Επιλέγουμε την πιθανότητα P με την οποία θα γίνει η ανταλλαγή των γονιδίων. Θέτουμε i=1.

Βήμα 2ο: Παράγουμε ένα τυχαίο αριθμό r[0,1).

Βήμα 3ο: Αν r<P τα γονίδια που βρίσκονται στη θέση i ανταλλάσσουν θέσεις, δηλαδή O1[i]=P2[i] και O2[i]=P1[i]. Αν δεν ισχύει r<P τότε O1[i]=P1[i] και O2[i]=P2[i]. Αυξάνουμε το i κατά 1 i=i+1.

Βήμα 4ο: Αν i>N όπου N ο αριθμός των γονιδίων του χρωμοσώματος, ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1=[1,0,0,1,1,0,1,0,1,0], P2=[0,0,1,0,1,0,0,1,0,0] και P=0.5. Ο αλγόριθμος της ομοιόμορφης διασταύρωσης φαίνεται στο σχήμα που ακολουθεί:

Εικόνα 73 Ομοιόμορφη διασταύρωση

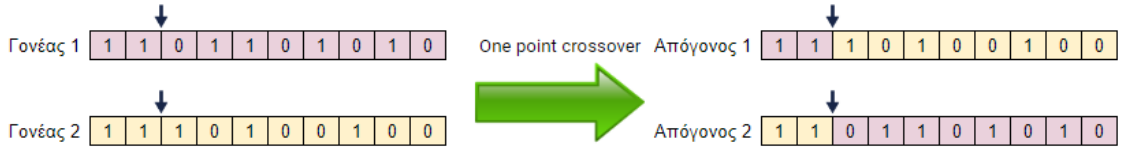
Σημείωση: Η ομοιόμορφη διασταύρωση λόγω του μεγάλου αριθμού τυχαίων αριθμών που δημιουργούνται είναι πιο αργή από άλλες μορφές διασταύρωσης. Για να αυξήσουμε την ταχύτητα μπορούμε να κάνουμε τα εξής:

- Αν η πιθανότητα P είναι 0.5 αντί να παράγουμε τυχαίους αριθμούς στο διάστημα [0,1) μπορούμε να παράγουμε τυχαία 2 ακεραίους π χ 0 και 1.
- Αντί να παράγουμε τυχαίους αριθμούς για όλα τα γονίδια, παράγουμε μόνο για τα γονίδια που είναι διαφορετικά στους δύο γονείς.

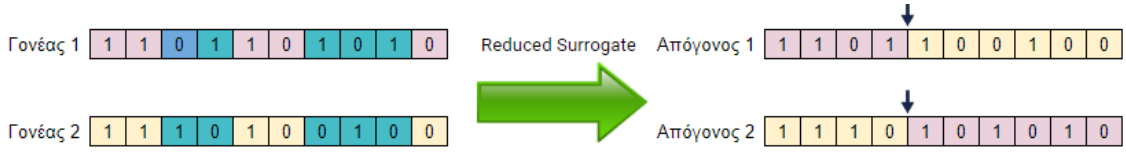
Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.4 Reduced surrogate crossover - SC

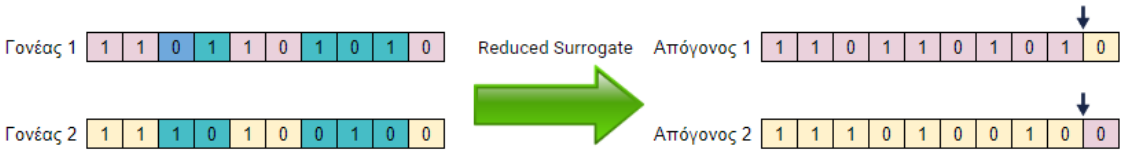
Ο αλγόριθμος reduced surrogate προτάθηκε το 1987 από τον L. Booker (Booker, 1987). Ένα από τα προβλήματα της διασταύρωσης ενός σημείου είναι ότι πολλές φορές οι απόγονοι είναι ίδιοι με τους γονείς. Ο αλγόριθμος reduced surrogate προσπαθεί να περιορίσει αυτό το πρόβλημα. Για να το πετύχει δημιουργεί μια λίστα με τις θέσεις όπου τα γονίδια των γονέων είναι διαφορετικά και επιλέγει ως σημείο διασταύρωσης ένα από τα στοιχεία της λίστας. Προφανώς ο αλγόριθμος δε χρησιμοποιείται στην κωδικοποίηση μετάθεσης.



Εικόνα 74 One point crossover. Οι γονείς είναι οι ίδιοι γιατί μέχρι το σημείο διασταύρωσης όλα τα γονίδια ήταν ίδια.



Εικόνα 75 Reduced surrogate. Επιλέγουμε τυχαία ένα από τα πιθανά σημεία διασταύρωσης.



Εικόνα 76 Reduced surrogate. Οι απόγονοι είναι ίδιοι με τους γονείς αν επιλέξουμε το τελευταίο σημείο διασταύρωσης

Σημείωση: Αν επιλέξουμε ως σημείο διασταύρωσης το τελευταίο διαθέσιμο σημείο, οι απόγονοι θα είναι ίδιο με τους γονείς.

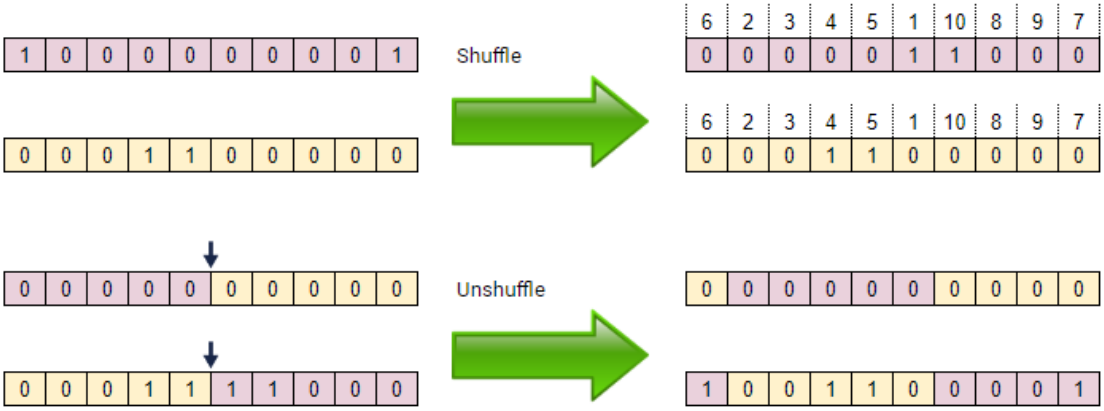
Η υλοποίηση του αλγόριθμου σε php βρίσκεται εδώ.

7.5 Shuffle crossover

Όταν ένας γονέας έχει τα καλά του γονίδια στην αρχή και στο τέλος, η πιθανότητα να συνυπάρχουν στον ίδιο απόγονο είναι πολύ μικρή (positional bias (Eshelman L. C., 1989)). Για να λύσει αυτό το πρόβλημα ο F.J. Burkowski (Burkowski, 1999) δημιούργησε τον αλγόριθμο shuffle crossover.

Ο αλγόριθμος shuffle crossover ακολουθεί τα βήματα του αλγορίθμου one point crossover με τη διαφορά ότι πριν γίνει η διασταύρωση τα γονίδια και στους δύο γονείς ανακατεύονται με τον ίδιο τρόπο. Μετά την διασταύρωση τα γονίδια επιστρέφουν στην αρχική τους θέση. Προφανώς ο συγκεκριμένος αλγόριθμος δεν χρησιμοποιείται αν έχουμε κωδικοποίηση μετάθεσης.

Παράδειγμα: Έστω ότι έχουμε ένα πρόβλημα με βέλτιστη λύση την (1, 1, 1, 1, 1, 1, 1, 1, 1, 1). Στο σχήμα που ακολουθεί τα γονίδια στις θέσεις 1,6 και τα γονίδια στις θέσεις 7,10 αντάλλαξαν θέσεις. Αυτό μας επέτρεψε να πάρουμε έναν απόγονο ο οποίος περιέχει τους άσσους και από τους δύο γονείς.



Εικόνα 77 Shuffle crossover

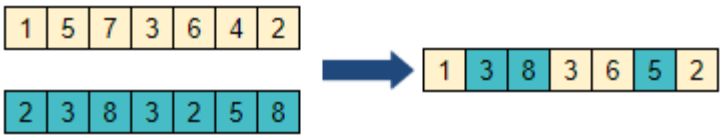
Η υλοποίηση του αλγορίθμου shuffle crossover σε php βρίσκεται εδώ.

7.6 Διακριτή διασταύρωση - Discrete crossover

Ο αλγόριθμος της διακριτής διασταύρωσης (Stjepan Picek D. J., 2013) (H. Voigt, 1995) (Eiben, 2003) σε αντίθεση με τους περισσότερους αλγορίθμους διασταύρωσης παράγει έναν απόγονο από δύο γονείς (μπορεί να χρησιμοποιηθεί και για περισσότερους γονείς).

Κάθε γονίδιο του απογόνου παίρνει τυχαία, με την ίδια πιθανότητα, μια τιμή από τα γονίδια των γονέων που βρίσκονται στην αντίστοιχη θέση. Μπορεί να χρησιμοποιηθεί για οποιαδήποτε μορφή κωδικοποίησης εκτός από την κωδικοποίηση μετάθεσης. Σε αντίθεση με άλλες μορφές διασταύρωσης που χρησιμοποιούνται στους γενετικούς αλγορίθμους στους οποίους η αναπαράσταση γίνεται με τη χρήση ακεραίων ή αριθμών κινητής υποδιαστολής, δεν εισάγει κάποια καινούργια πληροφορία στον απόγονο. Τα γονίδια προέρχονται αποκλειστικά από τους γονείς και η εισαγωγή νέας πληροφορίας μπορεί να γίνει μόνο από τον τελεστή της μετάλλαξης.

Στο σχήμα που ακολουθεί βλέπουμε τη διακριτή διασταύρωση δύο ατόμων στα οποία η αναπαράσταση γίνεται με ακέραιους αριθμούς.



Εικόνα 78 Discrete crossover

Η υλοποίηση τους αλγορίθμου σε php βρίσκεται εδώ.

7.7 Διασταύρωση ανάμιξης άλφα - Blend alpha crossover (BLX-α)

Ο αλγόριθμος BLX-α δημιουργήθηκε το 1993 από τους Eshelman, L.J. and Schaffer, J.D. (Eshelman L. a., 1993) και είναι ένας αλγόριθμος διασταύρωσης για γενετικούς αλγορίθμους στους οποίους η αναπαράσταση των λύσεων γίνεται με αριθμούς κινητής υποδιαστολής. Ο αλγόριθμος παράγει έναν απόγονο από δύο ή και περισσότερους γονείς.

Αν έχουμε δύο γονείς P_1, P_2 για κάθε γονίδιο i του απογόνου O επιλέγεται τυχαία ένας αριθμός στο διάστημα $[X_1^i, X_2^i]$ όπου $X_1^i = \min(P_1^i, P_2^i) - a * d_i$, $X_2^i = \max(P_1^i, P_2^i) - a * d_i$, $d_i = |P_1^i - P_2^i|$. Αν θέσουμε $\gamma_i = (1 + 2a) * u_i - a$ τότε έχουμε $O^i = (1 - \gamma_i) * min(P_1^i, P_2^i) + \gamma_i * max(P_1^i, P_2^i)$(1) με $u_i \in [0,1]$ ένας τυχαίος αριθμός και α μια σταθερά που συνήθως παίρνει την τιμή α=0.5 (Eshelman L. a.,

1993), αν και κάποιες έρευνες έχουν καλύτερα αποτελέσματα με διαφορετική τιμή του α (Masato Takahashi, 2001). Για α=0 παίρνουμε οποιαδήποτε τιμή ανάμεσα στην ελάχιστη και τη μέγιστη. Ο αλγόριθμος αυτός είναι γνωστός με την ονομασία flat crossover (Radcliffe, 1991).

Από τη σχέση (1) αν την γράψουμε $O^i - min(P_1^i, P_2^i) = \gamma_i * [max(P_1^i, P_2^i) - min(P_1^i, P_2^i)]$ προκύπτει ότι αν η διαφορά των γονέων είναι μικρή τότε θα είναι μικρή και η διαφορά του απογόνου από τους γονείς.

Εικόνα 79 Πιθανές τιμές απογόνου BLX-α

Η υλοποίηση του αλγορίθμου BLX-α βρίσκεται εδώ.

7.8 Parent-centric BLX-α (PBX-α)

Ο αλγόριθμος PBX-α προτάθηκε το 2004 από τους M.Lozano et. al. (Manuel Lozano, 2004) (C.García-Martínez, 2008) για τους γενετικούς αλγορίθμους στους οποίους τα γονίδια του χρωμοσώματος είναι αριθμοί κινητής υποδιαστολής. Είναι μια παραλλαγή του αλγορίθμου BLX-α, η οποία όπως καταλαβαίνουμε και από την ονομασίας της, δίνει μεγαλύτερη πιθανότητα στα γονίδια του απογόνου να πάρουν τιμές κοντά σε αυτές που έχουν τα γονίδια των γονέων. Ο αλγόριθμος παράγει έναν απόγονο από δύο ή και περισσότερους γονείς.

Αν έχουμε δύο γονείς $P_1 = (P_1^1, P_1^1, \dots, P_1^i)$, $P_2 = (P_2^1, P_2^1, \dots, P_2^i)$ με $P_1^i, P_2^i \in [a^i, b^i]$ παίρνουμε τυχαία (με την ίδια πιθανότητα) έναν από τους απογόνους $O_1 = (O_1^1, O_1^1, \dots, O_1^i)$, και $O_2 = (O_2^1, O_2^1, \dots, O_2^i)$), όπου O_1^i ένας τυχαίος αριθμός στο διάστημα $[L_1^i, U_1^i]$ και O_2^i ένας τυχαίος αριθμός στο διάστημα $[L_2^i, U_2^i]$ με:

$$L_1^i = max(a_i, P_1^i - \alpha \cdot d_i)$$

$$L_2^i = max(a_i, P_2^i - \alpha \cdot d_i)$$

$$U_1^i = min(b_i, P_1^i + \alpha \cdot d_i)$$

$$U_2^i = min(b_i, P_2^i + \alpha \cdot d_i)$$

$$d_i = \left| P_1^i - P_2^i \right|$$

Εικόνα 80 Τιμές παραμέτρων Parent-centric BLX-α (PBX-α)

Επειδή η επιλογή των γονέων είναι έτσι και αλλιώς τυχαία, το κάθε άτομο έχει την ίδια πιθανότητα να επιλεγεί ως γονέας P1και ως γονέας P2. Για αυτό το λόγο μπορούμε να ορίσουμε (C.García-Martínez, 2008) πιο απλά τον αλγόριθμο PBX-α ως εξής.

Αν έχουμε δύο γονείς $P_1 = (P_1^1, P_1^1, \dots, P_1^i)$, $P_2 = (P_2^1, P_2^1, \dots, P_2^i)$ με $P_1^i, P_2^i \in [a^i, b^i]$ με τη διασταύρωση PBX-α παίρνουμε τον απόγονο $O = (O^1, O^2, \dots, O^i)$ όπου O^i) ένας τυχαίος αριθμός στο διάστημα $[L^i, U^i]$ με:

$$L^i = max(a_i, P_1^i - \alpha \cdot d_i)$$

$$U^i = min(b_i, P_1^i + \alpha \cdot d_i)$$

$$d_i = \left| P_1^i - P_2^i \right|$$

Εικόνα 81 Τιμές παραμέτρων Parent-centric BLX-α (PBX-α) v2

Ο γονέας P_1 ονομάζεται θηλυκός γονέας (female parent) και καθορίζει την περιοχή από την οποία θα πάρουμε τα γονίδια του απογόνου. Ο γονέας P_2 ονομάζεται αρσενικός γονέας (male parent) και καθορίζει το εύρος αυτής της περιοχής. Το εύρος της περιοχής επηρεάζεται επίσης από τη σταθερά $a \in [0.5,1]$. Η αύξηση της τιμής του α αυξάνει τον αριθμό των τιμών που μπορεί να πάρει το γονίδιο.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P_1= (2.5, 4.6, 1.8, 7.2) και P_2 = (4.6, 2.1, 1.8, 8). Τα κατώτατά όρια των γονιδίων είναι a=(2.4, 1.5, 0, 5.3) και τα ανώτατα b=(5.5, 4.6, 4, 10).

Για a=0.6 τα γονίδια θα πάρουν τυχαία τιμές ανάμεσα στα όρια:

Εικόνα 82 Όρια γονιδίων για α=0.6

Για a=0.9 τα γονίδια θα πάρουν τυχαία τιμές ανάμεσα στα όρια:

Εικόνα 83 Όρια γονιδίων για α=0.9

Παρατηρούμε ότι για μεγαλύτερο α το εύρος των τιμών που μπορεί να πάρουν τα γονίδια είναι μεγαλύτερο.

Η υλοποίηση του αλγόριθμου PBX-α βρίσκεται εδώ.

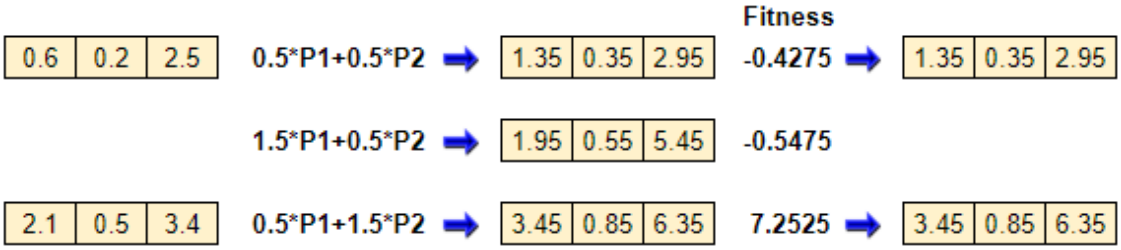
7.9 Γραμμική διασταύρωση - Linear crossover

Ο αλγόριθμος της γραμμικής διασταύρωσης προτάθηκε το 1999 από τον Wright (Wright, 1999) στην προσπάθειά του να δημιουργήσει καλύτερους απογόνους όταν η αναπαράσταση των λύσεων γίνεται με αριθμούς κινητής υποδιαστολής.

Για κάθε ζεύγος γονέων P1 και P2 δημιουργούνται τρεις υποψήφιοι απόγονοι και επιλέγουμε τους δύο με την καλύτερη απόδοση:

- O1=0.5*P1+0.5*P2
- O2=1.5*P1+0.5*P2
- O3=0.5*P1+1.5*P2

Παράδειγμα: Θέλουμε να μεγιστοποιήσουμε τη συνάρτηση f(x, y, z) = x^2+2y-z.



Εικόνα 84 Linear crossover

Η υλοποίηση του αλγόριθμου linear crossover βρίσκεται εδώ.

7.10 Προσομοίωση δυαδικής διασταύρωσης - Simulated binary crossover (SBX)

Ο αλγόριθμος simulated binary crossover δημιουργήθηκε το 1995 από τους K. Deb και Ram Bhushan Agrawal (K. Deb, 1995) για τους γενετικούς αλγορίθμους που χρησιμοποιούν αναπαράσταση αριθμών κινητής υποδιαστολής. Αν έχουμε δύο γονείς P_1 και P_2 παράγουμε ένα τυχαίο αριθμό $u \in [0,1)$ και οι δύο απόγονοι προκύπτουν από τους τύπους

$$O_1 = 0.5 \cdot [(1 + \beta) \cdot P_1 + (1 - \beta) \cdot P_2] \quad (1)$$

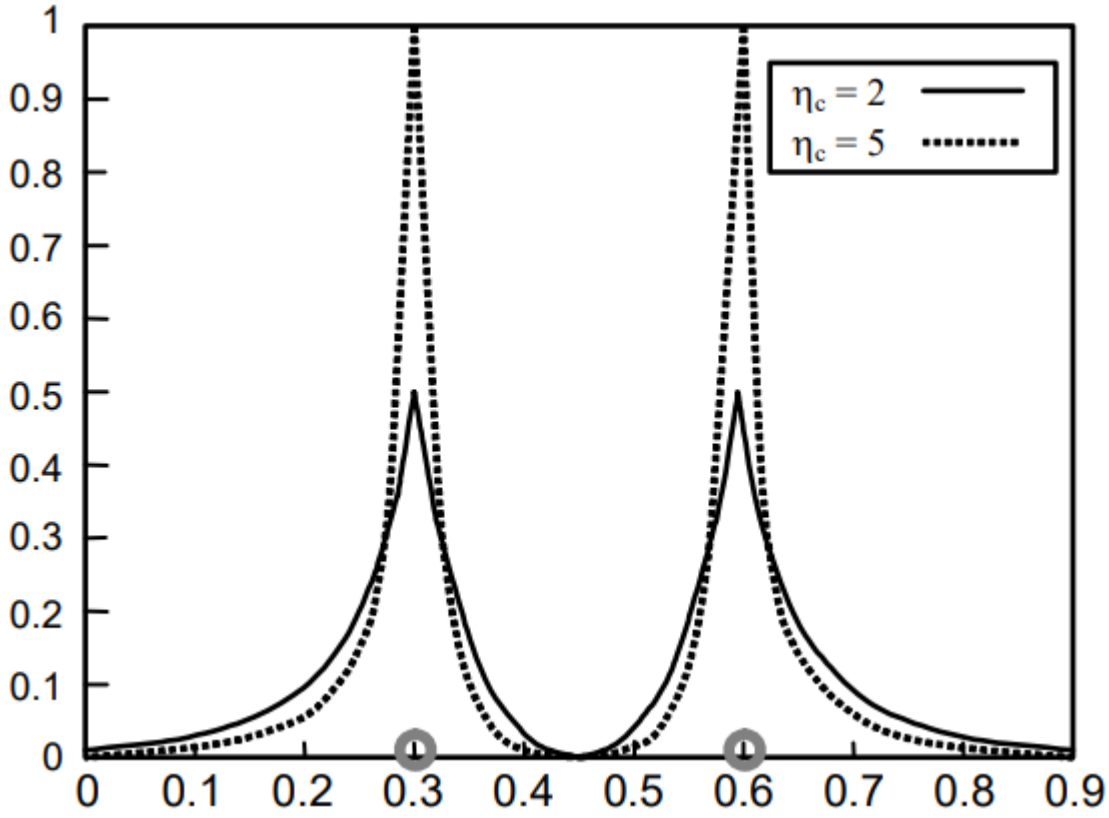
$$O_2 = 0.5 \cdot [(1 - \beta) \cdot P_1 + (1 + \beta) \cdot P_2] \quad (2)$$

$$\beta = (2u)^{1/(\eta+1)}, \text{ αν } u \leq 0.5$$

$$\beta = [1/(2 \cdot (1 - u))]^{1/(\eta+1)}, \text{ αν } u > 0.5$$

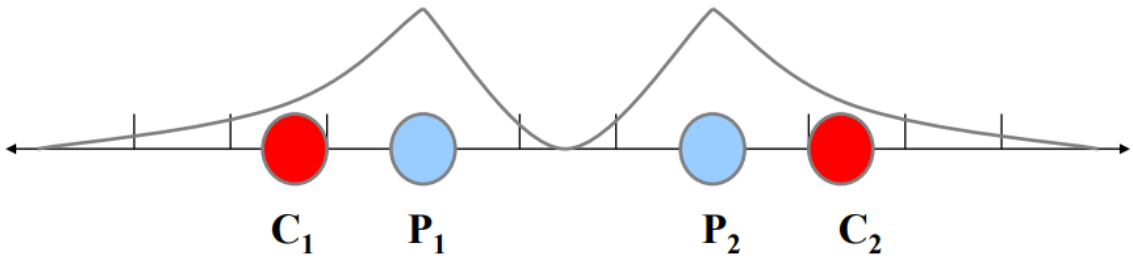
Εικόνα 85 Τύποι SBX

Το η είναι μια σταθερά η οποία καθορίζει την κατανομή. Αν το η είναι μεγάλο, οι απόγονοι θα βρίσκονται κοντά στους γονείς.



Εικόνα 86 Τιμές που παίρνουν οι απόγονοι αν οι γονείς είναι P1=0.3, P2=0,6 για η=2 και η=5

Όπως καταλαβαίνουμε και από το όνομα ο αλγόριθμος SBX προσομοιώνει τη διασταύρωση ενός σημείου που χρησιμοποιούμε στους δυαδικούς γενετικούς αλγορίθμους. Οι απόγονοι έχουν μεγαλύτερη πιθανότητα να βρίσκονται κοντά στους γονείς και από τις σχέσεις (1) και (2) προκύπτει ότι ο μέσος όρος της τιμής των απογόνων είναι ίδιος με το μέσο όρο της τιμής των γονέων.



Εικόνα 87 Οι απόγονοι έχουν μεγαλύτερη πιθανότητα να βρίσκονται κοντά στους γονείς

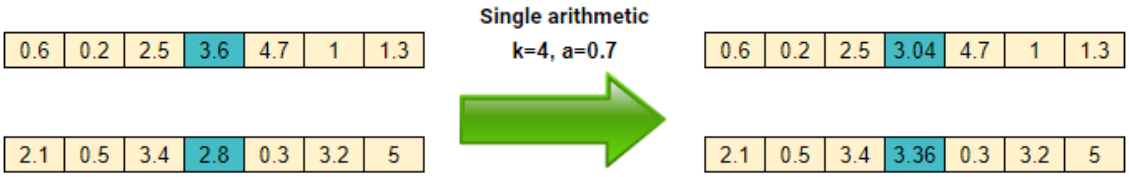
Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.11 Αριθμητική διασταύρωση - Arithmetic crossover

Για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση αριθμών κινητής υποδιαστολής έχουν αναπτυχθεί πάρα πολλοί αλγόριθμοι διασταύρωσης (Michalewicz, 1992) (F. Herrera M. L., 1998) (Stjepan Picek D. J., 2013). Μια κατηγορία αυτών των αλγορίθμων είναι οι αλγόριθμοι αριθμητικής διασταύρωσης.

7.11.1 Single arithmetic crossover

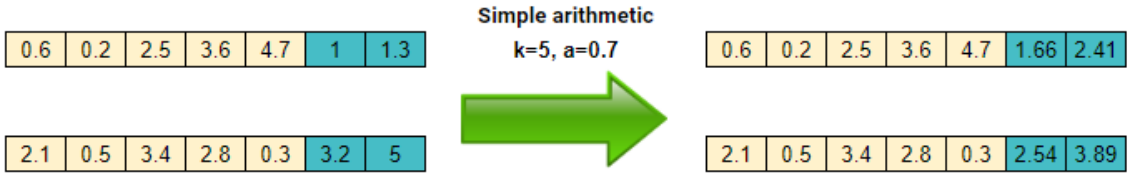
Δύο γονείς P_1 και P_2 επιλέγονται τυχαία για διασταύρωση με πιθανότητα P_c. Στη συνέχεια παράγουμε έναν τυχαίο αριθμό $k \in [1, N]$ όπου N ο αριθμός των γονιδίων. Τα γονίδια των απογόνων που βρίσκονται σε όλες τις θέσεις εκτός από τη θέση k αντιγράφονται ως έχουν στους απογόνους. Το γονίδιο που βρίσκεται στη θέση k των απογόνων παίρνει την τιμή $O_1^k = a * P_2^k + (1-a) * P_1^k$, $O_2^k = a * P_1^k + (1-a) * P_2^k$, με $a \in (0,1)$ μια σταθερά. Προφανώς αν a=0.5 οι δύο απόγονοι θα είναι ίδιοι.



Εικόνα 88 Single arithmetic crossover για a=0.7 και k=4

7.11.2 Simple arithmetic crossover

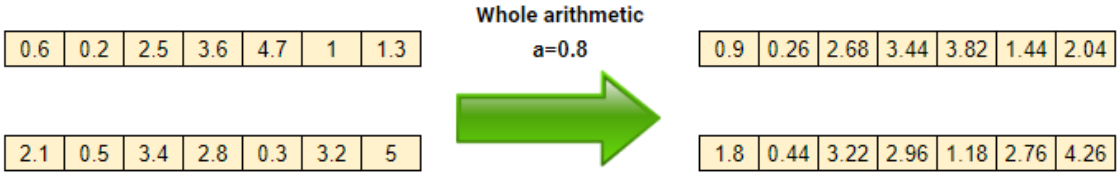
Δύο γονείς P_1 και P_2 επιλέγονται τυχαία για διασταύρωση με πιθανότητα P_c. Στη συνέχεια παράγουμε έναν τυχαίο αριθμό $k \in [1, N]$ όπου N ο αριθμός των γονιδίων. Τα γονίδια των απογόνων που βρίσκονται στις θέσεις 1 έως k αντιγράφονται ως έχουν στους απογόνους. Τα γονίδια που βρίσκονται στις θέσεις $i \in [k+2, N]$ των απογόνων παίρνουν τιμές $O_1^i = a * P_2^i + (1-a) * P_1^i$, $O_2^i = a * P_1^i + (1-a) * P_2^i$, με $a \in (0,1)$ μια σταθερά. Προφανώς αν a=0.5 οι δύο απόγονοι θα είναι ίδιοι.



Εικόνα 89 Simple arithmetic crossover για a=0.7 και k=5

7.11.3 Whole arithmetic crossover

Δύο γονείς P_1 και P_2 επιλέγονται τυχαία για διασταύρωση με πιθανότητα P_c. Για $i \in [1, N]$ όπου N ο αριθμός των γονιδίων, τα γονίδια των δύο απογόνων προκύπτουν από τους τύπους. $O_1^i = a * P_q^i + (1-a) * P_2^i$, $O_2^i = a * P_2^i + (1-a) * P_1^i$, με $a \in (0,1)$ μια σταθερά. Προφανώς αν a=0.5 οι δύο απόγονοι θα είναι ίδιοι.



Εικόνα 90 Whole arithmetic crossover για a=0.8

Η τιμή του a μπορεί να μην είναι σταθερή αλλά είτε κάθε φορά να παίρνει μια τυχαία τιμή (Stjepan Picek D. J., 2013) (D. Dumitrescu, 2000) (local crossover) είτε να αλλάζει με βάση των αριθμό των γενεών (Michalewicz, 1992) (non-uniform arithmetical crossover)

Η υλοποίηση του αλγορίθμου single arithmetic crossover σε php βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου simple arithmetic crossover σε php βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου whole arithmetic crossover σε php βρίσκεται εδώ.

7.12 Ευρετική διασταύρωση - Heuristic Crossover

Το όρο ευρετική διασταύρωση μπορούμε να τον χρησιμοποιήσουμε για όλους τους αλγορίθμους διασταύρωσης οι οποίοι κάνουν χρήση της συνάρτησης καταλληλότητας των γονέων. Ο γονέας με την καλύτερη απόδοση κληρονομεί περισσότερο γενετικό υλικό στους απογόνους. Σε αυτό το κεφάλαιο θα αναλύσουμε την ευρετική διασταύρωση του Wright (F. Herrera M. L., 1998) (Genetic Algorithms for Real Parameter Optimization, 1990), την οποία πολλές φορές τη συναντάμε με την απλή ονομασία ευρετική διασταύρωση. Ο αλγόριθμος χρησιμοποιείται όταν η αναπαράσταση των λύσεων γίνεται με τη χρήση αριθμών κινητής υποδιαστολής.

Αν έχουμε δύο γονείς $P_1 = (P_1^1, P_1^1, \dots, P_1^i)$, $P_2 = (P_2^1, P_2^1, \dots, P_2^i)$ ο αλγόριθμος της ευρετικής διασταύρωσης θα παράγει έναν απόγονο $O = (O^1, O^2, \dots, O^i)$. Για να βρούμε τον απόγονο υπολογίζουμε την απόδοση των δύο γονέων. Αν $f(P_1) \leq f(P_2)$ θέτουμε $P_{max} = P_2$ και $P_{min} = P_1$ ενώ αν $f(P_1) > f(P_2)$ θέτουμε $P_{max} = P_1$ και $P_{min} = P_2$. Ο απόγονος Ο προκύπτει από τον τύπο $O^i = P_{max}^i + r * (P_{max}^i - P_{min}^i)$. Το r μπορεί να είναι είτε ένας τυχαίος αριθμός στο διάστημα $r \in [0,1]$, είτε να έχει σταθερή τιμή r=0.5.

Είναι προφανές ότι οι τιμές των γονιδίων του απογόνου μπορούν να βγαίνουν έξω από τα επιτρεπόμενα όρια. Για αυτό το λόγο ο σχεδιαστής του γενετικού αλγορίθμου πρέπει να θέσει ένα όριο k. Αν τα γονίδια του απογόνου δεν είναι έγκυρες λύσεις, ο αλγόριθμος της ευρετικής διασταύρωσης επαναλαμβάνεται με διαφορετικό r. Αν μετά από k επαναλήψεις δεν προκύψει έγκυρος απόγονος, τότε επιστρέφεται ως απόγονος ο γονέας με την καλύτερη απόδοση. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε κάποιον άλλο αλγόριθμο διασταύρωσης.

Σημειώσεις:

- Σε κάποιες έρευνες ο αλγόριθμος επιστρέφει δύο απογόνους με τον δεύτερο απόγονο να είναι ένα αντίγραφο του γονέα με την καλύτερη απόδοση.
- Σε κάποιες υλοποιήσεις ο τυχαίος αριθμός r παράγεται μία φορά για όλα τα γονίδια, ενώ σε κάποιες άλλες παράγεται ένας τυχαίος αριθμός για κάθε γονίδιο.
- Σε κάποιες υλοποιήσεις ο τυχαίος αριθμός r δεν είναι ακριβώς τυχαίος αλλά ακολουθεί κάποια κατανομή.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.13 Διασταύρωση σειράς - Order Crossover (OX1)

Ο αλγόριθμος Order crossover προτάθηκε από τον Lawrence Davis (Davis L. , Applying adaptive algorithms to epistatic domains., 1985) και έχει αρκετές ομοιότητες με τον αλγόριθμο διασταύρωσης δύο σημείων. Χρησιμοποιείται όταν έχουμε κωδικοποίηση μετάθεσης.

Παράγουμε δύο τυχαίους αριθμούς C1,C2 στο διάστημα [1,N]. Τα γονίδια του γονέα ένα που βρίσκονται μεταξύ των σημείων C1 και C2 αντιγράφονται στον απόγονο δύο και αυτά του γονέα δύο στον απόγονο ένα. Στη συνέχεια ξεκινώντας από το δεύτερο σημείο διασταύρωσης C2 τοποθετούμε με τη σειρά που εμφανίζονται τα γονίδια του γονέα ένα στον απόγονο ένα και του γονέα δύο στον απόγονο δύο, παραλείποντας όποια γονίδια υπάρχουν ήδη στους απογόνους. Έστω ότι έχουμε τους γονείς P1,P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1,O2. Ο αλγόριθμος OX περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Παράγουμε δύο τυχαίους r1, r2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων.

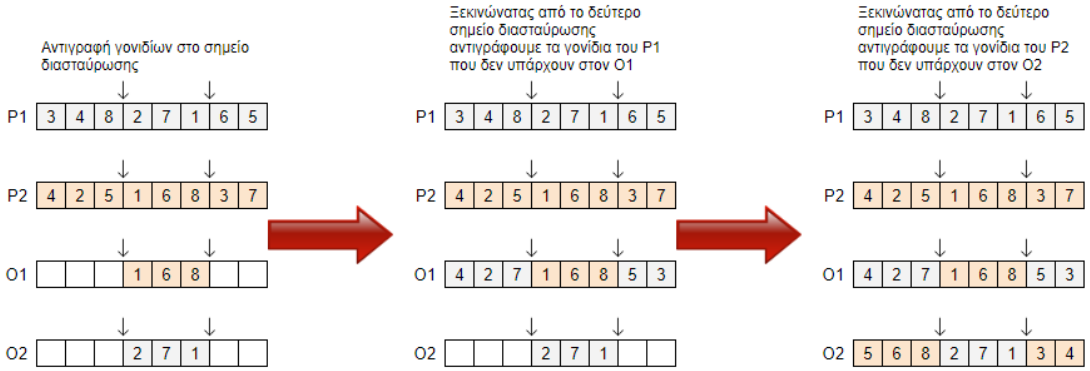
Βήμα 2ο: Αντιγράφουμε τα γονίδια μεταξύ r1 και r2 του P1 στον O1 και μεταξύ r1 και r2 του P2 στον O2.

Βήμα 3ο: Ξεκινώντας από το σημείο r2 (r2> r1) αντιγράφουμε με τη σειρά που εμφανίζονται τα γονίδια από τον P2 στον O2 αν δεν υπάρχουν ήδη στον O2. Όταν φτάσουμε στο τέλος του P2 ξεκινάμε από την αρχή.

Βήμα 4ο: Ξεκινώντας από το σημείο r2 (r2> r1) αντιγράφουμε με τη σειρά που εμφανίζονται τα γονίδια από τον P1 στον O1 αν δεν υπάρχουν ήδη στον O1. Όταν φτάσουμε στο τέλος του P1 ξεκινάμε από την αρχή.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [3, 4, 8, 2, 7, 1, 6, 5] , P2 = [4, 2, 5, 1, 6, 8, 3, 7] και σημεία διασταύρωσης C1=4, C2=6. Αντιγράφοντας τα γονίδια μεταξύ των σημείων διασταύρωσης παίρνουμε τους απογόνους O1 = [_ , _ , _ , 1 , 6 , 8 , _ , _] και O2 = [_ , _ , _ , 2 , 7 , 1 , _ , _]. Από τον P2 και ξεκινώντας από το δεύτερο σημείο διασταύρωσης παίρνουμε τα γονίδια 37425168. Τα γονίδια 2, 7 και 1 υπάρχουν ήδη στον απόγονο O2 οπότε τα διαγράφουμε 34568.

Τοποθετούμε τα γονίδια με αυτή τη σειρά στον O2 ξεκινώντας από το δεύτερο σημείο διασταύρωσης και παίρνουμε τον απόγονο O2 = [5 , 6 , 8 , 2 , 7 , 1 , 3 , 4]. Με την ίδια διαδικασία παίρνουμε τον απόγονο O1 = [4 , 2 , 7 , 1 , 6 , 8 , 5 , 3].



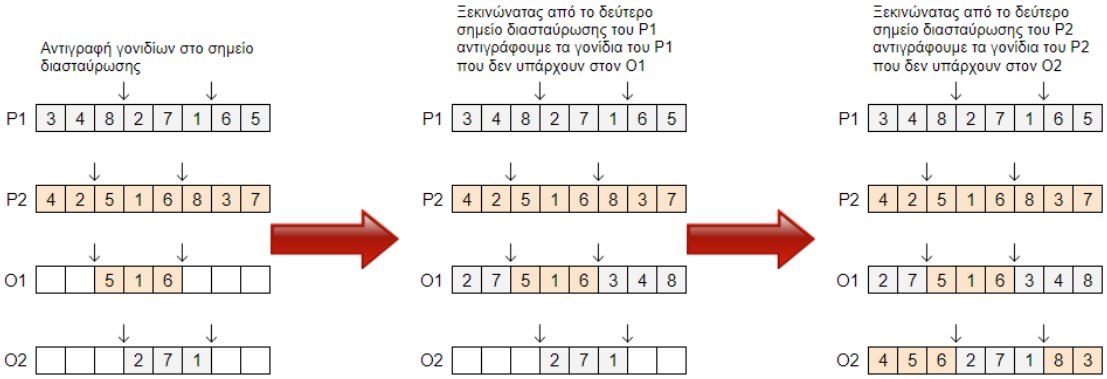
Εικόνα 91 - Order crossover OX1

Υπάρχουν πολλές παραλλαγές του αλγορίθμου order crossover. Δε θα τις δούμε αναλυτικά καθώς δε δίνουν καλύτερα αποτελέσματα (Kusum Deep, 2011), αλλά θα γίνει μια απλή αναφορά.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.13.1 Order Crossover (OX3)

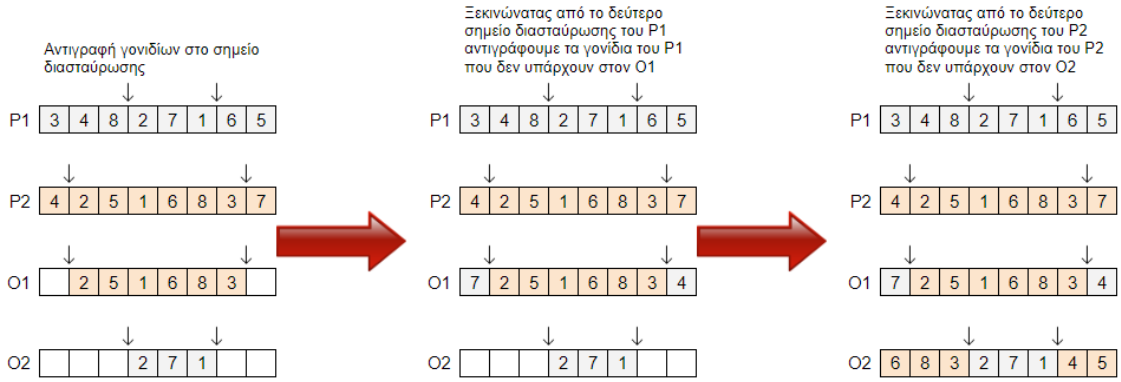
Το μόνο που αλλάζει σε σχέση με τον OX1 είναι ότι τα σημεία διασταύρωσης στους δύο γονείς είναι διαφορετικά. Ο αριθμός των γονιδίων μεταξύ των σημείων διασταύρωσης παραμένει ο ίδιος και στους δύο γονείς.



Εικόνα 92 Order crossover OX3

7.13.2 Order Crossover (OX4)

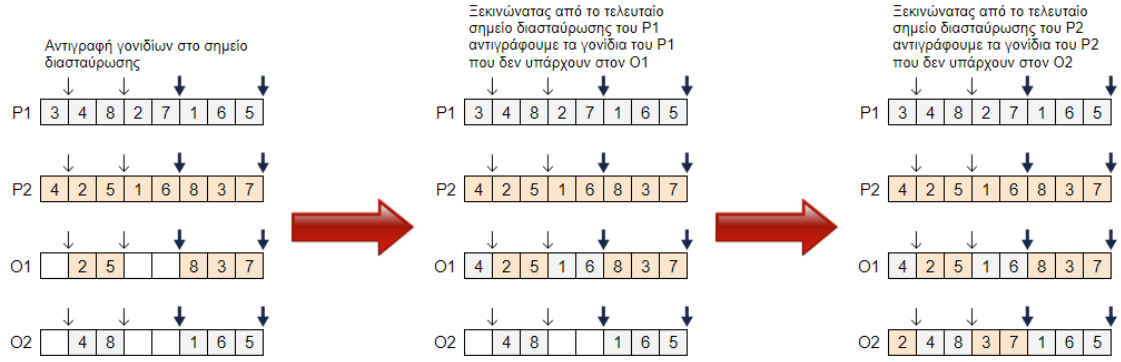
Σε αυτή την έκδοση του order crossover τα σημεία διασταύρωσης στους δύο γονείς είναι διαφορετικά αλλά και ο αριθμός των γονιδίων μεταξύ των σημείων διασταύρωσης είναι επίσης διαφορετικός.



Εικόνα 93 Order crossover OX4

7.13.3 Order Crossover (OX5)

Ίδιος με τον OX1. Η μόνη διαφορά είναι ότι έχουμε δύο ζευγάρια σημείων διασταύρωσης.



Εικόνα 94 Order crossover OX5

7.14 Order based crossover (OX2 ή OBX)

Ο αλγόριθμος προτάθηκε το 1991 από τον Syswerda (Syswerda G. , 1991) και είναι αποκλειστικά για Γ.Α. που χρησιμοποιούν κωδικοποίηση μετάθεσης. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Η υλοποίηση του αλγορίθμου περιγράφεται στα παρακάτω βήματα:

Βήμα 1: Παράγουμε n τυχαίους αριθμούς στο διάστημα [1,N], όπου N ο αριθμός των γονιδίων. Οι αριθμοί αυτοί αποτελούν τα σημεία διασταύρωσης.

Βήμα 2: Βρίσκουμε ποια γονίδια του P1 βρίσκονται στις n θέσεις και τα τοποθετούμε σε μια λίστα L.

Βήμα 3: Αναζητούμε αυτά τα γονίδια στον P2 και σημειώνουμε τις θέσεις που βρέθηκαν.

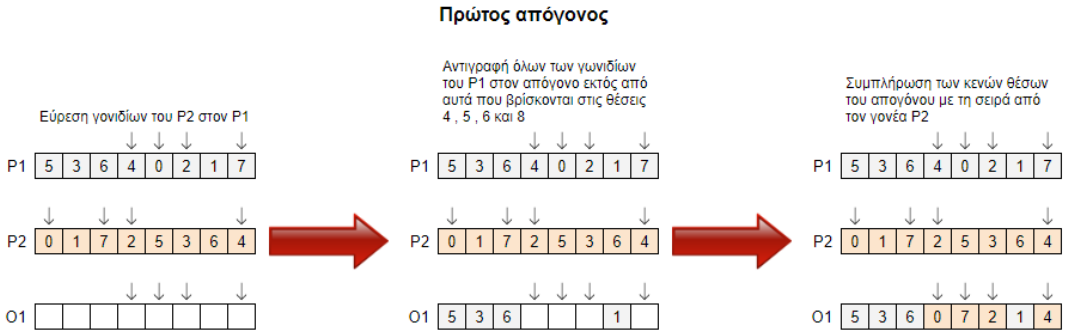
Βήμα 4: Τοποθετούμε τα γονίδια του P2 εκτός από αυτά που σημειώσαμε στο προηγούμενο βήμα στις ΑΝΤΙΣΤΟΙΧΕΣ θέσεις του O1.

Βήμα 5: Συμπληρώνουμε τις υπόλοιπες θέσεις του O1 βάζοντας τα στοιχεία της λίστας L με τη σειρά.

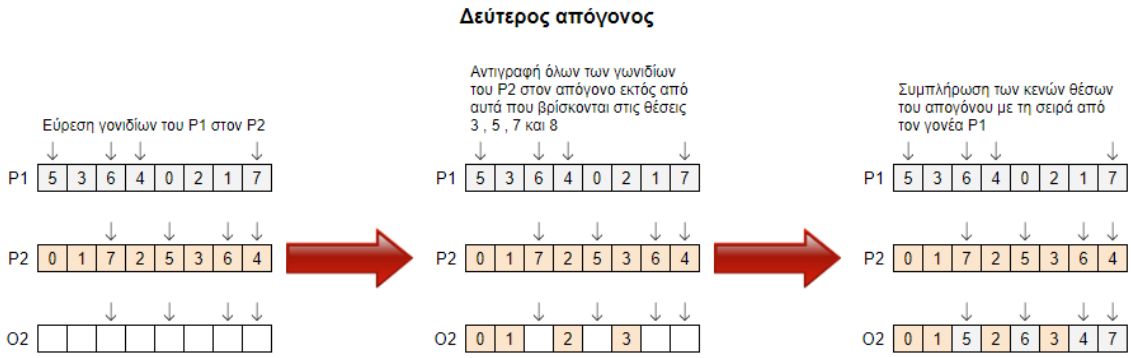
Βήμα 6: Επιστρέφουμε στο Βήμα 2 και αλλάζουμε τους ρόλους των P1, P2 για να πάρουμε τον δεύτερο απόγονο.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [5, 3, 6, 4, 0, 2, 1, 7] και P2 = [0, 1, 7, 2, 5 , 3, 6, 4]. Παράγονται οι τυχαίοι αριθμοί 1, 3, 4 και 8. Στις θέσεις 1, 3, 4 και 8 του P2 βρίσκονται τα γονίδια 0, 7, 2 και 4. Αναζητούμε αυτά τα γονίδια στον P1 και τα βρίσκουμε στις θέσεις 4, 5, 6 και 8. Αντιγράφουμε όλα τα γονίδια του P1 στον απόγονο O1 στις αντίστοιχες θέσεις εκτός από αυτά που υπάρχουν στις θέσεις 4, 5, 6, 8 οπότε ο απόγονος O1 γίνεται O1 = [5, 3, 6, _, _, _, 1, _].

Συμπληρώνουμε τις θέσεις 4, 5, 6 και 8 του απογόνου O1 βάζοντας με τη σειρά τα γονίδια που υπάρχουν στις θέσεις 1, 3, 4, 8 του P2, δηλαδή τα γονίδια 0, 7, 2, 4 και παίρνουμε τον απόγονο O1 = [5, 3, 6, 0, 7, 2, 1, 4]. Ακολουθώντας τα ίδια βήματα αλλάζοντας τις θέσεις των P1, P2 παίρνουμε τον δεύτερο απόγονο O2 = [0, 1, 5, 2, 6, 3, 4, 7].



Εικόνα 95 Order based crossover πρώτος απόγονος



Εικόνα 96 Order based crossover δεύτερος απόγονος

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.15 Διασταύρωση θέσης - Position Crossover (PX ή POS)

Ο τελεστής διασταύρωσης position crossover (Syswerda G. , 1991) είναι μια παραλλαγή των OX1 και OX2. Όπως και στον OX2 παράγουμε n τυχαίους αριθμούς στο διάστημα [1, N], όπου N ο αριθμός των γονιδίων. Τα γονίδια του γονέα P1 που βρίσκονται στις θέσεις P1[i] με i στο διάστημα [1, N] αντιγράφονται στον απόγονο O1. Στη συνέχεια αρχίζοντας από την αρχή του δεύτερου γονέα P2 συμπληρώνουμε με τη σειρά τις κενές θέσεις του απογόνου. Αλλάζοντας τη σειρά των P1, P2 παίρνουμε τον δεύτερο απόγονο. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα:

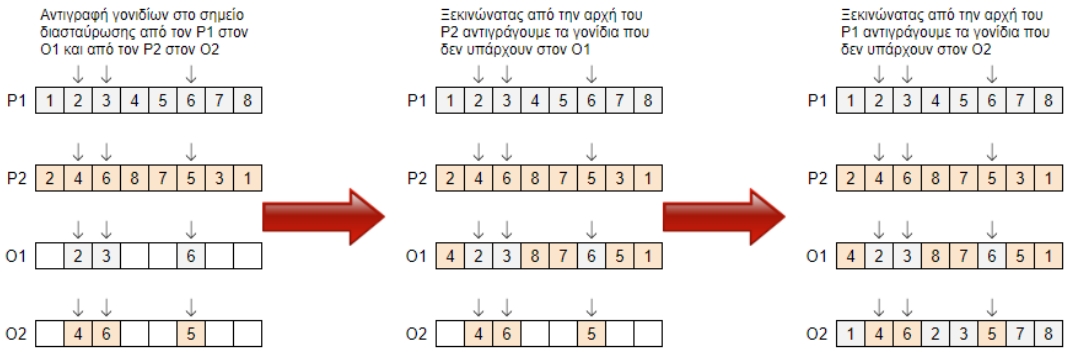
Βήμα 1: Παράγουμε n τυχαίους αριθμούς στο διάστημα [1, N], όπου N ο αριθμός των γονιδίων. Οι αριθμοί αυτοί αποτελούν τα σημεία διασταύρωσης.

Βήμα 2: Βρίσκουμε τα γονίδια του P 1που βρίσκονται στις n θέσεις και τα τοποθετούμε στις αντίστοιχες θέσεις του απογόνου O1. Βρίσκουμε τα γονίδια του P2 που βρίσκονται στις n θέσεις και τα τοποθετούμε στις αντίστοιχες θέσεις του απογόνου O2. Θέτουμε i=1.

Βήμα 3: Ελέγχουμε αν το i-στο στοιχείο του P2 δηλαδή το P2[i] υπάρχει στον απόγονο O1. Αν δεν υπάρχει το τοποθετούμε στην πρώτη ελεύθερη θέση του O1. Ελέγχουμε αν το i-στο στοιχείο του P1 δηλαδή το P1[i] υπάρχει στον απόγονο O2. Αν δεν υπάρχει το τοποθετούμε στην πρώτη ελεύθερη θέση του O2.

Βήμα 4: Αυξάνουμε το i κατά 1, i=i+1. Αν i=N+1 ο αλγόριθμος τερματίζει. Αν όχι επιστρέφουμε στο Βήμα 3.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [1, 2, 3, 4, 5, 6, 7, 8] και P2 = [2, 4, 6, 8, 7, 5, 3, 1]. Οι τυχαίοι αριθμοί που παράγονται είναι οι 2, 3 και 6. Τοποθετούμε τα γονίδια που βρίσκονται σε αυτές τις θέσεις στον γονέα P1 και παίρνουμε τον απόγονο O1 = [_ , 2 , 3 , _ , _ , 6 , _ , _]. Ξεκινώντας από την αρχή του P2 συμπληρώνουμε τις κενές θέσεις του C1 με τα γονίδια του P2 που δεν υπάρχουν ήδη στον C1 και παίρνουμε τον απόγονο C1 = [4 , 2 , 3 , 8 , 7 , 6 , 5 , 1]. Με τον ίδιο τρόπο, αλλάζοντας τις θέσεις των P1, P2 παίρνουμε τον δεύτερο απόγονο C2 = [1 , 4 , 6 , 2 , 3 , 5 , 7 , 8].



Εικόνα 97 Position crossover

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.16 Sinusoidal Motion Crossover (SMX)

Ο συγκεκριμένος τελεστής διασταύρωσης προτάθηκε πρόσφατα από τους Varun Kumar S G1 , και Dr. R. Panneerselvam (Varun Kumar S G1, 2017) για την επίλυση του vehicle routing problem (VRP) που αποτελεί γενίκευση του TSP. Παράγει πάντα τους ίδιους απογόνους από τους ίδιους γονείς καθώς δεν κάνει χρήση τυχαίων αριθμών. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Ξεκινάμε από τον γονέα P1. Θέτουμε K=1 και αντιγράφουμε το γονίδιο που βρίσκεται στην θέση K στον απόγονο O1.

Βήμα 2ο: Αν το γονίδιο που βρίσκεται στη θέση K του P2 δεν υπάρχει στον O1 το αντιγράφουμε στην πρώτη ελεύθερη θέση του O1, αυξάνουμε το K κατά 1 και πηγαίνουμε στο Βήμα 3. Αν υπάρχει πηγαίνουμε στο Βήμα 4. Αν το K είναι μεγαλύτερο από N όπου N ο αριθμός των γονιδίων ο αλγόριθμος τερματίζει.

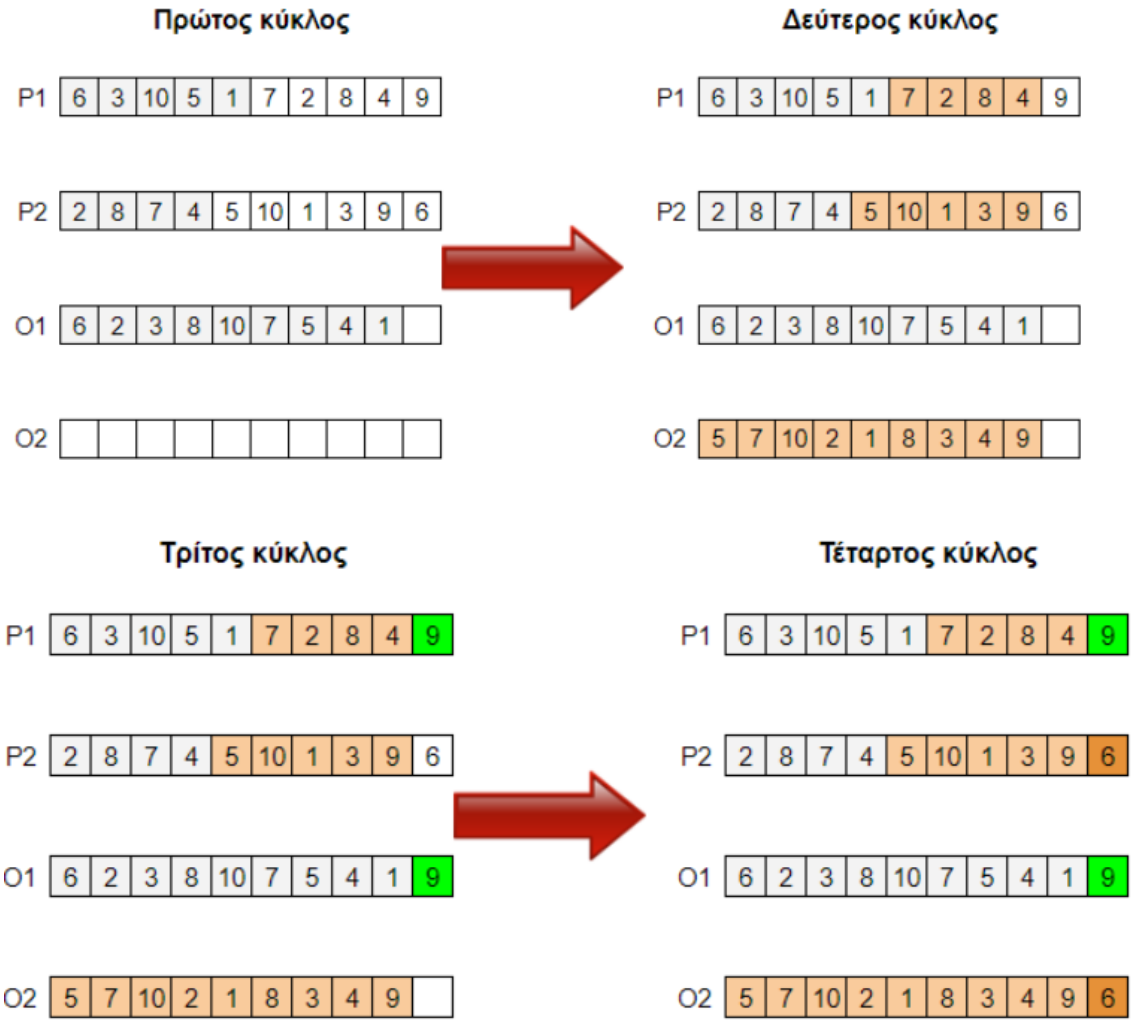
Βήμα 3ο: Αν το γονίδιο που βρίσκεται στη θέση K του P1 δεν υπάρχει στον O1 το αντιγράφουμε στην πρώτη ελεύθερη θέση του O1 και επιστρέφουμε στο Βήμα 2. Αν υπάρχει πηγαίνουμε στο Βήμα 5.

Βήμα 4ο: Αν το γονίδιο που βρίσκεται στη θέση K του P2 δεν υπάρχει στον O2 το αντιγράφουμε στην πρώτη ελεύθερη θέση του O2, αυξάνουμε το K κατά 1 και πηγαίνουμε στο Βήμα 5. Αν υπάρχει επιστρέφουμε στο Βήμα 2. Αν το K είναι μεγαλύτερο από N όπου N ο αριθμός των γονιδίων ο αλγόριθμος τερματίζει.

Βήμα 5ο: Αν το γονίδιο που βρίσκεται στη θέση Κ του P1 δεν υπάρχει στον O2 το αντιγράφουμε στην πρώτη ελεύθερη θέση του O2 και επιστρέφουμε στο Βήμα 4. Αν υπάρχει πηγαίνουμε στο Βήμα 3.

Παράδειγμα: Έστω οι γονείς P1 = [6, 3, 10, 5, 1, 7, 2, 8, 4, 9] και P2 = [2, 8, 7, 4, 5, 10, 1, 3, 9, 6] . Ξεκινάμε από τον P1 και τοποθετούμε το 6 στον O1. Πηγαίνουμε στην θέση 1 του P2. Το 2 δεν υπάρχει στον O1 οπότε το τοποθετούμε. Πηγαίνουμε στην επόμενη θέση του P1. Το 3 δεν υπάρχει στον O1 οπότε το τοποθετούμε. Επαναλαμβάνοντας την διαδικασία παίρνουμε τον απόγονο O1 = [6, 2, 3, 8, 10, 7, 5, 4, 1, _].

Το 5 υπάρχει στον O1 επομένως αρχίζουμε να τοποθετούμε γονίδια στον δεύτερο απόγονο O2 = [5, 7, 10, 2, 1, 8, 3, 4, 9, _]. Το 9 υπάρχει στον O2. Επιστρέφουμε στον O1. Τοποθετούμε το 9 και επιστρέφουμε στον O2 για να τοποθετήσουμε το 6. Οι τελικοί απόγονοι είναι O1 = [6 , 2 , 3 , 8 , 10 , 7, 5 , 4 , 1 , 9] και O2 = [5 , 7 , 10 , 2 , 1 , 8 , 3 , 4 , 9 , 6].



Εικόνα 98 Sinusoidal motion crossover

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.17 Sequential constructive crossover (SCX) - 1 offspring

Η πρώτη έκδοση του συγκεκριμένου τελεστή διασταύρωσης παρουσιάστηκε από τον Zakir, H. A. (2010) (Ahmed Z. H., 2010). Ο ίδιος μερικά χρόνια αργότερα παρουσίασε μια νέα έκδοση Zakir, H. A. (2013)

(Ahmed Z. H., 2011). Ο αλγόριθμος δημιουργήθηκε αποκλειστικά για το TSP. Στη συνέχεια θα δούμε αναλυτικά τον αλγόριθμο και τις διαφορές των δύο εκδόσεων.

Έστω ότι θέλουμε να κάνουμε διασταύρωση μεταξύ των χρωμοσωμάτων P1: (1, 5, 7, 3, 6, 4, 2) και P2: (1, 6, 2, 4, 3, 5, 7). Η απόσταση μεταξύ δύο σημείων δίνεται από τον ακόλουθο πίνακα.

Node	1	2	3	4	5	6	7
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

Εικόνα 99 Πίνακας κόστους

Βήμα πρώτο: Ξεκινάμε από το γονίδιο που βρίσκεται στην πρώτη θέση σε έναν από τους απογόνους. Ονομάζουμε αυτό το γονίδιο p και το προσθέτουμε στο χρωμόσωμα του απογόνου.

Βήμα δεύτερο: Από το γονίδιο p ψάχνουμε να βρούμε στο χρωμόσωμα και των δύο γονέων πιο γονίδιο είναι το επόμενο χωρίς αυτό να υπάρχει ήδη στο χρωμόσωμα του απογόνου. Σε αυτό το σημείο διαφέρουν οι δύο εκδόσεις του αλγορίθμου. Στην πρώτη έκδοση αν η αναζήτηση φτάσει στο τέλος του χρωμοσώματος χωρίς να βρει κάποιο γονίδιο τότε κάνουμε νέα αναζήτηση στη λίστα που περιέχει τις πόλεις (2,3,4,...,n). Στην δεύτερη έκδοση η αναζήτηση συνεχίζεται από την αρχή του χρωμοσώματος του γονέα.

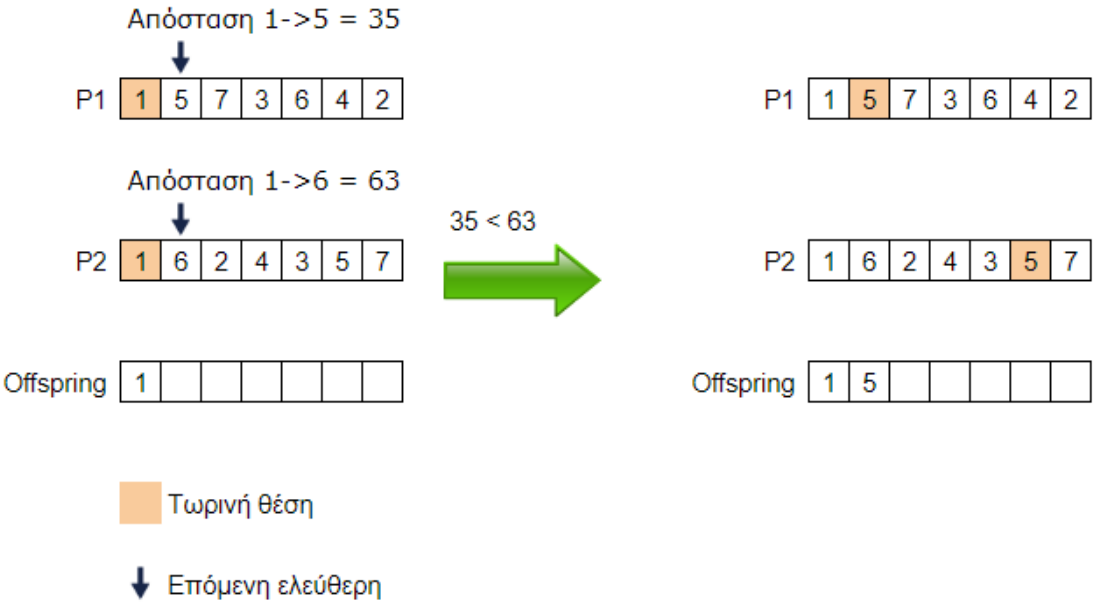
Βήμα τρίτο: Ονομάζουμε a και b τα γονίδια που βρέθηκαν από του προηγούμενο βήμα στους γονείς P1 και P2 αντίστοιχα.

Βήμα τέταρτο: Υπολογίζουμε τις αποστάσεις C_{pa} και C_{pb}. Αν $C_{pa} < C_{pb}$ προσθέτουμε το γονίδιο a στο χρωμόσωμα του απογόνου και θέτουμε p=a. Αλλιώς προσθέτουμε το γονίδιο b στο χρωμόσωμα του απογόνου και θέτουμε p=b. Αν το χρωμόσωμα του απογόνου έχει ολοκληρωθεί ο αλγόριθμος σταματάει. Αν όχι επιστρέφουμε στο βήμα 2.

Ποιο αναλυτικά έστω ότι έχουμε τα χρωμοσώματα P1: (1, 5, 7, 3, 6, 4, 2) και P2: (1, 6, 2, 4, 3, 5, 7) με μήκος διαδρομής 312 και 335 αντίστοιχα.

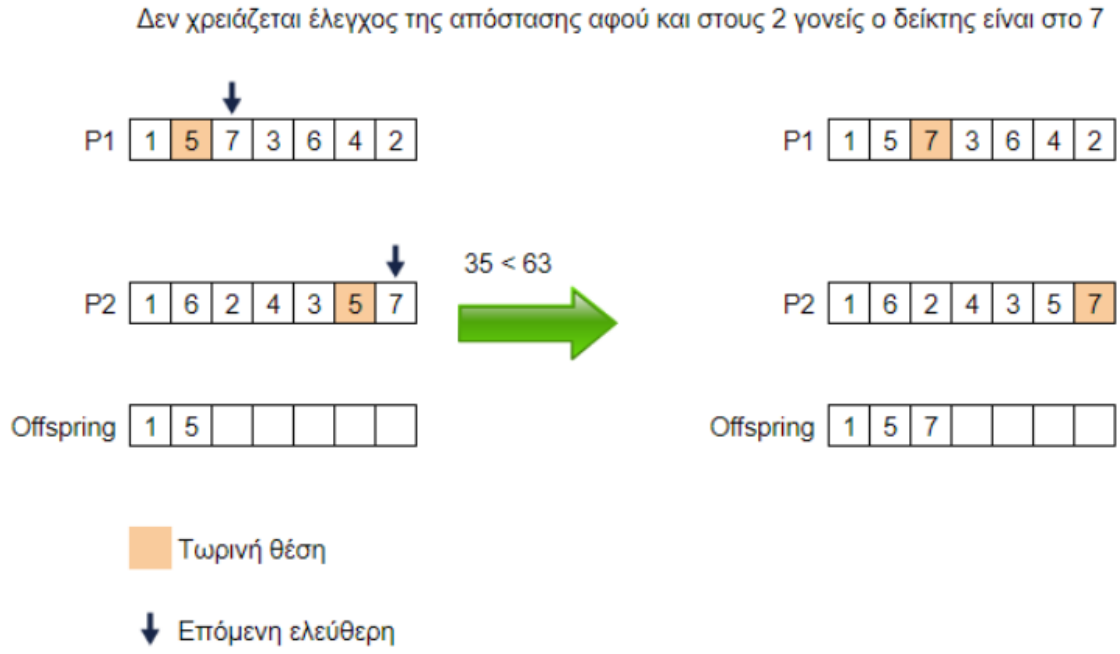
7.17.1 Πρώτη έκδοση του αλγορίθμου

Επιλέγουμε για αρχή το γονίδιο 1 (p=1). Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 6. Από τον πίνακα έχουμε C_{15} = 35 και C_{16} = 63. Επειδή $C_{15} < C_{16}$ έχουμε p=5 και εισάγουμε το 5 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5).



Εικόνα 100 Πρώτο βήμα αλγορίθμου

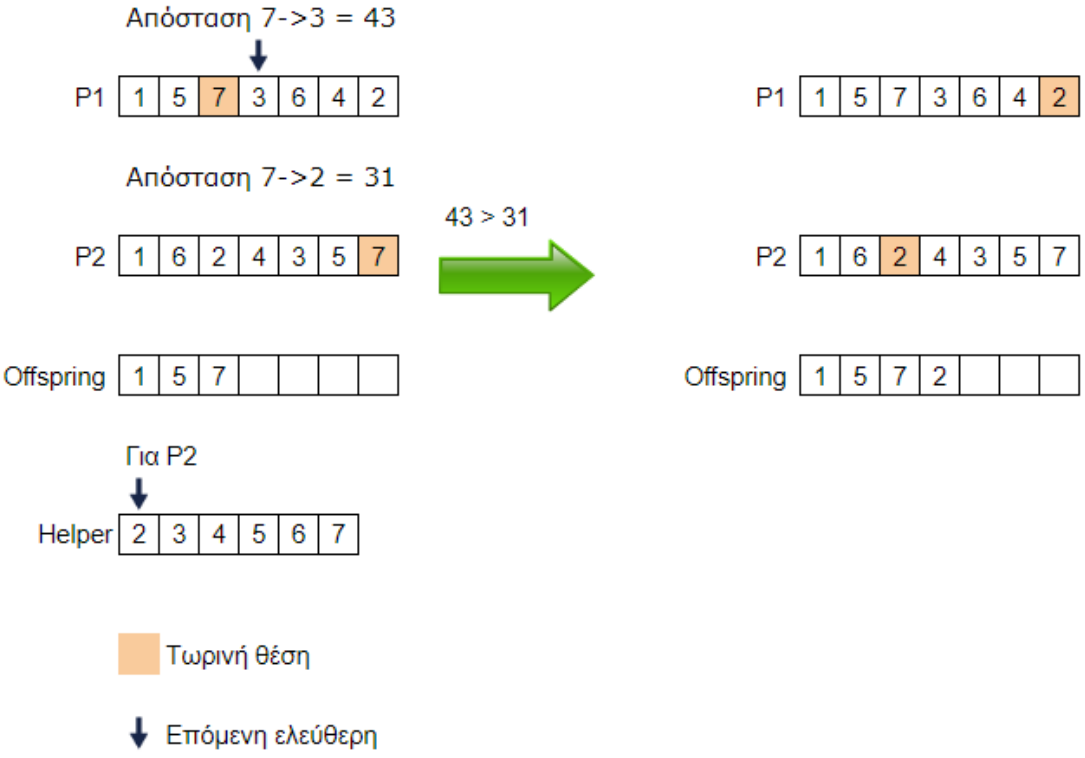
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 5 είναι το 7. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 είναι το 7. Οπότε έχουμε p=7 και εισάγουμε το 7 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7).



Εικόνα 101 Δεύτερο βήμα αλγορίθμου

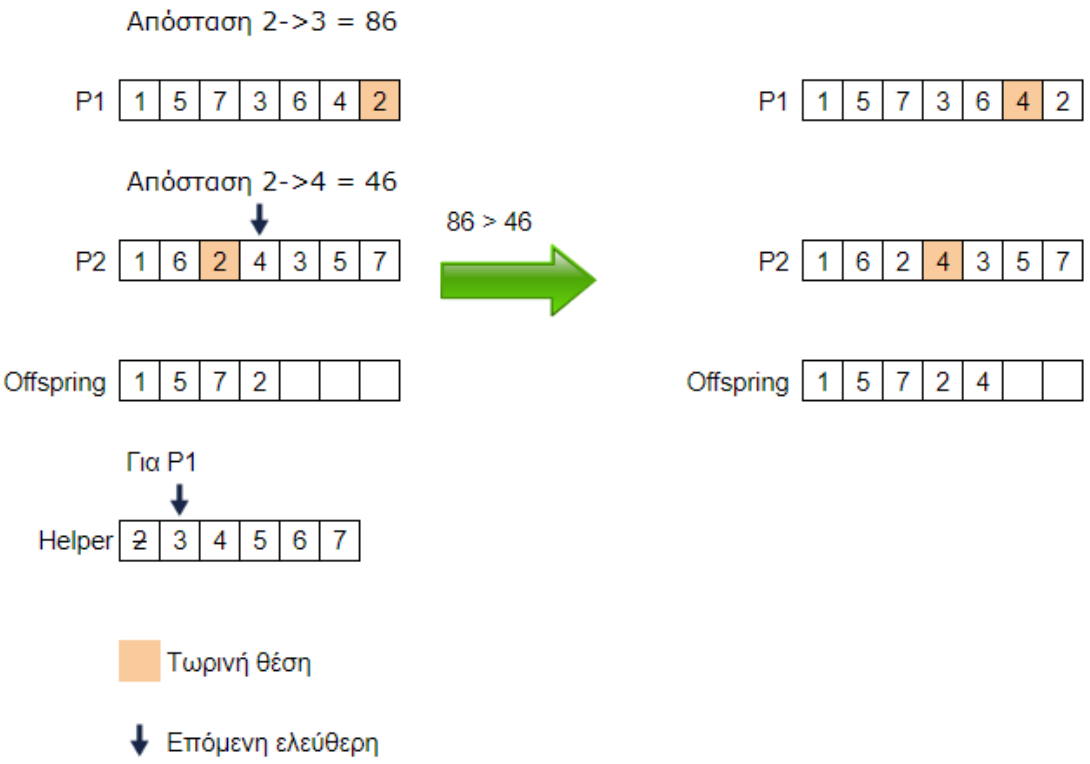
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 7 είναι το 3. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο καθώς βρισκόμαστε στο τέλος του χρωμοσώματος. Οπότε θα πρέπει να βρούμε το πρώτο διαθέσιμο γονίδιο στη λίστα (2,3,4,5,6,7) που είναι το 2. Από τον πίνακα έχουμε C_{73} =

43, C_{72} = 31. Επειδή $C_{72} < C_{73}$ έχουμε p=2 και εισάγουμε το 2 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,2).



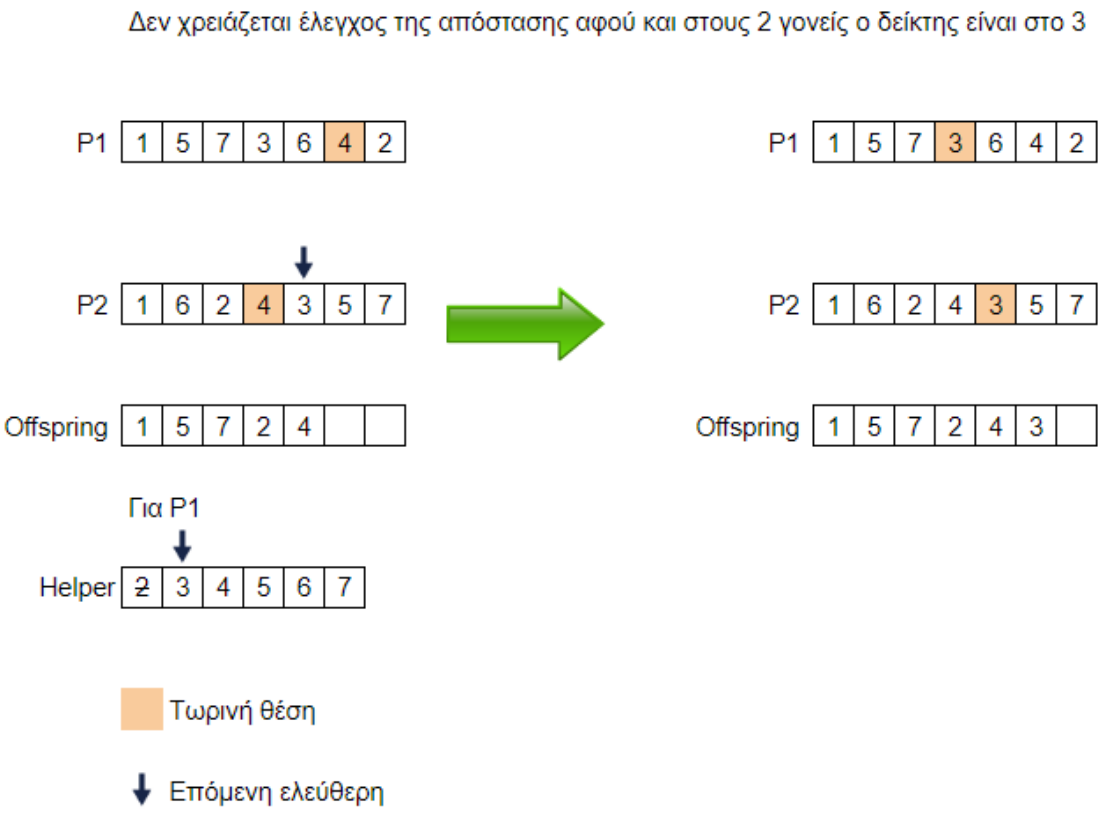
Εικόνα 102 Τρίτο βήμα αλγορίθμου

Για p=2 δεν υπάρχει διαθέσιμο γονίδιο για τον γονέα P1.Το πρώτο διαθέσιμο γονίδιο στη λίστα (2, 3, 4, 5, 6, 7) για τον P1 είναι το 3. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο είναι το 4. Από τον πίνακα έχουμε C_{24} = 46 και C_{23} = 86. Επειδή $C_{24} < C_{23}$ έχουμε p=4 και εισάγουμε το 4 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,2,4).



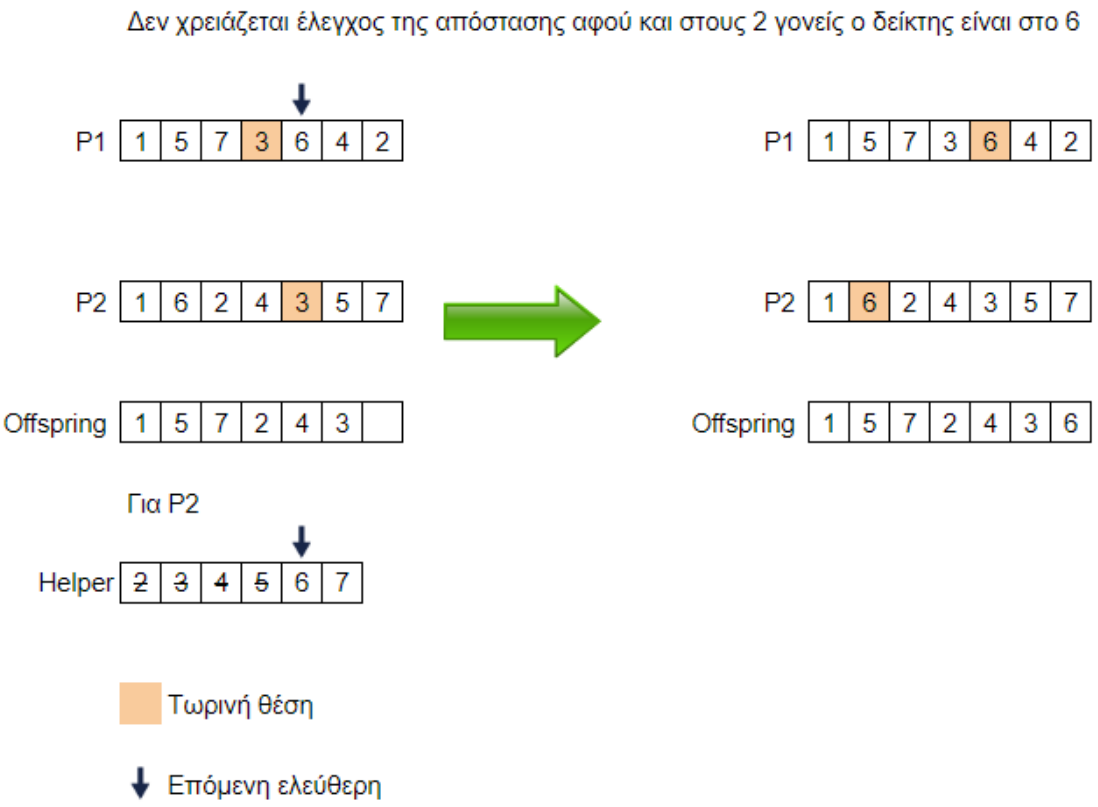
Εικόνα 103 Τέταρτο βήμα αλγορίθμου

Για p=4 δεν υπάρχει διαθέσιμο γονίδιο για τον P1 οπότε ψάχνουμε στη λίστα (2, 3, 4, 5, 6, 7) και βρίσκουμε ότι το πρώτο διαθέσιμο γονίδιο είναι το 3. Το πρώτο διαθέσιμο γονίδιο για τον P2 είναι το 3. Οπότε έχουμε p=3 και εισάγουμε το 3 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,2,4,3).



Εικόνα 104 Πέμπτο βήμα αλγορίθμου

Για p=3 το πρώτο διαθέσιμο γονίδιο για τον P1 είναι το 6. Για τον P2 το πρώτο διαθέσιμο γονίδιο είναι το 6. Οπότε έχουμε p=6 και εισάγουμε το 6 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,2,4,3,6).

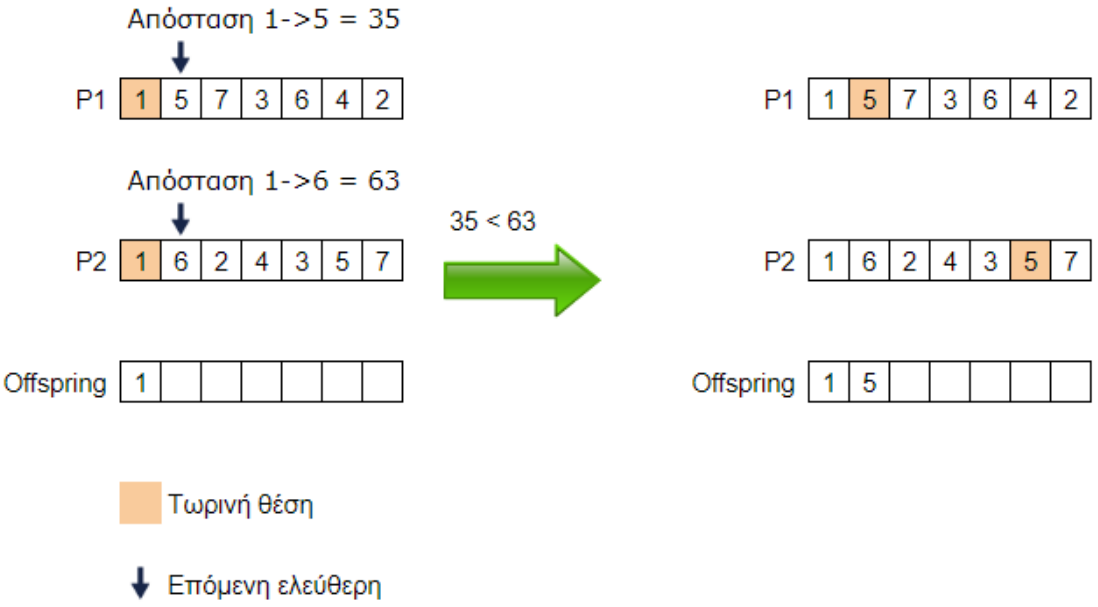


Εικόνα 105 Έκτο βήμα αλγορίθμου

Ο αλγόριθμος έχει τελειώσει και το μήκος του απογόνου (1,5,7,2,4,3,6) είναι 266, πήραμε δηλαδή μικρότερη διαδρομή και από τους δύο γονείς.

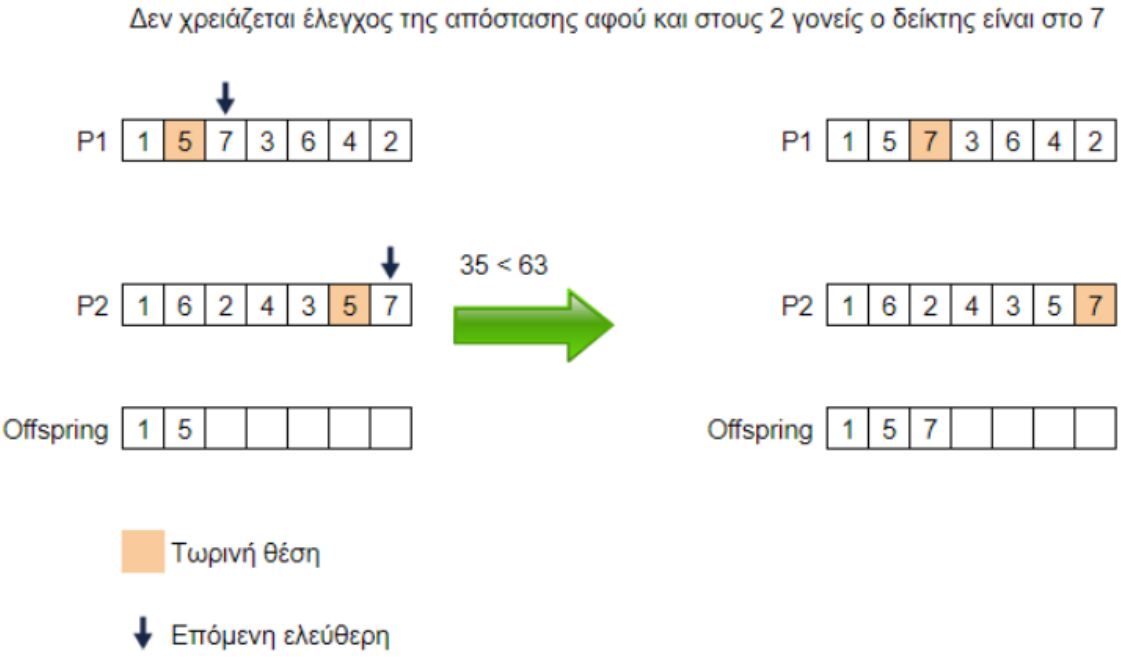
7.17.2 Δεύτερη έκδοση αλγορίθμου

Επιλέγουμε για αρχή το γονίδιο 1 (p=1). Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 6. Από τον πίνακα έχουμε C_{15} = 35 και C_{16} = 63. Επειδή $C_{15} < C_{16}$ έχουμε p=5 και εισάγουμε το 5 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5).



Εικόνα 106 Πρώτο βήμα αλγορίθμου

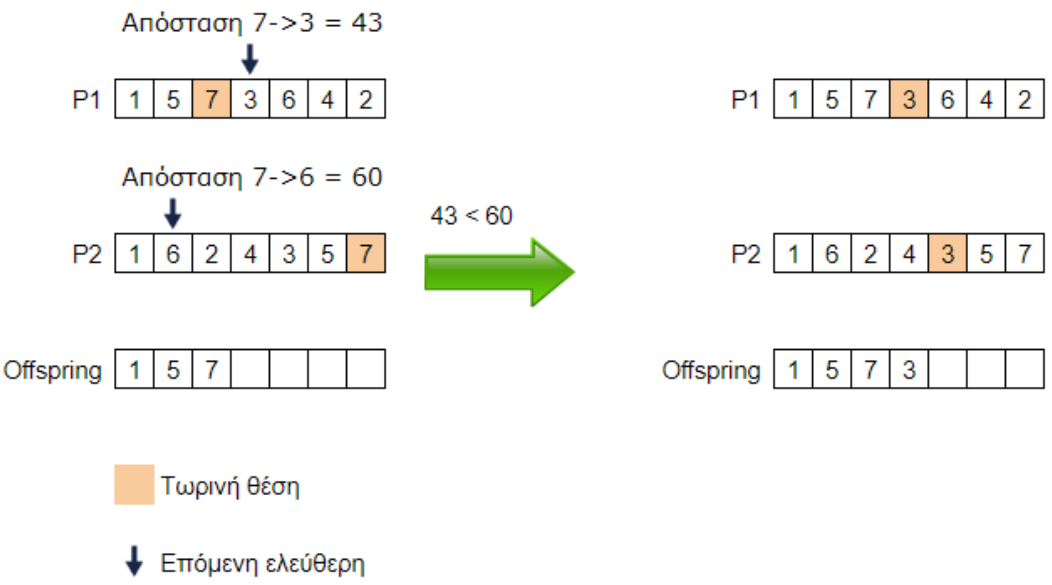
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 5 είναι το 7. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 είναι το 7. Οπότε έχουμε p=7 και εισάγουμε το 7 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7).



Εικόνα 107 Δεύτερο βήμα αλγορίθμου

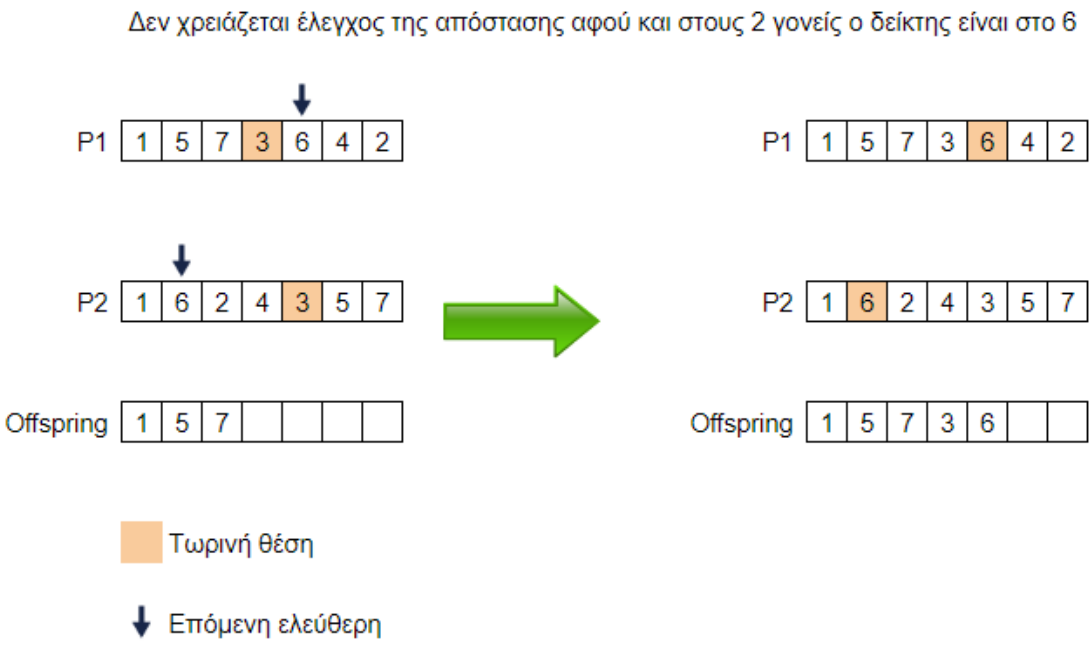
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 7 είναι το 3. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο καθώς βρισκόμαστε στο τέλος του χρωμοσώματος. Οπότε συνεχίζουμε την αναζήτηση από την αρχή του γονέα P2 και παίρνουμε το πρώτο διαθέσιμο γονίδιο 6. Από τον πίνακα

έχουμε C_{73}= 43 και C_{76} = 60. Επειδή $C_{73} < C_{76}$ έχουμε p=3 και εισάγουμε το 3 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3).



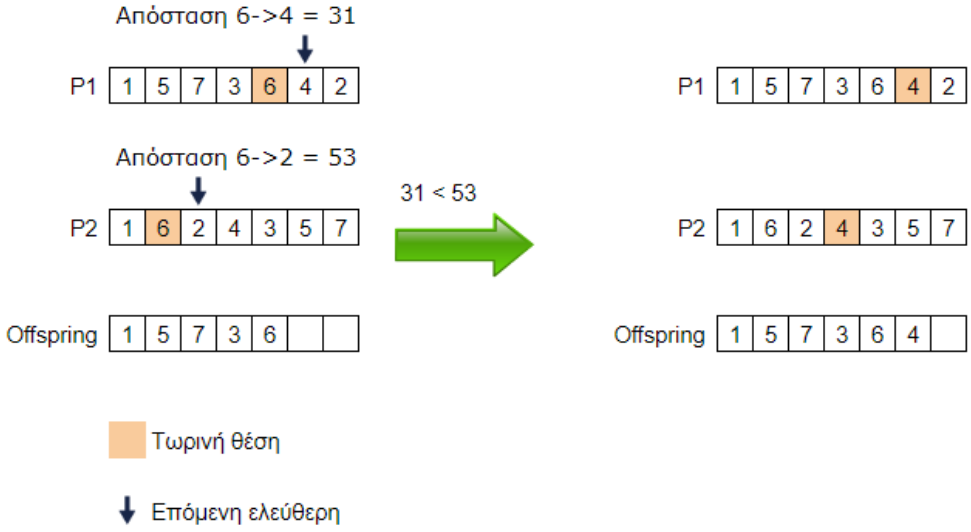
Εικόνα 108 Τρίτο βήμα αλγορίθμου

Το πρώτο διαθέσιμο γονίδιο για τον P1 από το γονίδιο 3 είναι το 6. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο οπότε πρέπει να ψάξουμε στη λίστα (1, 6, 2, 4) η οποία μας δίνει ότι το διαθέσιμο γονίδιο για τον P2 είναι το γονίδιο 6. Οπότε έχουμε p=6 και εισάγουμε το 6 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,6).



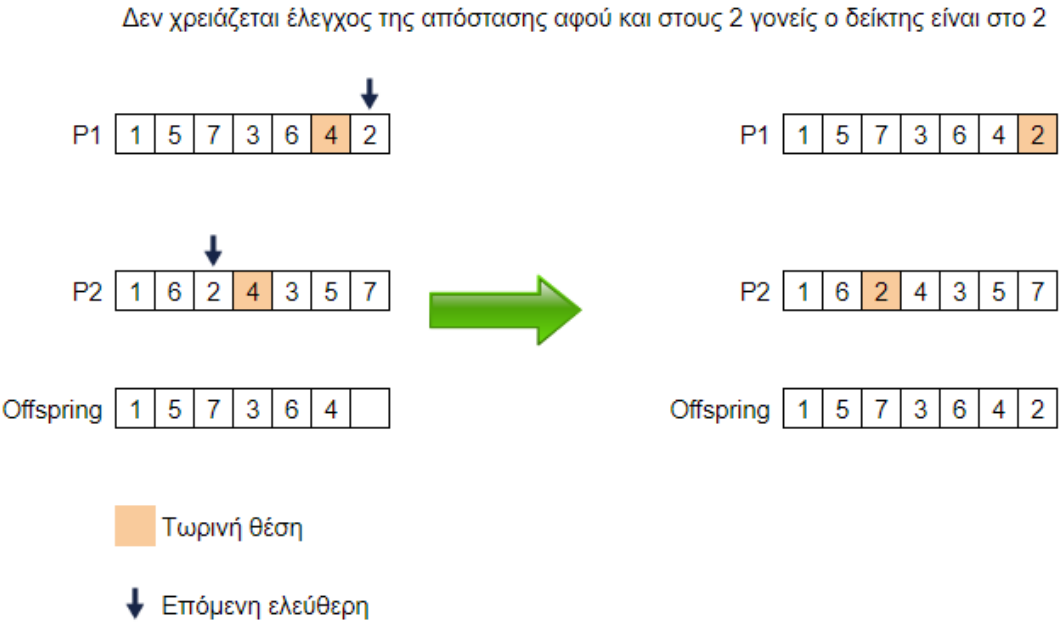
Εικόνα 109 Τέταρτο βήμα αλγορίθμου

Το πρώτο διαθέσιμο γονίδιο για τον P1 από το γονίδιο 6 είναι το 4. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 είναι το 2. Από τον πίνακα έχουμε C_{64}= 31 και C_{62} = 53. Επειδή C_{64}< C_{62} έχουμε p=4 και εισάγουμε το 4 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,6,4).



Εικόνα 110 Πέμπτο βήμα αλγορίθμου

Για p=4 το πρώτο διαθέσιμο γονίδιο για τον P1 είναι το 2. Για τον P2 το πρώτο διαθέσιμο γονίδιο είναι το 2. Οπότε έχουμε p=2 και εισάγουμε το 2 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,6,4,2).



Εικόνα 111 Έκτο βήμα αλγορίθμου

Ο αλγόριθμος έχει τελειώσει και το μήκος του απογόνου (1,5,7,3,6,4,2) είναι 312, δηλαδή χειρότερο από την 1ή έκδοση. Αν όμως πάρουμε γονείς P1 = (1,4,6,3,5,2,7) και P2 = (1,2,7,5,3,6,4) με

αποδόσεις 320 και 266 αντίστοιχα παίρνουμε τον απόγονο (1,4,2,7,3,5,6) με απόδοση 257 με την 1η έκδοση και τον απόγονο (1,4,2,7,5,3,6) με απόδοση 230 με τη δεύτερη έκδοση.

Σημείωση 1: Στο παράδειγμα που παρουσιάσαμε κάναμε την υπόθεση ότι οι δύο γονείς ξεκινάνε από την ίδια πόλη. Κάτι τέτοιο προφανώς δεν συμβαίνει τις περισσότερες φορές, αλλά επειδή η κάθε διαδρομή είναι ένας κύκλος hamilton, μπορούμε να υποθέσουμε ότι ισχύει, αφού μπορούμε να κάνουμε τον δεύτερο γονέα να ξεκινάει από το ίδιο γονίδιο.

Σημείωση 2: Έρευνα έδειξε ότι έχουμε καλύτερα αποτελέσματα αν δεν ξεκινάμε πάντα από το πρώτο γονίδιο του πρώτου γονέα, αλλά από ένα τυχαίο.

Η υλοποίηση του αλγορίθμου σε php για την 1η έκδοση βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου σε php για την 2η έκδοση βρίσκεται εδώ.

7.18 Enhanced Sequential Constructive Crossover (ESCX)

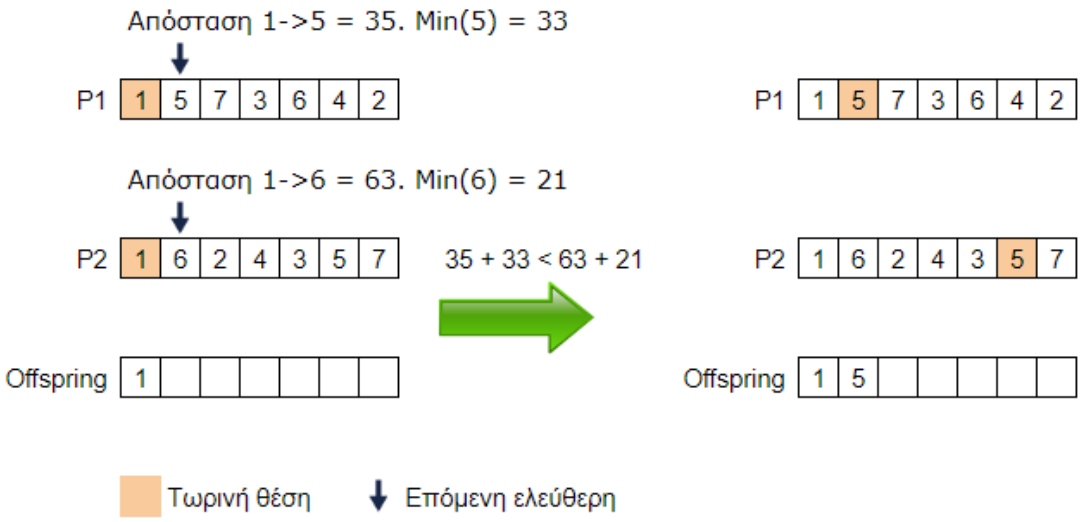
Επειδή ο τελεστής διασταύρωσης SCX είναι ένας από τους καλύτερους τελεστές για την επίλυση του TSP πολλοί προσπάθησαν να τον βελτιώσουν. Μία από αυτές τις "βελτιώσεις" / παραλλαγές είναι ο αλγόριθμος ESCX (Alanzi, 2017). Ο αλγόριθμος λειτουργεί όπως ακριβώς και ο SCX μόνο που όταν φτάνει στο στάδιο της σύγκρισης των αποστάσεων προσθέτει και το ελάχιστο της απόστασης του υποψήφιου γονίδιου από όλα τα άλλα γονίδια που δεν υπάρχουν ήδη στον απόγονο.

Για παράδειγμα έστω ότι θέλουμε να κάνουμε διασταύρωση μεταξύ των χρωμοσωμάτων P1: (1, 5, 7, 3, 6, 4, 2) και P2: (1, 6, 2, 4, 3, 5, 7). Η απόσταση μεταξύ δύο σημείων δίνεται από τον ακόλουθο πίνακα.

Node	1	2	3	4	5	6	7
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

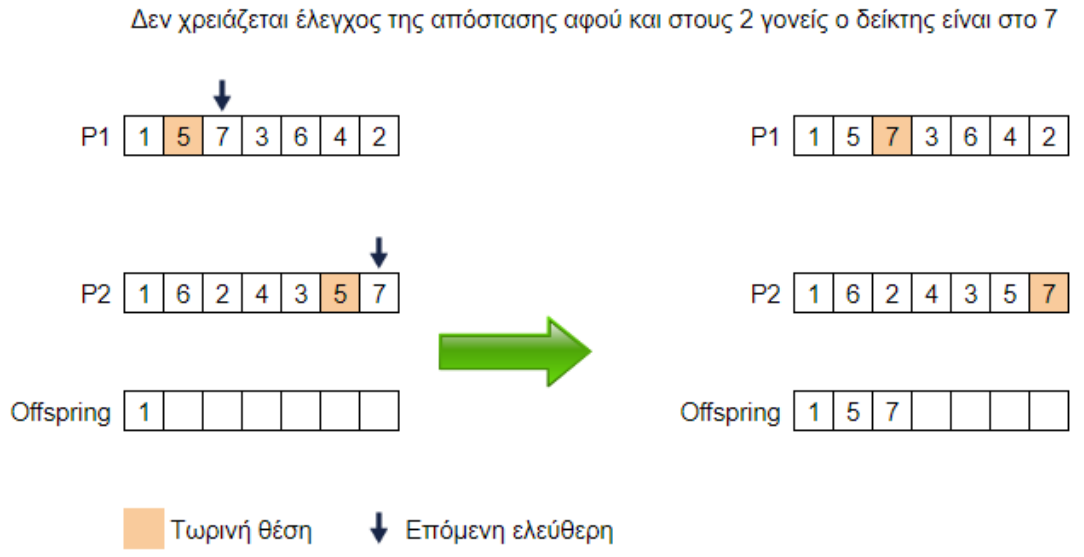
Εικόνα 112 Πίνακας κόστους

Επιλέγουμε για αρχή το γονίδιο 1 (p=1). Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 είναι το 6. Από τον πίνακα έχουμε C_{15}= 35 και C_{16} = 63. Οι αποστάσεις του γονιδίου 5 με όλα τα άλλα γονίδια εκτός από το 1 είναι 63, 33, 65, 999, 76, και 72 με την ελάχιστη τιμή να είναι το $min_5 = 33$. Οι αποστάσεις του γονιδίου 6 με όλα τα άλλα γονίδια εκτός από το 1 είναι 53, 89, 31, 21, 999, και 52 με την ελάχιστη τιμή να είναι το $min_6 = 21$. Επειδή $C_{15} + min_5 < C_{16} + min_6$ έχουμε p=5 και εισάγουμε το 5 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5).



Εικόνα 113 Πρώτο βήμα αλγορίθμου

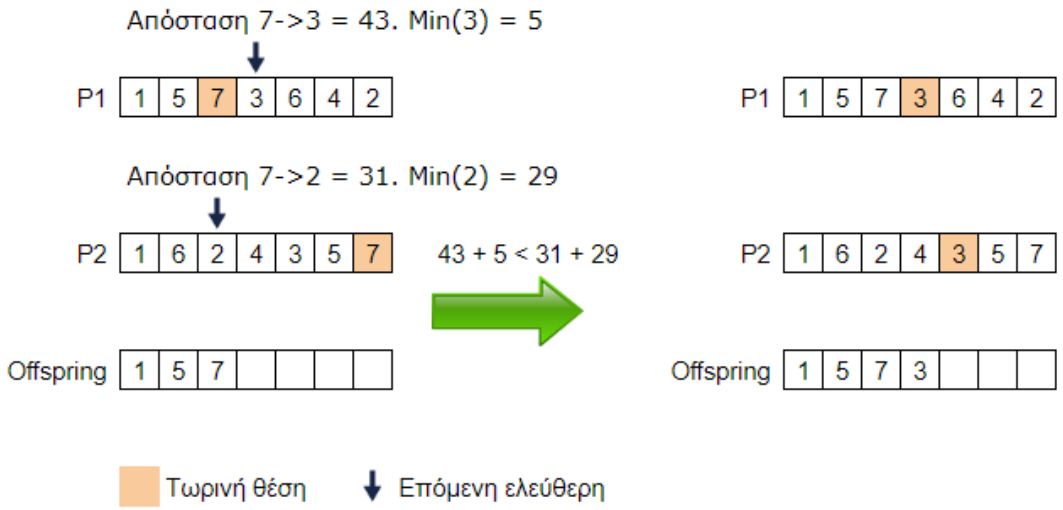
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 5 είναι το 7. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 είναι το 7. Οπότε έχουμε p=7 και εισάγουμε το 5 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7).



Εικόνα 114 Δεύτερο βήμα αλγορίθμου

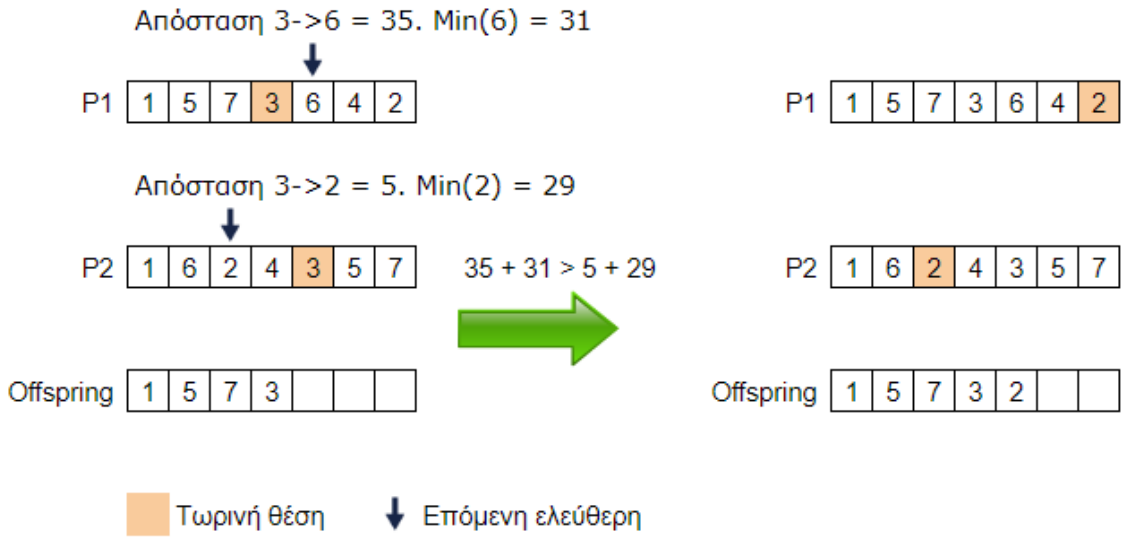
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 7 είναι το 3. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο καθώς βρισκόμαστε στο τέλος του χρωμοσώματος. Οπότε θα πρέπει να βρούμε το πρώτο διαθέσιμο γονίδιο στη λίστα (2,3,4,5,6,7) που είναι το 2. Από τον πίνακα έχουμε C_{73}= 43, C_{72} = 31. Οι αποστάσεις του γονιδίου 3 με όλα τα άλλα γονίδια εκτός από τα 1,5 και 7 είναι 5, 999, 16, και 35 με την ελάχιστη τιμή να είναι το $min_3 = 5$. Οι αποστάσεις του γονιδίου 2 με όλα τα άλλα γονίδια εκτός από τα 1,5 και 7 είναι 999, 86, 46, και 29 με την ελάχιστη τιμή να είναι το $min_2 = 29$.

Επειδή $C_{72} + min_2 > C_{73} + min_3$ έχουμε p=3 και εισάγουμε το 3 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3).



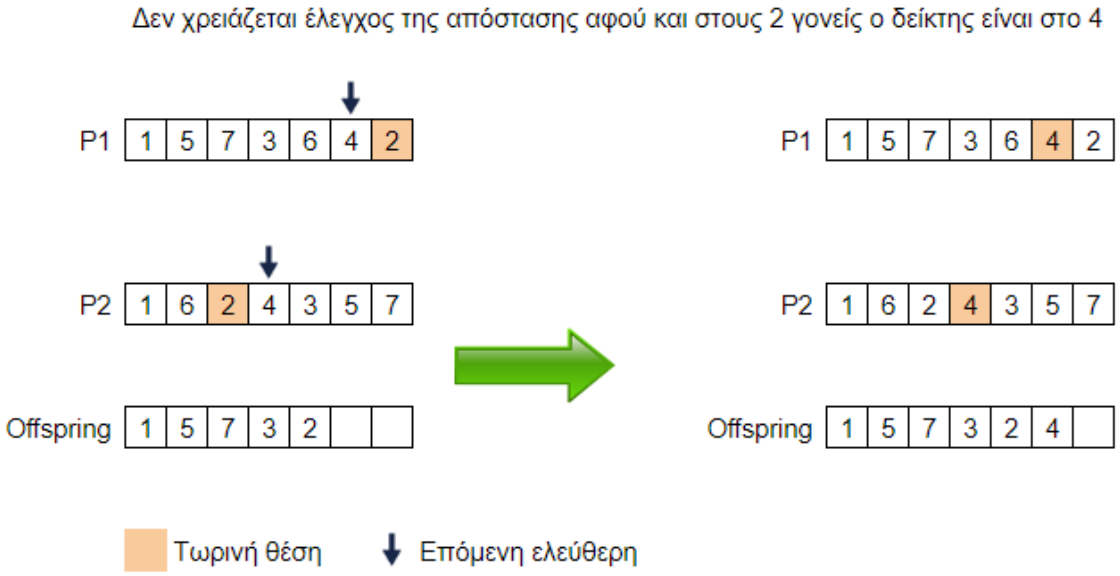
Εικόνα 115 Τρίτο βήμα αλγορίθμου

Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 3 είναι το 6. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο άρα ελέγχουμε τη λίστα (2,3,4,5,6,7) και βρίσκουμε το 2. Από τον πίνακα έχουμε C_{36}= 35, C_{32}= 5. Οι αποστάσεις του γονιδίου 6 με όλα τα άλλα γονίδια εκτός από τα 1, 3, 5 και 7 είναι 53, 31, και 999 με την ελάχιστη τιμή να είναι το $min_6 = 31$. Οι αποστάσεις του γονιδίου 2 με όλα τα άλλα γονίδια εκτός από τα 1, 3, 5 και 7 είναι 999, 46, και 29 με την ελάχιστη τιμή να είναι το $min_2 = 29$. Επειδή $C_{32} + min_2 < C_{36} + min_6$ έχουμε p=2 και εισάγουμε το 2 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,2).



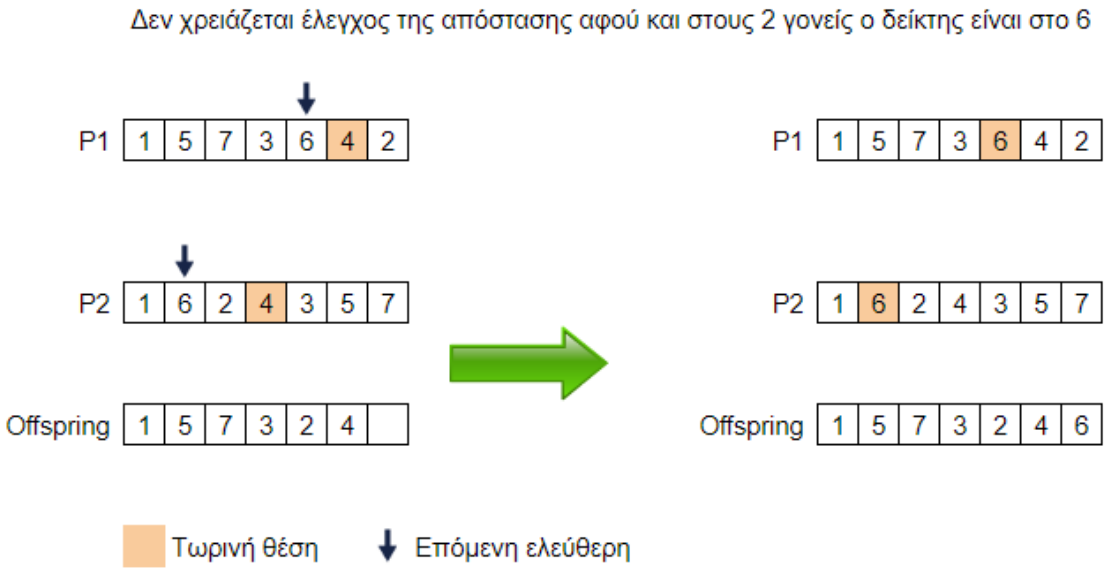
Εικόνα 116 Τέταρτο βήμα αλγορίθμου

Για τον γονέα P1 δεν υπάρχει διαθέσιμο γονίδιο από το γονίδιο 2 άρα ελέγχουμε τη λίστα (2,3,4,5,6,7) και βρίσκουμε το 4. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 2 είναι το 4. Οπότε έχουμε p=4 και εισάγουμε το 4 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,2,4).



Εικόνα 117 Πέμπτο βήμα αλγορίθμου

Για τον γονέα P1 δεν υπάρχει διαθέσιμο γονίδιο από το γονίδιο 4 άρα ελέγχουμε τη λίστα (2,3,4,5,6,7) και βρίσκουμε το 6. Για τον γονέα P2 δεν υπάρχει διαθέσιμο γονίδιο άρα ελέγχουμε τη λίστα (2,3,4,5,6,7) και βρίσκουμε το 6. Οπότε έχουμε p=6 και εισάγουμε το 6 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,5,7,3,2,4).



Εικόνα 118 Έκτο βήμα αλγορίθμου

Σημείωση: Στο paper που δημοσιεύτηκε ο αλγόριθμος υπάρχει ένα λάθος στη σύγκριση που κάνει με τον SCX. Στο παράδειγμα του ESCX χρησιμοποιεί την πρώτη έκδοση του SCX δηλαδή ελέγχει τη λίστα (2,3,4,5,6,7) και βρίσκει ότι η διαδρομή που παράγει ο ESCX με γονείς P1: (1, 5, 7, 3, 6, 4, 2) και P2: (1, 6, 2, 4, 3, 5, 7) έχει μήκος 290. Στη συνέχεια αναφέρει ότι ο SCX με τους ίδιους γονείς παράγει διαδρομή μήκους 312. Κάτι τέτοιο ισχύει όμως στην περίπτωση που χρησιμοποιήσουμε την δεύτερη έκδοση του SCX καθώς όπως είδαμε η πρώτη μας δίνει διαδρομή μήκους 266. Για αυτό τον λόγο η υλοποίηση του ESCX έγινε και με τις δύο εκδόσεις του SCX.

Σημείωση: Παρόλο που σαν ιδέα φαίνεται καλή, στην πράξη ο ESCX δε δίνει καλύτερα αποτελέσματα (Hommadi, 2018) από τον SCX και απαιτεί πολύ μεγαλύτερη υπολογιστική ισχύ. Για αυτό το λόγο δεν χρησιμοποιείται.

Η υλοποίηση του αλγορίθμου σε php για την 1η έκδοση βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου σε php για την 2η έκδοση βρίσκεται εδώ.

7.19 Bidirectional Circular Sequential Constructive Crossover

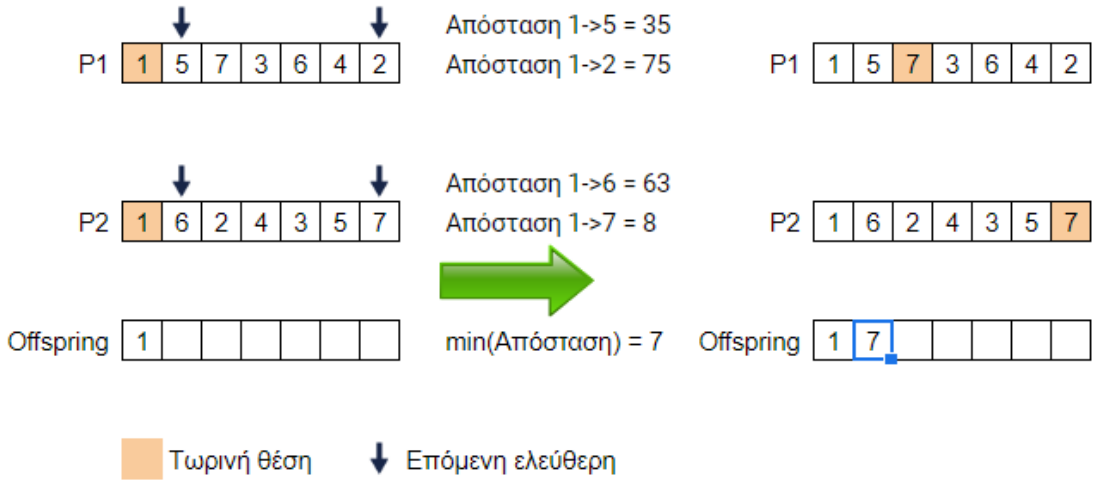
Ο αλγόριθμος BCSCX (S. Kang) αποτελεί παραλλαγή του αλγορίθμου sequential constructive crossover και πιο συγκεκριμένα της δεύτερης έκδοσής του. Η μόνη διαφορά με τον SCX είναι ότι παίρνουμε 4 συνολικά υποψήφια γονίδια (δύο για κάθε γονέα) αφού κάνουμε έλεγχο και προς τα αριστερά αλλά και προς τα δεξιά από το γονίδιο στο οποίο βρισκόμαστε.

Για παράδειγμα έστω ότι θέλουμε να κάνουμε διασταύρωση μεταξύ των χρωμοσωμάτων P1: (1, 5, 7, 3, 6, 4, 2) και P2: (1, 6, 2, 4, 3, 5, 7). Η απόσταση μεταξύ δύο σημείων δίνεται από τον ακόλουθο πίνακα.

Node	1	2	3	4	5	6	7
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

Εικόνα 119 Πίνακας κόστους

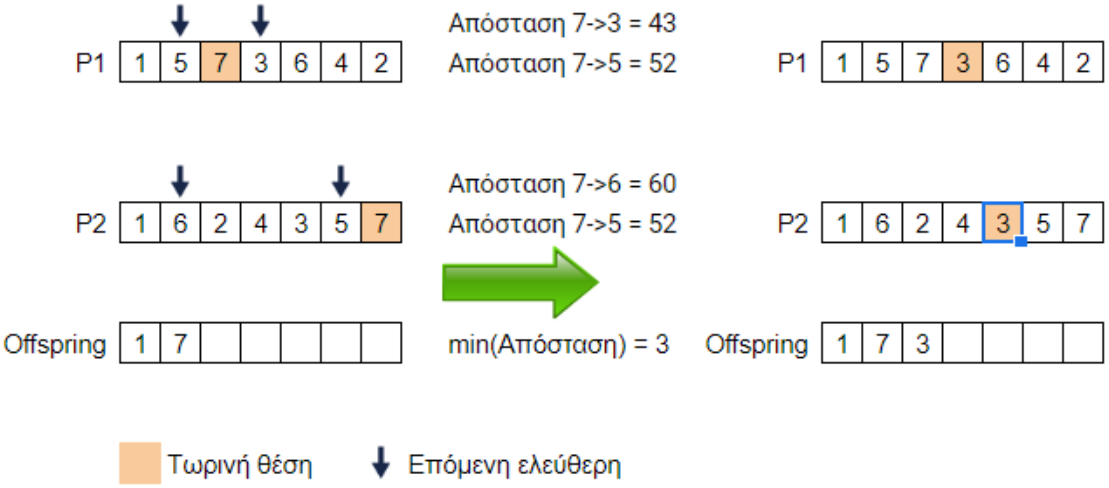
Επιλέγουμε για αρχή το γονίδιο 1 (p=1). Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 προς τα δεξιά είναι το 5 και προς τα αριστερά το 2. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 1 προς τα δεξιά είναι το 6 και προς τα αριστερά το 7. Από τον πίνακα έχουμε C_{15}= 35, C_{16} = 63, C_{12}= 75, C_{17}= 8. Επειδή το μικρότερο είναι το C_{17} εισάγουμε το 7 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7) και θέτουμε p=7.



Εικόνα 120 Πρώτο βήμα αλγορίθμου

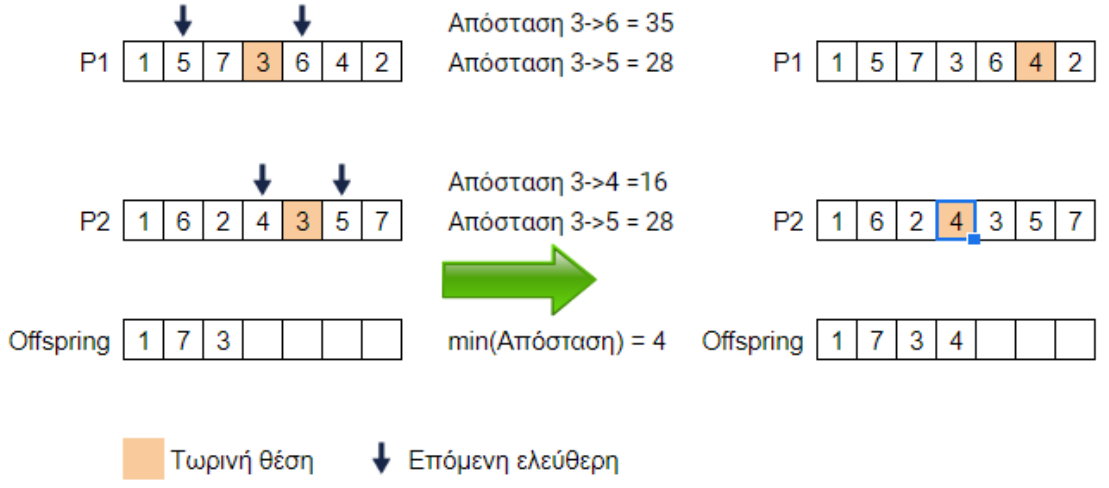
Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 7 προς τα δεξιά είναι το 3 και προς τα αριστερά το 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 7 προς τα δεξιά είναι το

6 και προς τα αριστερά το 5. Από τον πίνακα έχουμε C_{73}= 43, C_{75}= 52, C_{76}= 60. Επειδή το μικρότερο είναι το C_{73} εισάγουμε το 3 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7,3) και θέτουμε p=3.



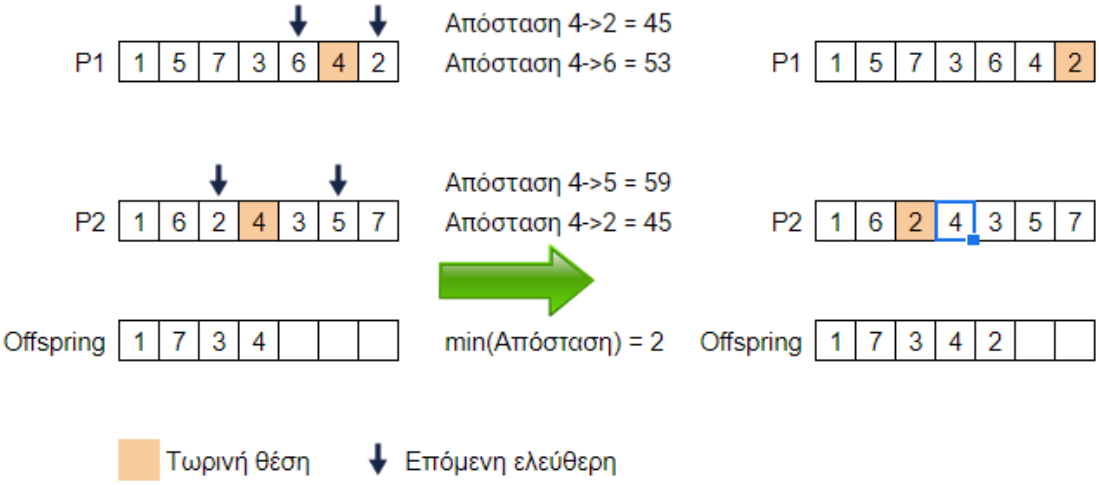
Εικόνα 121 Δεύτερο βήμα αλγορίθμου

Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 3 προς τα δεξιά είναι το 6 και προς τα αριστερά το 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 3 προς τα δεξιά είναι το 5 και προς τα αριστερά το 4. Από τον πίνακα έχουμε C_{34}= 16, C_{35} = 28, C_{36}= 35. Επειδή το μικρότερο είναι το C_{34} εισάγουμε το 4 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7,3,4) και θέτουμε p=4.



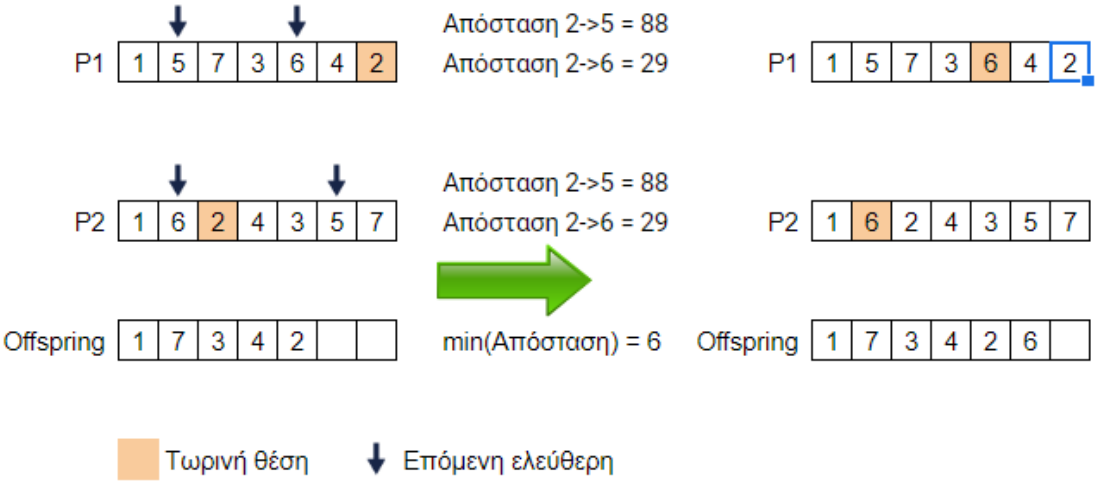
Εικόνα 122 Τρίτο βήμα αλγορίθμου

Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 4 προς τα δεξιά είναι το 2 και προς τα αριστερά το 6. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 4 προς τα δεξιά είναι το 5 και προς τα αριστερά το 2. Από τον πίνακα έχουμε C_{42}= 45, C_{45}= 59, C_{46}= 53. Επειδή το μικρότερο είναι το C_{42} εισάγουμε το 2 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7,3,4,2) και θέτουμε p=2.



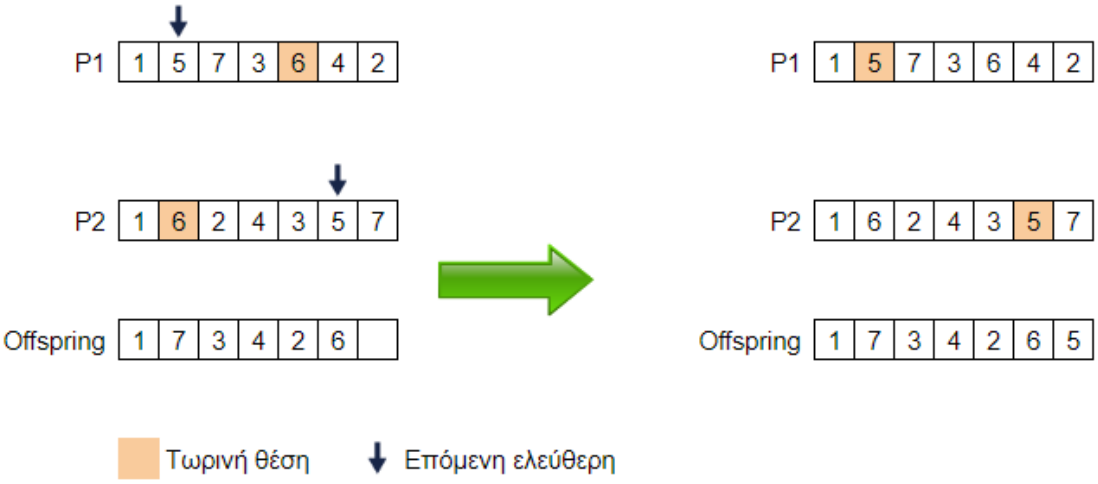
Εικόνα 123 Τέταρτο βήμα αλγορίθμου

Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 2 προς τα δεξιά είναι το 5 και προς τα αριστερά το 6. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 3 προς τα δεξιά είναι το 5 και προς τα αριστερά το 6. Από τον πίνακα έχουμε C_{25} = 88, C_{26} = 29. Επειδή το μικρότερο είναι το C_{26} εισάγουμε το 6 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7,3,4,2,6) και θέτουμε p=6.



Εικόνα 124 Πέμπτο βήμα αλγορίθμου

Για τον γονέα P1 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 και προς τις δύο κατευθύνσεις είναι το γονίδιο 5. Για τον γονέα P2 το πρώτο διαθέσιμο γονίδιο από το γονίδιο 6 και προς τις δύο κατευθύνσεις είναι το 5. Οπότε εισάγουμε το 5 στο χρωμόσωμα του απογόνου το οποίο γίνεται (1,7,3,4,2,6,5).



Εικόνα 125 Έκτο βήμα αλγορίθμου

Η τελική διαδρομή είναι (1,7,3,4,2,6,5) και έχει μήκος 248, που είναι μικρότερο από το μήκος που μας έδωσαν και οι SCX και ο ESCX. Όπως και στον SCX έτσι και στον BCSCX έχουμε καλύτερα αποτελέσματα (Hommadi, 2018) αν δεν ξεκινάμε πάντα από το πρώτο γονίδιο του πρώτου γονέα, αλλά από ένα τυχαίο.

Η υλοποίηση του αλγορίθμου σε php για την 2η έκδοση βρίσκεται εδώ.

7.20 Κυκλική διασταύρωση - Cycle crossover (CX)

Ο αλγόριθμος κυκλικής διασταύρωσης δημιουργήθηκε από τον Oliver et al (Oliver, 1987)για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση μετάθεσης. Ο αλγόριθμος δημιουργεί τους απογόνους κάνοντας κύκλους. Σε κάθε κύκλο τοποθετούνται στον απόγονο γονίδια από τον ένα μόνο γονέα. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Η υλοποίηση του αλγορίθμου περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Αναζητούμε την πρώτη ελεύθερη θέση στον απόγονο O1 και θέτουμε i αυτή τη θέση. Προφανώς στην αρχή i=1.

Βήμα 2ο: Τοποθετούμε το γονίδιο που βρίσκεται στη θέση i του γονέα P1 στη θέση i του απόγονου O1. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει.

Βήμα 3ο: Αν το γονίδιο που βρίσκεται στη θέση i του γονέα P2 υπάρχει στον απόγονο O1 πηγαίνουμε στο Βήμα 5. Αναζητούμε το γονίδιο που βρίσκεται στη θέση i του γονέα P2 στο γονέα P1(έστω ότι το βρίσκουμε στη θέση j).

Βήμα 4ο: Τοποθετούμε το γονίδιο που βρίσκεται στη θέση j του γονέα P1 στη θέση j του O1. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει. Θέτουμε i=j και επιστρέφουμε στο Βήμα 2.

Βήμα 5ο: Αναζητούμε την πρώτη ελεύθερη θέση στον απόγονο O1 και θέτουμε i αυτή τη θέση.

Βήμα 6ο: Τοποθετούμε το γονίδιο που βρίσκεται στη θέση i του γονέα P2 στη θέση i του απόγονου O1. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει.

Βήμα 7ο: Αν το γονίδιο που βρίσκεται στη θέση i του γονέα P1 υπάρχει στον απόγονο O1 πηγαίνουμε στο Βήμα 1. Αναζητούμε το γονίδιο που βρίσκεται στη θέση i του γονέα P1 στο γονέα P2(έστω ότι το βρίσκουμε στη θέση j).

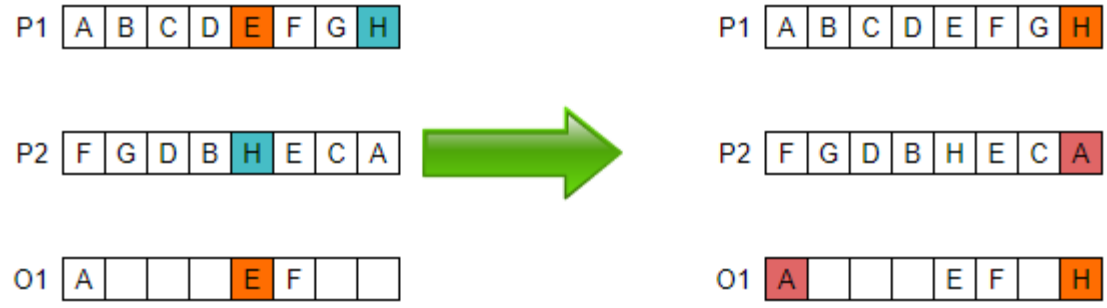
Βήμα 8ο: Τοποθετούμε το γονίδιο που βρίσκεται στη θέση j του γονέα P2 στη θέση j του O1. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει. Θέτουμε i=j και επιστρέφουμε στο Βήμα 6.

Για τη δημιουργία του δεύτερου απογόνου, ακολουθούμε την ίδια διαδικασία ξεκινώντας από το γονέα P2.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1=[A, B, C, D, E, F, G, H] και P2=[F, G, D, B, H, E, C, A]. Ξεκινάμε από την αρχή του γονέα P1 (θέση 1) και τοποθετούμε το γονίδιο που βρίσκεται σε αυτή τη θέση (A) στην αντίστοιχη θέση του πρώτου απογόνου. Στην αντίστοιχη θέση (θέση 1) του γονέα P2 βρίσκεται το γονίδιο F. Αναζητούμε το F στον γονέα P1 και τοποθετούμε το F στην αντίστοιχη θέση (θέση 6) του πρώτου απογόνου.

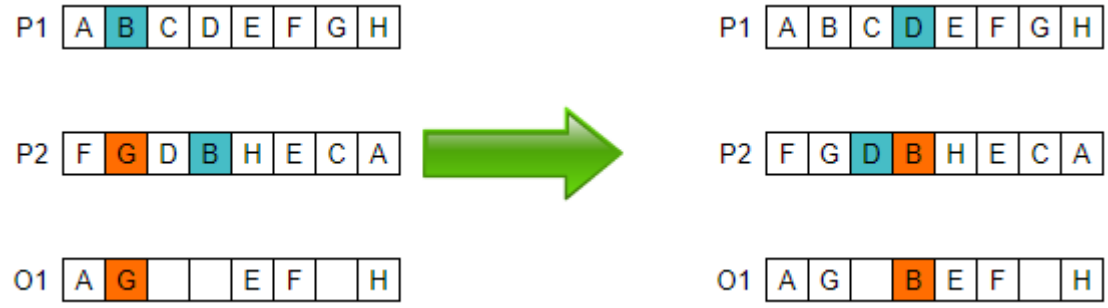
Εικόνα 126 Τοποθέτηση γονιδίων από τον γονέα P1

Στην αντίστοιχη θέση (θέση 6) του γονέα P2 βρίσκεται το γονίδιο E. Αναζητούμε το E στον γονέα P1 και τοποθετούμε το E στην αντίστοιχη θέση (θέση 5) του πρώτου απογόνου. Στην αντίστοιχη θέση (θέση 5) του γονέα P2 βρίσκεται το γονίδιο H. Αναζητούμε το H στον γονέα P1 και τοποθετούμε το H στην αντίστοιχη θέση (θέση 8) του πρώτου απογόνου. Στην αντίστοιχη θέση (θέση 8) του γονέα P2 βρίσκεται το γονίδιο A. Το γονίδιο A υπάρχει ήδη στον πρώτο απόγονο O1 = [A , _ , _ , _ , E , F , _ , H] οπότε ξεκινάει ο δεύτερος κύκλος.



Εικόνα 127 Ολοκλήρωση πρώτου κύκλου

Η πρώτη ελεύθερη θέση στον απόγονο O1 είναι η θέση 2. Ξεκινάμε από την θέση 2 του γονέα P2 αυτή τη φορά και τοποθετούμε το γονίδιο που βρίσκεται σε αυτή τη θέση (G) στην αντίστοιχη θέση του πρώτου απογόνου. Στην αντίστοιχη θέση (θέση 2) του γονέα P1 βρίσκεται το γονίδιο B. Αναζητούμε το B στον γονέα P2 και τοποθετούμε το B στην αντίστοιχη θέση (θέση 4) του πρώτου απογόνου.



Εικόνα 128 Τοποθέτηση γονιδίων από τον γονέα P2

Στην αντίστοιχη θέση (θέση 4) του γονέα P1 βρίσκεται το γονίδιο D. Αναζητούμε το D στον γονέα P2 και τοποθετούμε το D στην αντίστοιχη θέση (θέση 3) του πρώτου απογόνου. Στην αντίστοιχη θέση (θέση 3) του γονέα P1 βρίσκεται το γονίδιο C. Αναζητούμε το C στον γονέα P2 και τοποθετούμε το C στην αντίστοιχη θέση (θέση 7) του πρώτου απογόνου. Η δημιουργία του πρώτου απογόνου έχει ολοκληρωθεί O1 = [A , G , D , B , E , F , C , H]. Για τον δεύτερο απόγονο ακολουθούμε την ίδια διαδικασία αλλά ξεκινάμε από τον γονέα P2.

Εικόνα 129 Ολοκλήρωση αλγορίθμου

Σημείωση: Μερικά βιβλία προτείνουν μετά την ολοκλήρωση του πρώτου κύκλου να τοποθετήσουμε στον πρώτο απόγονο τα γονίδια από τον δεύτερο γονέα στις αντίστοιχες θέσεις και στον δεύτερο απόγονο τα γονίδια από τον πρώτο γονέα στις αντίστοιχες θέσεις. Αυτό γίνεται για να μειώσουμε την υπολογιστική ισχύ που χρειάζεται ο αλγόριθμος αλλά δεν δίνει πάντα τα ίδια αποτελέσματα.

Παράδειγμα: Για P1=[A, B, C, D, E, F, G, H] και P2=[F, G, D, B, H, E, C, A] όπως είδαμε ο πρώτος απόγονος είναι O1 = [A , G , D , B , E , F , C , H]. Αν μετά την ολοκλήρωση του πρώτου κύκλου τοποθετήσουμε στον απόγονο O1 = [A , _ , _ , _ , E , F , _ , H] τα γονίδια από τον P2 στις αντίστοιχες κενές θέσεις τότε προκύπτει ο ίδιο απόγονος.. Αν όμως έχουμε τους γονείς P1=[H, C, B, D, F, G, E, A] και P2=[F, C, A, B, D, E, G, H], μετά την ολοκλήρωση του πρώτου κύκλου προκύπτει ο απόγονος O1= [H , _ , B, D , F , _ , _ , A] . Αν τοποθετήσουμε τα γονίδια του P2 στις αντίστοιχες κενές θέσεις προκύπτει ο απόγονος O1 = [H , C , B, D , F , E ,G , A]. Αν όμως συνεχίσουμε τον αλγόριθμο ο δεύτερος κύκλος θα μας δώσει τον απόγονο O1 = [H , C , B, D , F , _ , _ , A] και ο τρίτος κύκλος τον απόγονο O1 = [H , C , B, D , F , G , E , A].

Το βασικό μειονέκτημα αυτής της μεθόδους είναι ότι μερικές φορές οι απόγονοι είναι ίδιοι με τους γονείς. Για παράδειγμα αν έχουμε τους γονείς P1 = [3 , 4 , 8 , 2 , 7 , 1 ,6 , 5] και

P1 = [4 , 2 , 5 , 1 , 6 , 8 ,3 , 7] τότε προκύπτουν οι απόγονοι offspring1 = [3 , 4 , 8 , 2 , 7 , 1 ,6 , 5] και offspring2 = [4 , 2 , 5 , 1 , 6 , 8 ,3 , 7].

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.21 Partially Mapped Crossover Operator (PMX)

Ο αλγόριθμος προτάθηκε από τους Goldberg and Lingle (1985) (Lingle, 1985) (Parashar, 2011) για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση μετάθεσης. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Η υλοποίηση του αλγορίθμου περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Παράγουμε δύο τυχαίους αριθμούς C1, C2 στο διάστημα [1,N], όπου N ο αριθμός των γονιδίων του χρωμοσώματος.

Βήμα 2ο: Τοποθετούμε τα γονίδια που βρίσκονται στις θέσεις C1 έως C2 του γονέα P1 στον απόγονο O2 στις αντίστοιχες θέσεις

Βήμα 3ο: Τοποθετούμε τα γονίδια που βρίσκονται στις θέσεις C1 έως C2 του γονέα P2 στον απόγονο O1 στις αντίστοιχες θέσεις

Βήμα 4ο: Ξεκινώντας από την αρχή του γονέα P1 τοποθετούμε τα γονίδια που δεν υπάρχουν ήδη στον απόγονο O1 στις αντίστοιχες θέσεις.

Βήμα 5ο: Ξεκινώντας από την αρχή του γονέα P2 τοποθετούμε τα γονίδια που δεν υπάρχουν ήδη στον απόγονο O2 στις αντίστοιχες θέσεις.

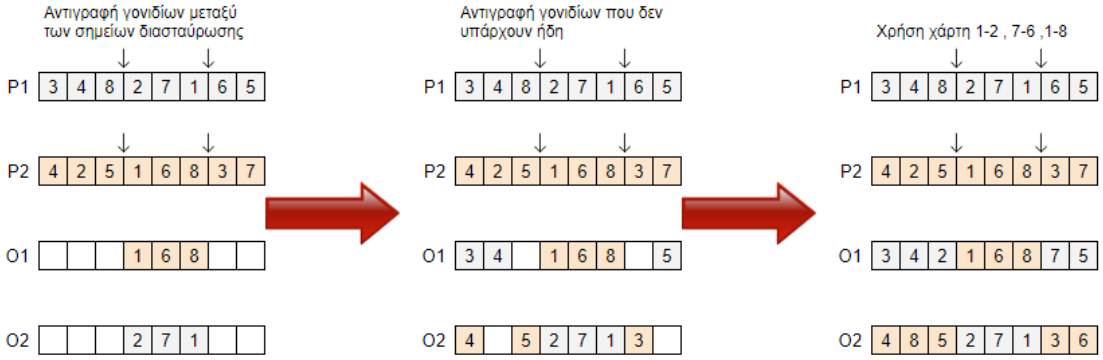
Βήμα 6ο: Δημιουργούμε ένα χάρτη (μια αντιστοιχία) των γονιδίων που βρίσκονται στις θέσεις C1 έως C2 του γονέα P1με τα αντίστοιχα γονίδια που βρίσκονται στις θέσεις C1 έως C2 του γονέα P2.

Βήμα 7ο: Συμπληρώνουμε τις κενές θέσεις των απογόνων O1,O2, κάνοντας χρήση του χάρτη που δημιουργήσαμε στο προηγούμενο βήμα. Η διαδικασία θα γίνει κατανοητή στο παράδειγμα που ακολουθεί.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [3 , 4 , 8 |, 2 , 7 , 1 |, 6 , 5], P2 = [4 , 2 , 5 |, 1 , 6 , 8 |, 3 , 7] με σημεία διασταύρωσης C1=4 και C2=6. Αρχικά παίρνουμε τους απογόνους O1 = [_ , _ , _ |, 1 , 6 , 8 |, _ , _] και O2 = [_ , _ , _ |, 2 , 7 , 1 |, _ , _] από τα γονίδια που βρίσκονται ανάμεσα στα σημεία διασταύρωσης C1 και C2. Στη συνέχεια συμπληρώνουμε στον απόγονο O1 τα γονίδια από το γονέα P1 O1 = [3 , 4 , _ |, 1 , 6 , 8 |, _ , 5] και στον απόγονο O2 τα γονίδια από το γονέα P2 O2 = [4 , _ , 5 |, 2 , 7 , 1 |, 3 , _] αφήνοντας καινές τις θέσεις στις οποίες το γονίδιο υπάρχει ήδη στον απόγονο.

Δημιουργούμε τον χάρτη 21 , 76 , 18. Θέλουμε να βάλουμε το γονίδιο 8 στον απόγονο O1 αλλά αυτό υπάρχει ήδη. Ελέγχοντας τον χάρτη μπορούμε να βάλουμε στη θέση του το γονίδιο 1 αφού 18. Αλλά και το γονίδιο 1 υπάρχει ήδη στον απόγονο O1. Συνεχίζουμε τον έλεγχο στο χάρτη και επειδή 21 βάζουμε το γονίδιο 2 στον απόγονο O1 = [3 , 4 , 2 |, 1 , 6 , 8 |, _ , 5]. Στην επόμενη κενή θέση θέλουμε να βάλουμε το γονίδιο 6 αλλά υπάρχει ήδη στον απόγονο O1. Από τον χάρτη βλέπουμε ότι μπορούμε να βάλουμε το 7 επειδή 76 οπότε ο πρώτος απόγονος είναι ο O1 = [3 , 4 , 2 |, 1 , 6 , 8 |, 7 , 5].

Για τον δεύτερο απόγονο θέλουμε να βάλουμε στην πρώτη κενή θέση το γονίδιο 2 αλλά υπάρχει ήδη στον απόγονο O2. Από τον χάρτη 21 αλλά και το γονίδιο 1 υπάρχει στον απόγονο O2 οπότε συνεχίζουμε τον έλεγχο στο χάρτη και επειδή 18 βάζουμε το γονίδιο 8 στον O2 = [4 , 8 , 5 |, 2 , 7 , 1 |, 3 , _]. Στην επόμενη κενή θέση του απογόνου O2 θέλουμε να βάλουμε το γονίδιο 7. Επειδή υπάρχει ήδη στον απόγονο O2 ελέγχουμε το χάρτη και επειδή 76 τοποθετούμε το γονίδιο 6. Οπότε ο δεύτερος απόγονος είναι ο O2 = [4 , 8 , 5 |, 2 , 7 , 1 |, 3 , 6].



Εικόνα 130 Partially Mapped Crossover

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.22 Modified Partially-Mapped Crossover (MPMX)

Ο τελεστής Modified Partially-Mapped Crossover προτάθηκε από τον Brown (1989) (Donald E. Brown, 1989) για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση μετάθεσης και είναι μια παραλλαγή του αλγορίθμου Partially-Mapped crossover, η οποία για να αυξήσει την ταχύτητα του αλγορίθμου, στο τελευταίο βήμα τοποθετεί τα γονίδια που απομένουν τυχαία. Έστω ότι έχουμε τους

γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Η υλοποίηση του αλγορίθμου περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Παράγουμε δύο τυχαίους αριθμούς C1, C2 στο διάστημα [1,N], όπου N ο αριθμός των γονιδίων του χρωμοσώματος.

Βήμα 2ο: Τοποθετούμε τα γονίδια που βρίσκονται στις θέσεις C1 έως C2 του γονέα P1 στον απόγονο O1 στις αντίστοιχες θέσεις.

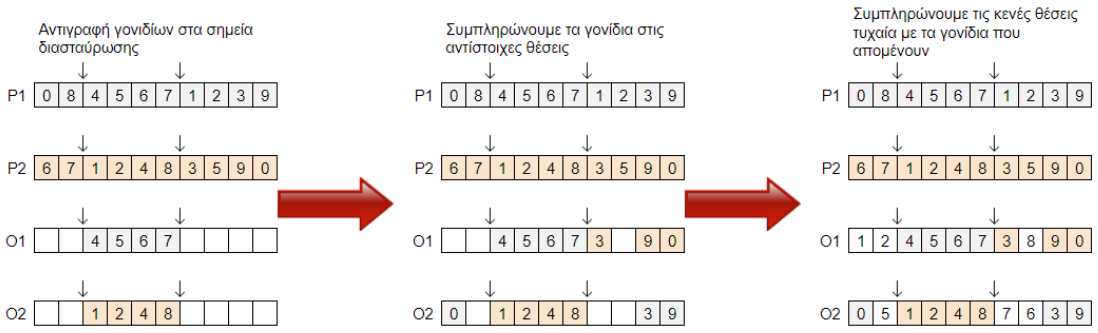
Βήμα 3ο: Τοποθετούμε τα γονίδια που βρίσκονται στις θέσεις C1 έως C2 του γονέα P2 στον απόγονο O2 στις αντίστοιχες θέσεις.

Βήμα 4ο: Ξεκινώντας από την αρχή του γονέα P1 τοποθετούμε τα γονίδια που δεν υπάρχουν ήδη στον απόγονο O2 στις αντίστοιχες θέσεις.

Βήμα 5ο: Ξεκινώντας από την αρχή του γονέα P2 τοποθετούμε τα γονίδια που δεν υπάρχουν ήδη στον απόγονο O1 στις αντίστοιχες θέσεις.

Βήμα 6ο: Συμπληρώνουμε τυχαία τις κενές θέσεις των O1, O2 με τα γονίδια που δεν υπάρχουν ήδη.

Παράδειγμα: Έστω οι γονείς P1 = [0 , 8 , 4 , 5 , 6 , 7 , 1 , 2 , 3 , 9], P2 = [6 , 7 , 1 , 2 , 4 ,8 , 3 , 5 , 9 , 0] και οι τυχαίοι αριθμοί C1 = 3 και C2 = 7. Αντιγράφοντας τα γονίδια του P1 που βρίσκονται μεταξύ των θέσεων 3 και 7 παίρνουμε τον απόγονο O1 = [_ , _ , 4 , 5 , 6 , 7 , _ , _ , _ , _]. Αντιγράφοντας τα γονίδια του P2 που δεν υπάρχουν ήδη στον O1 ο απόγονος γίνεται O1 = [_ , _ , 4 , 5 , 6 , 7 , 3 , _ ,9 , 0]. Οι 3 κενές θέσεις μπορούν να συμπληρωθούν με όποιο συνδυασμό των γονιδίων 2, 8 και 1 θέλουμε. Αντιγράφοντας τα γονίδια του P2 που βρίσκονται μεταξύ των θέσεων 3 και 7 παίρνουμε τον απόγονο O2 = [_ , _ , 1 , 2 , 4 , 8 , _ , _ , _ , _]. Αντιγράφοντας τα γονίδια του P1 που δεν υπάρχουν ήδη στον O2 ο απόγονος γίνεται O2 = [0 , _ , 1 , 2 , 4 , 8 , _ , _ ,3 , 9] . Οι 3 κενές θέσεις μπορούν να συμπληρωθούν με όποιο συνδυασμό των γονιδίων 5, 6, 7 θέλουμε.



Εικόνα 131 Modified Partially-Mapped Crossover

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.23 Edge recombination crossover (ERX ή ER) - 1 offspring

Παρουσιάστηκε πρώτη φορά από τους Whitley el al (Whitley, Starkweather, & Fuquay, 1989) (Manger, 2013) για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση μετάθεσης και επειδή δίνει πολύ καλά αποτελέσματα (P. Larrañaga, 1999) για το TSP έχουν δημιουργηθεί πολλές παραλλαγές (Dr. Anantkumar J. Umbarkar, 2015) (H.D. Nguyen, 2000).

Στην αρχή για κάθε γονίδιο δημιουργούμε μία λίστα με τα γονίδια που συνορεύει και στους δύο γονείς. Στην αρχή αυτή η λίστα θα περιέχει από 2 έως 4 γονίδια. 2 αν συνορεύει με τα ίδια γονίδια και στους δύο γονείς και 4 αν τα γονίδια είναι διαφορετικά. Όλες οι λίστες μαζί αποτελούν τον λεγόμενο χάρτη άκρων (edge map).

Ξεκινάμε από ένα τυχαίο γονίδιο. Για ευκολία ας υποθέσουμε ότι ξεκινάμε από τον πρώτο γονίδιο του πρώτο γονέα (έστω ότι είναι το γονίδιο 1). Ελέγχουμε τον χάρτη και βρίσκουμε με ποια γονίδια συνορεύει. Στη συνέχεια ελέγχοντας τον χάρτη βρίσκουμε ποιο από τα γονίδια με τα οποία συνορεύει το γονίδιο 1, συνορεύει με τα λιγότερα γονίδια. Αν υπάρχουν περισσότερες από μία επιλογές, επιλέγουμε μία τυχαία. Επαναλαμβάνουμε την ίδια διαδικασία μέχρι να συμπληρωθεί ο απόγονος. Εδώ θα πρέπει να σημειώσουμε ότι όταν μετράμε τα γονίδια, αφαιρούμε αυτά που υπάρχουν ήδη στον απόγονο. Μετράμε δηλαδή μόνο τις ενεργές άκρες (active edges). Ο αλγόριθμος ERX περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Για κάθε γονίδιο δημιουργούμε μια λίστα με τα γονίδια που συνορεύει και στους δύο γονείς.

Βήμα 2ο: Επιλέγουμε τυχαία ένα από τα γονίδια ως σημείο εκκίνησης (έστω ότι επιλέξαμε το γονίδιο g), το αφαιρούμε από όλες τις λίστες των γονιδίων και το προσθέτουμε στον απόγονο.

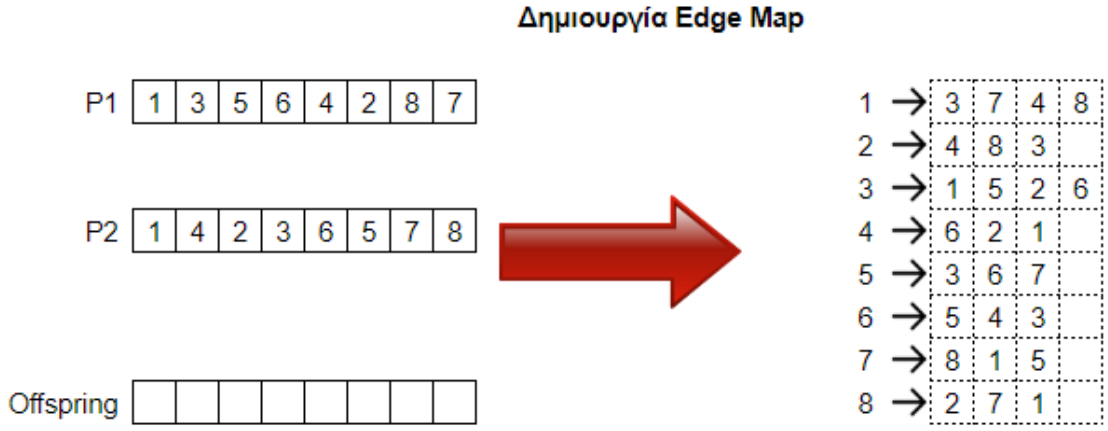
Βήμα 3ο: Από τη λίστα του γονιδίου g βρίσκουμε με ποια γονίδια συνορεύει. Αν δεν συνορεύει με κάποιο ο αλγόριθμος τερματίζει.

Βήμα 4ο: Για όλα τα γονίδια που συνορεύει τον g, βρίσκουμε τον αριθμό των γονιδίων που συνορεύουν ελέγχοντας τις αντίστοιχες λίστες.

Βήμα 5ο: Θέτουμε g το γονίδιο που συνορεύει με τα λιγότερα γονίδια (αν υπάρχουν περισσότερα από ένα επιλέγουμε τυχαία), το προσθέτουμε στον απόγονο, το αφαιρούμε από όλες τις λίστες των γονιδίων και επιστρέφουμε στο Βήμα 3.

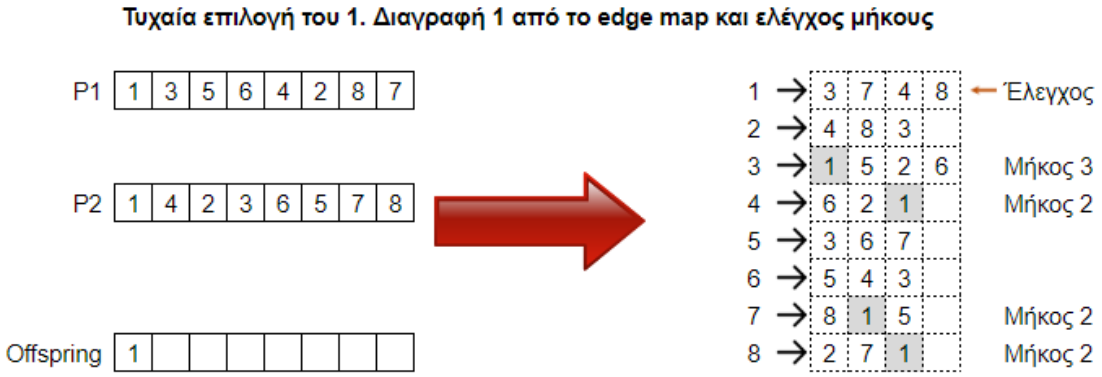
Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [1 , 3 , 5 , 6 , 4 , 2 , 8 , 7] και P2 = [1 , 4 , 2 , 3 , 6 , 5 , 7 , 8]. Το γονίδιο 1 συνορεύει με τα γονίδια 7 και 3 στον πρώτο γονέα και με τα γονίδια 4 και 8 στον δεύτερο. Το γονίδιο 2 συνορεύει με τα γονίδια 4 και 8 στον πρώτο γονέα και με τα γονίδια 4 και 3 στον δεύτερο κτλ. Προκύπτει ο ακόλουθος edge map:

- Το γονίδιο 1 συνορεύει με τα γονίδια: 3 4 7 8
- Το γονίδιο 2 συνορεύει με τα γονίδια: 3 4 8
- Το γονίδιο 3 συνορεύει με τα γονίδια: 1 2 5 6
- Το γονίδιο 4 συνορεύει με τα γονίδια: 1 2 6
- Το γονίδιο 5 συνορεύει με τα γονίδια: 3 6 7
- Το γονίδιο 6 συνορεύει με τα γονίδια: 3 4 5
- Το γονίδιο 7 συνορεύει με τα γονίδια: 1 5 8
- Το γονίδιο 8 συνορεύει με τα γονίδια: 1 2 7



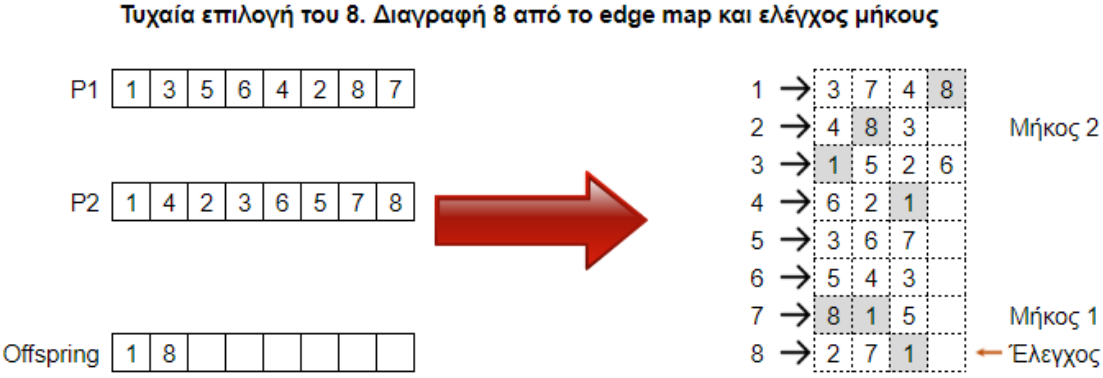
Εικόνα 132 Edge map

Έστω ότι ξεκινάμε τυχαία από το γονίδιο 1 το οποίο όπως φαίνεται στον edge map συνορεύει με τα γονίδια 3 , 4 , 7 και 8. Το γονίδιο 3 έχει 3 active edges τις 2 5 και 6. Το γονίδιο 4 έχει 2 active edges τις 2 και 6. Το γονίδιο 7 έχει 2 active edges τις 5 και 8. Το γονίδιο 8 έχει 2 active edges τις 2 και 7. Τα γονίδια 4, 7 και 8 έχουν τις ελάχιστες active edges (2). Επιλέγουμε τυχαία ένα από αυτά, έστω το γονίδιο 8.



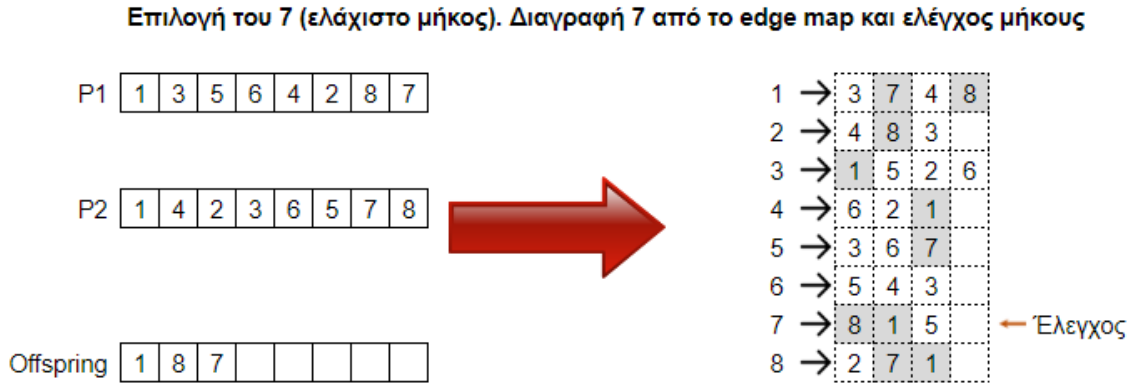
Εικόνα 133 Πρώτο βήμα αλγορίθμου

Από το γονίδιο 8 μπορούμε να πάμε στα γονίδια 2 και 7 (το γονίδιο 1 υπάρχει ήδη στον απόγονο). Το γονίδιο 2 έχει 2 active edges τις 3 και 4. Το γονίδιο 7 έχει 1 active edge (τα γονίδια 1 και 8 υπάρχουν ήδη στον απόγονο). Οπότε επιλέγουμε το γονίδιο 7 και ο απόγονος ως τώρα είναι ο O = [1 , 8 , 7 , _ , _ , _ , _ , _].



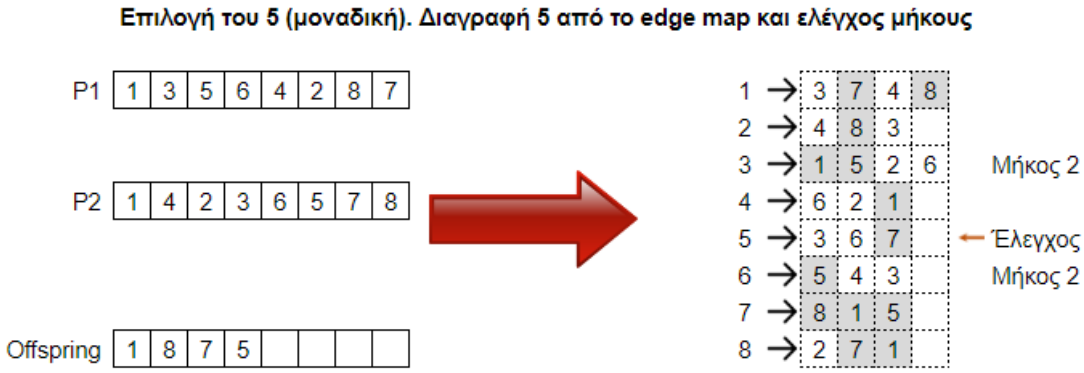
Εικόνα 134 Δεύτερο βήμα αλγορίθμου

Από το γονίδιο 7 μπορούμε να πάμε μόνο στο γονίδιο 5. Οπότε το επιλέγουμε και ο απόγονος γίνεται O = [1 , 8 , 7 , 5 , _ , _ , _ , _].



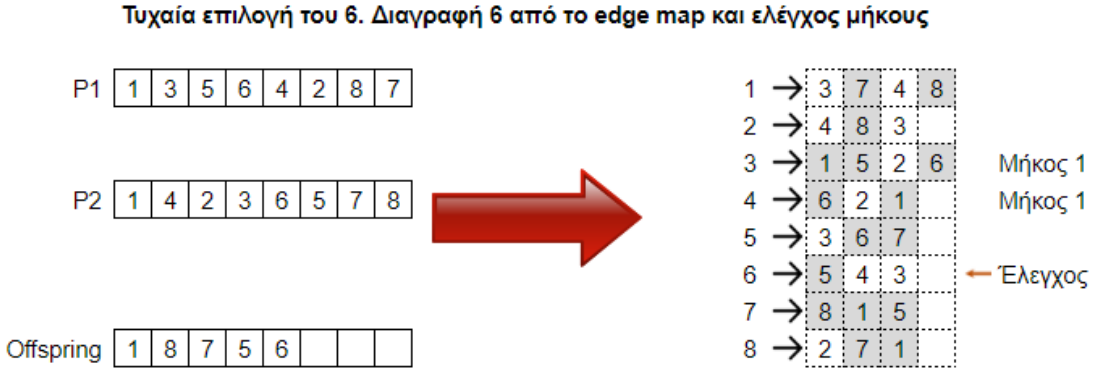
Εικόνα 135 Τρίτο βήμα αλγορίθμου

Από το γονίδιο 5 μπορούμε να πάμε στα γονίδια 3 και 6 (το γονίδιο 7 υπάρχει ήδη στον απόγονο). Το γονίδιο 3 έχει 2 active edges τις 2 και 6. Το γονίδιο 6 έχει 2 active edges τις 3 και 4. Επιλέγουμε τυχαία ένα από τα γονίδια 3 και 6, έστω το 6 οπότε ο απόγονος γίνεται O = [1 , 8 , 7 , 5 , 6 , _ , _ , _].



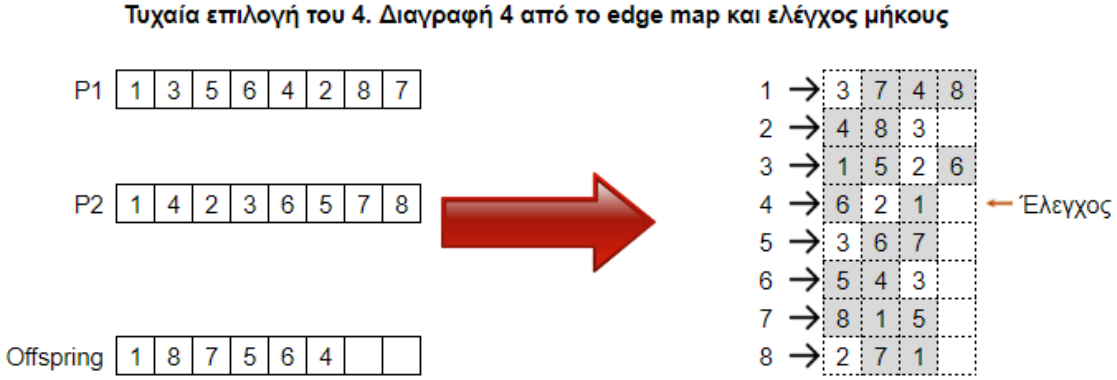
Εικόνα 136 Τέταρτο βήμα αλγορίθμου

Από το γονίδιο 6 μπορούμε να πάμε στα γονίδια 3 και 4. Από τον edge map βρίσκουμε ότι το γονίδιο 3 έχει μία active edge την 2 και το γονίδιο 4 έχει επίσης μία active edge την 2. Επιλέγουμε τυχαία το γονίδιο 4 και ο απόγονος γίνεται O = [1 , 8 , 7 , 5 , 6 , 4 , _ , _].



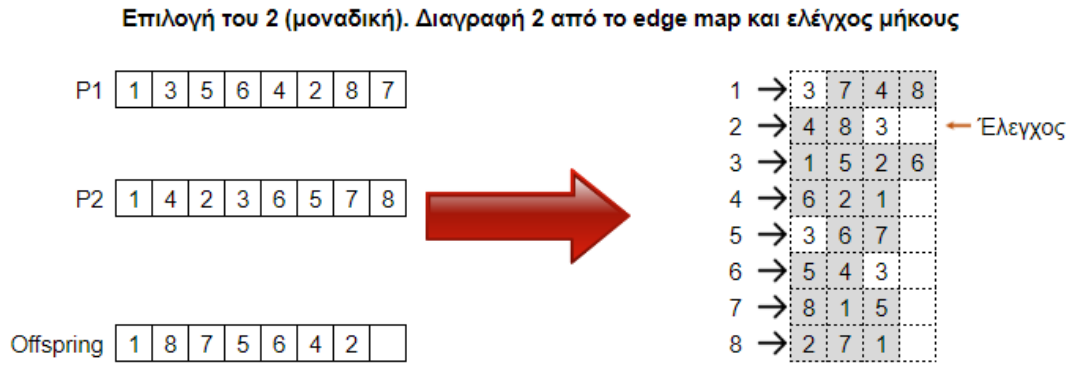
Εικόνα 137 Πέμπτο βήμα αλγορίθμου

Από το γονίδιο 4 μπορούμε να πάμε μόνο στο γονίδιο 2. Οπότε το επιλέγουμε και απόγονος γίνεται O = [1 , 8 , 7 , 5 , 6 , 4 , 2 , _].



Εικόνα 138 Έκτο βήμα αλγορίθμου

Από το γονίδιο 2 μπορούμε να πάμε μόνο στο γονίδιο 3. Οπότε το επιλέγουμε και απόγονος γίνεται O = [1 , 8 , 7 , 5 , 6 , 4 , 2 , 3].



Εικόνα 139 Έβδομο βήμα αλγορίθμου

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.24 Greedy Subtour Crossover - GSX-0

Ο συγκεκριμένος τελεστής διασταύρωσης προτάθηκε από τους Sengoku and Yushihara (H. Sengoku, 1999) και αναφέρεται για ιστορικούς λόγους καθώς όπως θα δούμε στη συνέχεια υπάρχουν δύο βελτιωμένες εκδόσεις. Χρησιμοποιείται αποκλειστικά σε γενετικούς αλγορίθμους με κωδικοποίηση μετάθεσης και παράγει έναν απόγονο. Ο αλγόριθμος GSX-0 περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Επιλέγουμε ένα τυχαίο αριθμό r στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων.

Βήμα 2ο: Τοποθετούμε το γονίδιο που βρίσκεται στη θέση r στον πρώτο γονέα στον απόγονο.

Βήμα 3ο: Αναζητούμε το γονίδιο που βρίσκεται στη θέση r του γονέα ένα στον γονέα δύο και σημειώνουμε τη θέση του (Έστω k).

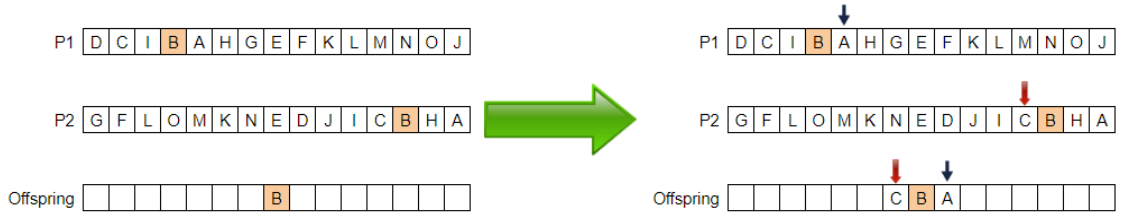
Βήμα 4ο: Ελέγχουμε αν το γονίδιο που βρίσκεται στη θέση r+1 του γονέα ένα υπάρχει ήδη στον απόγονο και αυξάνουμε το r κατά 1. Αν δεν υπάρχει το τοποθετούμε στο τέλος του απογόνου και πηγαίνουμε στο Βήμα 5. Αν υπάρχει πηγαίνουμε στο Βήμα 6. Σημείωση: Αν κατά τον έλεγχο φτάσουμε στο τέλος του γονέα δηλαδή αν r+1>N ξεκινάμε από την αρχή θέτοντας r=1.

Βήμα 5ο: Ελέγχουμε αν το γονίδιο που βρίσκεται στη θέση k-1 του γονέα δύο υπάρχει ήδη στον απόγονο και μειώνουμε το k κατά 1. Αν δεν υπάρχει το τοποθετούμε στην αρχή του απογόνου και πηγαίνουμε στο βήμα 4. Αν υπάρχει πηγαίνουμε στο Βήμα 6. Σημείωση: Αν κατά τον έλεγχο φτάσουμε στην αρχή του γονέα δηλαδή αν k-1<1 ξεκινάμε από το τέλος θέτοντας r=N. Αν ο αριθμός των γονιδίων του απογόνου γίνει ίσος με N ο αλγόριθμος τερματίζει.

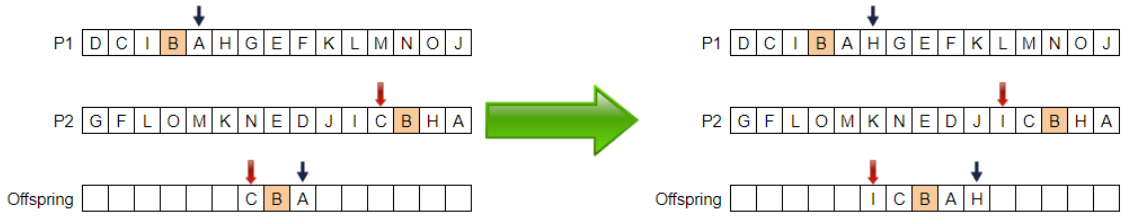
Βήμα 6ο: Συμπληρώνουμε τυχαία τα γονίδια του απογόνου που απομένουν.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [D, C, I, **B**, A, H, G, E, F, K, L, M, N, O, J] και P2 = [G, F, L, O, M, K, N, E, D, J, I, C, **B**, H, A]. Επιλέγουμε για αρχή το γονίδιο B. Βρίσκουμε τη θέση του B και στους δύο γονείς και τοποθετούμε το B στον απόγονο. Δεξιά του B στον πρώτο γονέα είναι το A και αριστερά του B στον δεύτερο γονέα είναι το C. Οπότε ο απόγονος γίνεται O = [C, B, A, _, _, _, _, _, _, _, _, _, _, _, _].

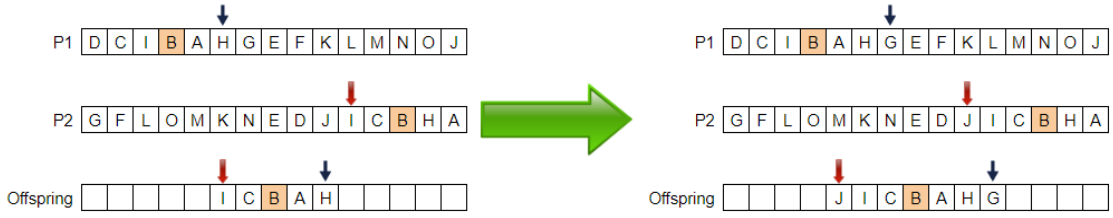
Επαναλαμβάνοντας τη διαδικασία ο απόγονος γίνεται O = [I, C, B, A, H, _, _, _, _, _, _, _, _, _, _], O = [J, I, C, B, A, H, G, _, _, _, _, _, _, _, _], O = [D, J, I, C, B, A, H, G, E, _, _, _, _, _, _]. Σε αυτό το σημείο εισάγουμε το F από τον γονέα ένα αλλά το γονίδιο E που θέλουμε να εισάγουμε από τον δεύτερο γονέα, υπάρχει ήδη στον απόγονο. Συμπληρώνουμε τυχαία τα γονίδια που απομένουν και παίρνουμε O = [M, K, O, D, J, I, C, B, A, H, G, E, F, L, N]. Παρατηρούμε ότι ο απόγονος περιέχει τρεις διαδρομές τις KO, OD και LN που δεν υπήρχαν σε κανέναν από τους δύο γονείς. Στα παρακάτω σχήματα φαίνονται τα βήματα του αλγορίθμου.



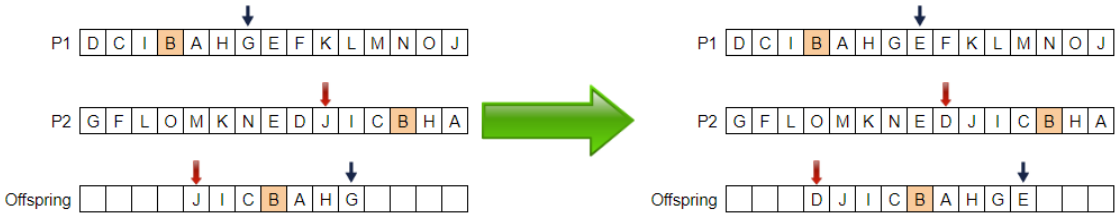
Εικόνα 140 Πρώτο βήμα αλγορίθμου



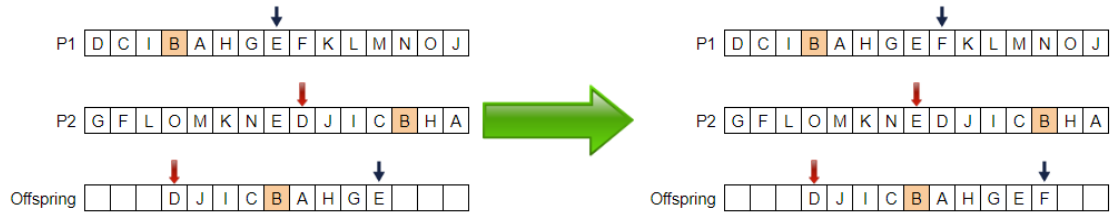
Εικόνα 141 Δεύτερο βήμα αλγορίθμου



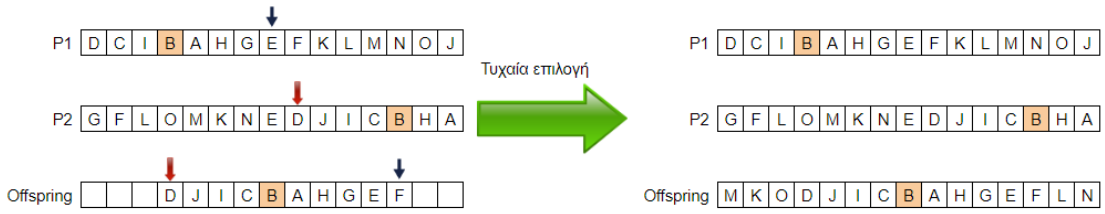
Εικόνα 142 Τρίτο βήμα αλγορίθμου



Εικόνα 143 Τέταρτο βήμα αλγορίθμου



Εικόνα 144 Πέμπτο βήμα αλγορίθμου



Εικόνα 145 Έκτο βήμα αλγορίθμου

Σημείωση: Μπορούμε να βάλουμε τα γονίδια του πρώτο γονέα από την αριστερή πλευρά και τα γονίδια του δεύτερου από την δεξιά. Δεν υπάρχει διαφορά καθώς αυθαίρετα θεωρούμε έναν από τους δύο "πρώτο" και τον άλλο δεύτερο.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.24.1 Greedy Subtour Crossover - GSX-1

Η πρώτη τροποποίηση του GSX έρχεται από τους Takeda et al. (Takeda, 1999). Το μόνο που αλλάζει είναι ότι αντί να συμπληρώσουμε τυχαία τα γονίδια στο τελευταίο βήμα του αλγορίθμου, τα συμπληρώνουμε με βάση τη σειρά που εμφανίζονται στον γονέα δύο ή ένα (κάποιες έρευνες αναφέρουν το γονέα δύο (Hung Dinh Nguyen, 2003), κάποιες άλλες του γονέα ένα (Hassan Ismkhan, 2013)).

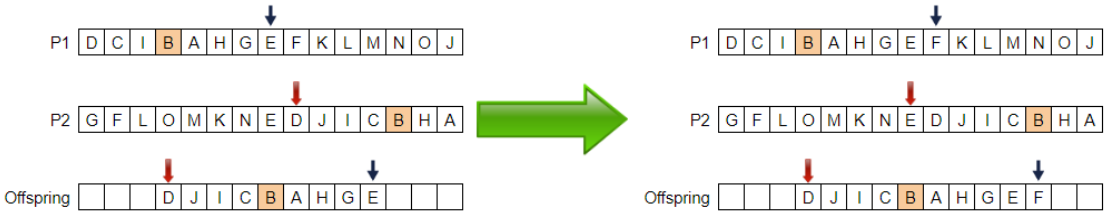
Παράδειγμα 1: (Συμπληρώνουμε με βάση τη σειρά εμφάνισης στον δεύτερο γονέα). Έστω ότι έχουμε το παράδειγμα που είδαμε στον GSX-0 και βρισκόμαστε στο βήμα που ο απόγονος είναι O = [D, J, I, C, B, A, H, G, E, F, _, _, _, _, _]. Υπενθυμίζουμε ότι οι δύο γονείς είναι P1 = [D, C, I, **B**, A, H, G, E, F, K, L, M, N, O, J] και P2 = [G, F, L, O, M, K, N, E, D, J, I, C, **B**, H, A].

Στο γονέα δύο συναντάμε τα γονίδια N, K, M, O, L οπότε τα εισάγουμε με τη σειρά στον απόγονο ο οποίος γίνεται O = [M, K, N, D, J, I, C, B, A, H, G, E, F, L, O]. Παρατηρούμε ότι ο απόγονος έχει μόνο μία διαδρομή την ND που δεν υπάρχει σε κανέναν από τους δύο γονείς, δηλαδή δύο λιγότερες από τον GSX-0. Αυτό είναι σημαντικό όταν ο Γ.Α. συνδυάζεται με ευρετικές συναρτήσεις που βρίσκουν τοπικά ελάχιστα.

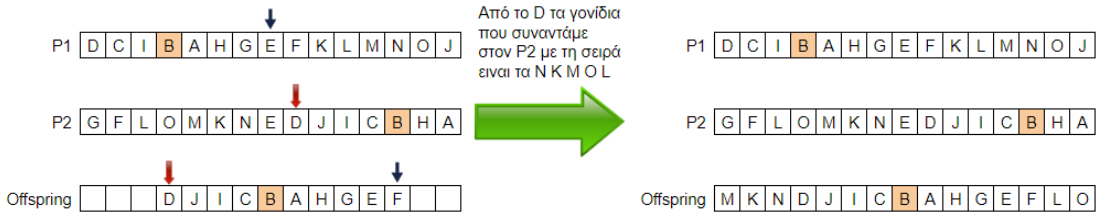
Παράδειγμα 2: (Συμπληρώνουμε με βάση τη σειρά εμφάνισης στον πρώτο γονέα). Έστω ότι έχουμε τους γονείς P1 = [7, 3, 1, 4, 6, 8, 2, 5], P2 = [7, 5, 6, 4, 2, 8, 3, 1] και ξεκινάμε από το γονίδιο 4. Βάζουμε το γονίδιο 4 στον απόγονο. Βάζουμε το γονίδιο 6 στον απόγονο από τον γονέα P1 όμως το 6 που θέλουμε να βάλουμε από τον γονέα P2 υπάρχει ήδη. Οπότε βάζουμε τα γονίδια με τη σειρά που

εμφανίζονται στον γονέα P1. O = [7, 3, 1, 4, 6, 8, 2, 5]. Παρατηρούμε ότι ο απόγονος είναι ίδιος με τον γονέα P1. Αυτός είναι και ο λόγος που δημιουργήθηκε ο GSX-2.

Στο σχήμα που ακολουθεί βλέπουμε τη διαφορά του GSX-1 για το παράδειγμα 1.



Εικόνα 146 Από το παράδειγμα του GSX-0 με τη χρήση του GSX-1



Εικόνα 147 Αποτέλεσμα GSX-1

Η υλοποίηση του αλγορίθμου σε php από parent1 βρίσκεται εδώ.

Η υλοποίηση του αλγορίθμου σε php από parent2 βρίσκεται εδώ.

7.24.2 Greedy Subtour Crossover - GSX-2

Ο αλγόριθμος GSX-1 έχει ένα σημαντικό μειονέκτημα. Αν οι δύο γονείς περιέχουν αριστερά και δεξιά του τυχαίου γονιδίου που επιλέχθηκε τα ίδια γονίδια σε αντίστροφη σειρά, δηλαδή αν στον γονέα ένα υπάρχει η υποδιαδρομή A B C και στον γονέα δύο η υποδιαδρομή C B A τότε μόλις βάλουμε το C δεξιά του B στον απόγονο, ο αλγόριθμος δεν μπορεί να συνεχίσει και αντιγράφει τα γονίδια του γονέα δύο στον απόγονο. Αυτό έχει ως αποτέλεσμα ο απόγονος να είναι σχεδόν ίδιος με τον γονέα δύο.

Για να λύσουν αυτό το πρόβλημα οι Hung Dinh Nguyen et al (Hung Dinh Nguyen, 2003) πρότειναν το εξής. Έστω x, y τα γονίδια που βρίσκονται αριστερά και δεξιά αντίστοιχα του γονιδίου που επιλέξαμε στον γονέα ένα. Έστω p, q τα γονίδια που βρίσκονται αριστερά και δεξιά αντίστοιχα του γονιδίου που επιλέξαμε στον γονέα δύο. Αν x=q ή p=y τότε αντί να προσθέτουμε τα γονίδια που βρίσκονται αριστερά του γονέα δύο, προσθέτουμε αυτά που βρίσκονται δεξιά.

Επίσης η προσθήκη γονιδίων εναλλάξ δεν σταματάει μόνο όταν ένα από τα γονίδια υπάρχει ήδη στον απόγονο αλλά και αν βρεθεί κάτι που οι Hung Dinh Nguyen et al (Hung Dinh Nguyen, 2003) ονομάζουν “exchangeable subtour”. Ως exchangeable subtour ορίζονται δύο διαδρομές (μία σε κάθε γονέα) που ξεκινάνε και τελειώνουν στο ίδιο γονίδιο και έχουν τα ίδια ενδιάμεσα γονίδια αλλά σε διαφορετική σειρά.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.25 Alternating-position crossover (AP)

Ο αλγόριθμος προτάθηκε από τους Larranaga et al. (Larranaga, 1994) για τους γενετικούς αλγορίθμους που χρησιμοποιούν κωδικοποίηση μετάθεσης. Ξεκινώντας από τον πρώτο γονέα εισάγουμε το πρώτο γονίδιο στον απόγονο. Στη συνέχεια ελέγχουμε επίσης το πρώτο γονίδιο του δεύτερου γονέα και αν δεν υπάρχει ήδη στον απόγονο, το εισάγουμε. Επαναλαμβάνουμε την διαδικασία για το δεύτερο γονίδιο του πρώτου γονέα, το δεύτερο γονίδιο του δεύτερου γονέα κτλ. μέχρι να γεμίσει ο απόγονος. Αν θέλουμε

και δεύτερο απόγονο εναλλάσσουμε τους γονείς. Έστω ότι έχουμε τους γονείς P1, P2 και θέλουμε να δημιουργήσουμε τους απογόνους O1, O2. Η υλοποίηση του αλγορίθμου περιγράφεται στα παρακάτω βήματα:

Βήμα 1ο: Θέτουμε i=1.

Βήμα 2ο: Ελέγχουμε αν το γονίδιο που βρίσκεται στη θέση i του γονέα P1 υπάρχει ήδη στον απόγονο. Αν δεν υπάρχει το τοποθετούμε στον απόγονο. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει.

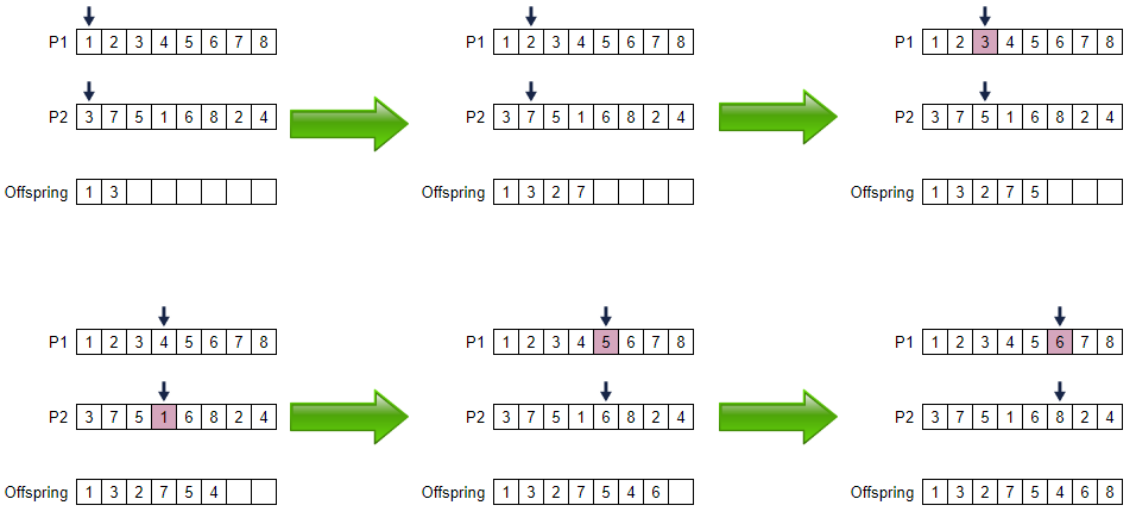
Βήμα 3ο: Ελέγχουμε αν το γονίδιο που βρίσκεται στη θέση i του γονέα P2 υπάρχει ήδη στον απόγονο. Αν δεν υπάρχει το τοποθετούμε στον απόγονο. Αν ο απόγονος έχει γεμίσει ο αλγόριθμος τερματίζει, αν όχι αυξάνουμε το i κατά 1 και επιστρέφουμε στο Βήμα 2ο.

Για να πάρουμε τον δεύτερο γονέα αλλάζουμε τη σειρά των P1,P2 και επαναλαμβάνουμε τη διαδικασία.

Παράδειγμα: Έστω ότι έχουμε τους γονείς P1 = [1, 2, 3, 4, 5, 6, 7, 8] και P2 = [3, 7, 5, 1, 6, 8, 2, 4].

- Στη θέση 1 του γονέα P1 υπάρχει το γονίδιο 1. Το τοποθετούμε στον απόγονο. [1]
- Στη θέση 1 του γονέα P2 υπάρχει το γονίδιο 3. Το τοποθετούμε στον απόγονο. [1,3]
- Στη θέση 2 του γονέα P1 υπάρχει το γονίδιο 2. Το τοποθετούμε στον απόγονο. [1,3,2]
- Στη θέση 2 του γονέα P2 υπάρχει το γονίδιο 7. Το τοποθετούμε στον απόγονο. [1,3,2,7]
- Στη θέση 3 του γονέα P1 υπάρχει το γονίδιο 3. Υπάρχει ήδη στον απόγονο. [1,3,2,7]
- Στη θέση 3 του γονέα P2 υπάρχει το γονίδιο 5. Το τοποθετούμε στον απόγονο. [1,3,2,7,5]
- Στη θέση 4 του γονέα P1 υπάρχει το γονίδιο 4. Το τοποθετούμε στον απόγονο. [1,3,2,7,5,4]
- Στη θέση 4 του γονέα P2 υπάρχει το γονίδιο 1. Υπάρχει ήδη στον απόγονο. [1,3,2,7,5,4]
- Στη θέση 5 του γονέα P1 υπάρχει το γονίδιο 4. Υπάρχει ήδη στον απόγονο. [1,3,2,7,5,4]
- Στη θέση 5 του γονέα P2 υπάρχει το γονίδιο 6. Το τοποθετούμε στον απόγονο. [1,3,2,7,5,4,6]
- Στη θέση 6 του γονέα P1 υπάρχει το γονίδιο 6. Υπάρχει ήδη στον απόγονο. [1,3,2,7,5,4,6]
- Στη θέση 6 του γονέα P2 υπάρχει το γονίδιο 8. ο τοποθετούμε στον απόγονο.. [1,3,2,7,5,4,6,8]

Ο απόγονος έχει γεμίσει άρα ο αλγόριθμος τερματίζει. Αν θέλουμε να πάρουμε και δεύτερο απόγονο ξεκινάμε από τον γονέα P2 και παίρνουμε τον απόγονο O2 = [3, 1, 7, 2, 5, 4, 6, 8].



Εικόνα 148 Alternating-position crossover

Επειδή όπως βλέπουμε το μόνο που αλλάζει στους δύο απογόνους είναι η σειρά των γονιδίων ανά δύο, η μέθοδος αυτή χρησιμοποιείται συνήθως για έναν απόγονο.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

7.26 Εναλλακτικοί αλγόριθμοι διασταύρωσης

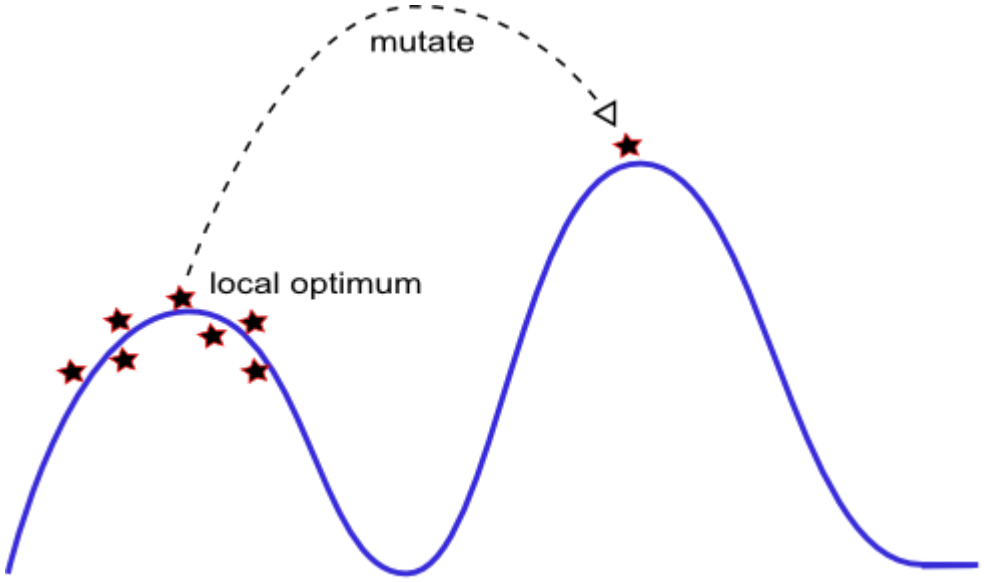
Στα προηγούμενα κεφάλαια αναλύσαμε ένα μεγάλο αριθμό αλγορίθμων διασταύρωσης για διαφορετικές μορφές κωδικοποίησης. Σε αυτό το κεφάλαιο θα γίνει μια απλή αναφορά σε μερικούς ακόμα αλγορίθμους.

- Average crossover (Davis L. , 1991) (Nomura, 1997).
- Unfair average crossover (Nomura, 1997).
- BLX-alpha-beta (Stjepan Picek D. J., 2013).
- Extended line crossover (H. Mühlenbein, 1993).
- Extended intermediate crossover (H. Mühlenbein, 1993).
- Linear BGA crossover (Mühlenbein, 1994).
- Fuzzy connectives based crossover (FCB) (F. Herrera E. H.-v., 1994).
- Confidence interval based crossover using L2 norm (CIXL2) (Nicolás García-Pedrajas, 2005).
- Laplace crossover (K. Deep, 2007).
- Cut on worst gene crossover (COWGC) (Ahmad B. A. Hassanat, 2017).
- Cut on worst L+R crossover (COWLRGC) (Ahmad B. A. Hassanat, 2017).
- Collision crossover (Ahmad B. A. Hassanat, 2017).
- Multiple crossover (Chang, 2006).
- Statistics-based adaptive nonuniform crossover (SANUX) (Dr. Anantkumar J. Umbarkar, 2015).
- Random respectful crossover (RRC) (Dr. Anantkumar J. Umbarkar, 2015).
- Unimodal normal distribution crossover (Ono I, 1997).
- Best order crossover (BOX) (Anca Andreica, 2014).
- Περισσότερους αλγορίθμους διασταύρωσης μπορείτε να βρείτε εδώ (Dr. Anantkumar J. Umbarkar, 2015) (Siew Mooi Lim, 2017).

8 Μετάλλαξη (Mutation)

Μετάλλαξη (mutation) ονομάζεται η διαδικασία κατά την οποία τροποποιείται ένα ή και περισσότερα γονίδια (genes) μιας υποψήφιας λύσης. Σε αντίθεση με τον τελεστή διασταύρωσης η μετάλλαξη εφαρμόζεται με μια πιθανότητα Pm σε ένα μόνο χρωμόσωμα κάθε φορά ή σε ένα γονίδιο.

Στόχος του τελεστή μετάλλαξης είναι να τροφοδοτεί τον πληθυσμό των υποψήφιων λύσεων με νέες λύσεις αυξάνοντας έτσι το χώρο αναζήτηση και αποτρέποντας τον γενετικό αλγόριθμο να εγκλωβιστεί σε ένα τοπικό ακρότατο.



Εικόνα 149 Τι μπορεί να συμβεί με τη μετάλλαξη

Η πιθανότητα μετάλλαξης (mutation rate) καθορίζεται στην αρχή του αλγόριθμου από τον χρήστη. Δεν υπάρχουν συγκεκριμένοι κανόνες αλλά συνήθως επιλέγεται ένας μικρός αριθμός (Jong, 1975) (Schaffer, 1989) (Ahmad Hassanat, 2019) (Grefenstette, 1986) καθώς στόχος μας είναι να αυξήσουμε το χώρο αναζήτησης λύσεων του Γ.Α. χωρίς να "χαλάσουμε" τις καλές λύσεις. Σημαντικό ρόλο στην επιλογή της πιθανότητας μετάλλαξης παίζει η κωδικοποίηση. Όταν χρησιμοποιείται η δυαδική κωδικοποίηση η πιθανότητα μετάλλαξης είναι συνήθως ανάμεσα σε 0.001 και 0.01 (πολλές φορές στην περίπτωση που ο αλγόριθμος μετάλλαξης είναι ο flip bit, η πιθανότητα μετάλλαξης είναι 1/L όπου L ο αριθμός των γονιδίων του ατόμου), ενώ σε άλλες μορφές κωδικοποίησης μπορεί να είναι αρκετά μεγαλύτερη.

Σε έρευνα που έγινε (Schaffer J.D., 1989) φαίνεται ότι υπάρχει κάποια σχέση ανάμεσα στο μέγεθος του πληθυσμού και την πιθανότητα μετάλλαξης. Πιο συγκεκριμένα όταν και τα δύο είναι πολύ μεγάλα (π.χ. Pop_size > 200 και Pmutation > 0.05) ή όταν και τα δύο είναι πολύ μικρά (π.χ. Pop_size < 20 και Pmutation < 0.002) η απόδοση του γενετικού αλγορίθμου μειώνεται.

Σε πολλές υλοποιήσεις γενετικών αλγορίθμων η πιθανότητα μετάλλαξης αλλάζει από γενιά σε γενιά. Ο γενετικός αλγόριθμος ξεκινάει με υψηλή πιθανότητα μετάλλαξης η οποία σταδιακά μειώνεται καθώς στην αρχή θέλουμε ο γενετικός αλγόριθμος να κάνει εξερεύνηση σε πολλές κατευθύνσεις, ενώ όσο πλησιάζουμε προς το τέλος και οι καλές λύσεις έχουν σχηματιστεί, θέλουμε να γίνονται μικρές αλλαγές στην περίπτωση που ο αλγόριθμος έχει εγκλωβιστεί σε ένα τοπικό ακρότατο (Ahmad Hassanat, 2019) (Dassanayake, 2015).

Εκτός από τις υλοποιήσεις στις οποίες η πιθανότητα μετάλλαξης εξαρτάται από τη γενιά, υπάρχουν και υλοποιήσεις στις οποίες το κάθε άτομο έχει διαφορετική πιθανότητα μετάλλαξης ανάλογα με την απόδοσή του. Θα δούμε αναλυτικά δύο από αυτές στο κεφάλαιο "Δυναμική πιθανότητα μετάλλαξης και διασταύρωσης". Η πρώτη (Min Dong, 2009) υλοποίηση δίνει μεγαλύτερη πιθανότητα μετάλλαξης στα άτομα με καλύτερη απόδοση, ώστε να αυξηθεί η ποικιλομορφία του πληθυσμού και να μη γεμίσει με αντίγραφα του ίδιου ατόμου. Η δεύτερη (S.N.Deepa, 2008) υλοποίηση δίνει στα άτομα με απόδοση κάτω από το μέσο όρο απόδοσης του πληθυσμού, μεγαλύτερη πιθανότητα μετάλλαξης καθώς τα άτομα αυτά είναι λιγότερο πιθανό να περιέχουν κάποια καλά γονίδια.

Στη συνέχεια θα δούμε αναλυτικά πολλούς αλγόριθμους μετάλλαξης για διαφορετικές μορφές κωδικοποίησης, αν και οι περισσότεροι αφορούν τους γενετικούς αλγορίθμους στους οποίους χρησιμοποιείται η κωδικοποίηση μετάθεσης. Στην κωδικοποίηση μετάθεσης δεν μας ενδιαφέρουν τα γονίδια που υπάρχουν στο άτομο, αλλά η σειρά τους. Έτσι είναι αναγκαία η δημιουργία αλγορίθμων

μετάλλαξης οι οποίοι θα αλλάζουν τη σειρά των γονιδίων και όχι τα γονίδια. Περισσότερους αλγορίθμους μετάλλαξης μπορείτε να βρείτε εδώ (Siew Mooi Lim, 2017).

8.1 Μετάλλαξη αντιστροφής bit - Bit flip mutation

Οι πρώτοι γενετικοί αλγόριθμοι χρησιμοποιούσαν αποκλειστικά δυαδική κωδικοποίηση και η μετάλλαξη αντιστροφής bit ήταν η πρώτη μορφή μετάλλαξης. Όλος ο πληθυσμός θεωρείται ως μια ακολουθία από bit και κάθε bit έχε πιθανότητα P_m να αντιστραφεί. Τα πρώτα χρόνια η πιθανότητα μετάλλαξης ήταν $P_m = 1/l$, όπου l ο αριθμός των γονιδίων του χρωμοσώματος, αλλά στη συνέχεια οι ερευνητές πειραματίστηκαν με διαφορετικές τιμές. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Ενώνουμε όλα τα άτομα του πληθυσμού, σχηματίζοντας έτσι μια ακολουθία από bit και θέτουμε i=1.

Βήμα 2ο: Παράγουμε έναν τυχαίο αριθμό $r \in [0,1)$. Αν $r < P_m$ το bit που βρίσκεται στη θέση i αντιστρέφεται. Αυξάνουμε το i κατά 1 i=i+1.

Βήμα 3ο: Αν i>N όπου N το μήκος της ακολουθίας από bit, ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.

Ένα από τα μειονεκτήματα της συγκεκριμένης μεθόδου μετάλλαξης είναι ότι απαιτεί την παραγωγή ενός τυχαίου αριθμού για κάθε bit. Για να κάνει τον αλγόριθμο πιο γρήγορο μειώνοντας τους τυχαίους αριθμούς που χρειάζονται, ο Goldberg (D.E., 1989) δημιούργησε μια τεχνική για να υπολογίσει πια bit θα αντιστραφούν την οποία ονόμασε mutation clock. Με τη μέθοδο mutation clock για να βρούμε το επόμενο bit που θα αντιστραφεί, παράγουμε ένα τυχαίο αριθμό $r \in [0,1)$ και μετακινούμαστε $\eta = -P_m \ln(1-r)$ bits στην ακολουθία των bit.

Μια άλλη τεχνική για να αυξήσουμε την ταχύτητα του αλγορίθμου, είναι να πάρουμε ένα τυχαίο άτομο από τον πληθυσμό με πιθανότητα μετάλλαξης P_m, να παράγουμε ένα τυχαίο αριθμό $\rho \in [1, N]$ όπου N ο αριθμός των γονιδίων του ατόμου και να αντιστρέψουμε το bit που βρίσκεται στη θέση N. Το βασικό μειονέκτημα αυτής της μεθόδου ότι σε κάθε άτομο μπορεί να συμβεί μόνο μια αντιστροφή bit.

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.2 Μετάλλαξη ορίων - Boundary mutation

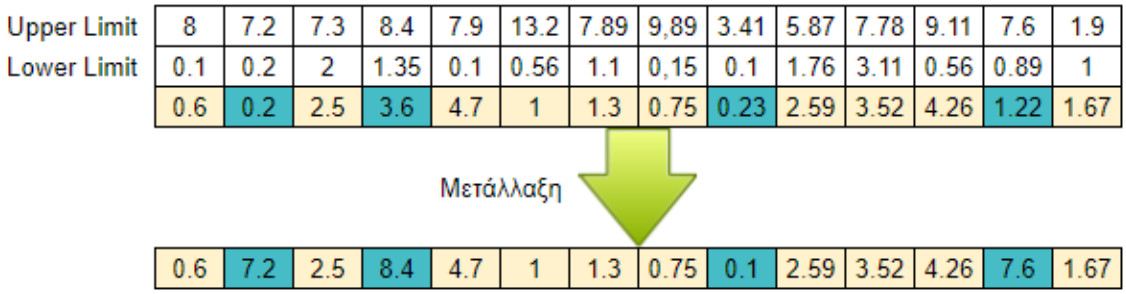
Ο αλγόριθμος της μετάλλαξης ορίων (Michalewicz, 1992) δημιουργήθηκε για τους γενετικούς αλγορίθμους οι οποίοι χρησιμοποιούν κωδικοποίηση αριθμών κινητής υποδιαστολής, είτε κωδικοποίηση ακεραίων αριθμών. Κάθε γονίδιο έχει πιθανότητα μετάλλαξης P_m. Αν το γονίδιο επιλεγεί για μετάλλαξη τότε παίρνει τυχαία ή την ανώτατη U_B ή την κατώτατη τιμή L_B που μπορεί να πάρει το συγκεκριμένο γονίδιο. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Ενώνουμε όλα τα άτομα του πληθυσμού, σχηματίζοντας έτσι μια ακολουθία γονιδίων και θέτουμε i=1.

Βήμα 2ο: Παράγουμε έναν τυχαίο αριθμό $r \in [0,1)$. Αν $r < P_m$ πηγαίνουμε στο Βήμα 3. Αν $r \geq P_m$ πηγαίνουμε στο Βήμα 4.

Βήμα 3ο: Παράγουμε έναν τυχαίο αριθμό k που μπορεί να πάρει τις τιμές 0 ή 1. Αν k=0 το γονίδιο που βρίσκεται στη θέση i παίρνει την ανώτατη τιμή U_B που μπορεί να πάρει. Αν k=1 το γονίδιο που βρίσκεται στη θέση i παίρνει την κατώτατη τιμή L_B που μπορεί να πάρει.

Βήμα 4ο: Αυξάνουμε το i κατά 1 i=i+1. Αν i>N όπου N το μήκος της ακολουθίας γονιδίων, ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.



Εικόνα 150 Μετάλλαξη ορίων για κωδικοποίηση με αριθμούς κινητής υποδιαστολής

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.3 Ομοιόμορφη μετάλλαξη - Uniform mutation

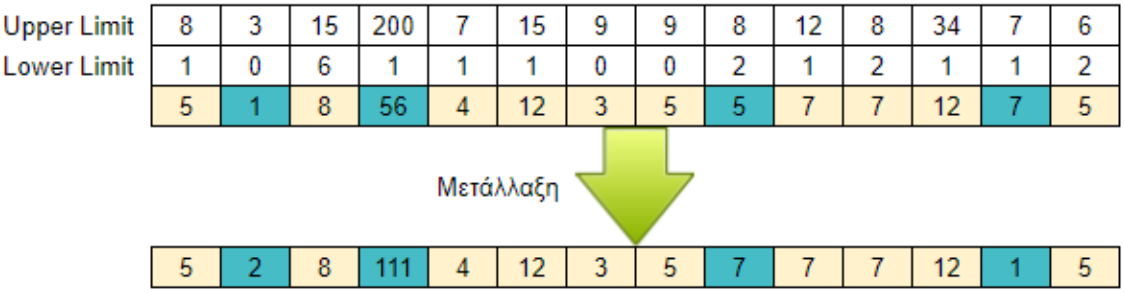
Ο αλγόριθμος της ομοιόμορφης μετάλλαξης (Michalewicz, 1992) δημιουργήθηκε για τους γενετικούς αλγορίθμους οι οποίοι χρησιμοποιούν κωδικοποίηση αριθμών κινητής υποδιαστολής, είτε κωδικοποίηση ακεραίων αριθμών.

Κάθε γονίδιο έχει πιθανότητα μετάλλαξης P_m. Αν το γονίδιο επιλεγεί για μετάλλαξη τότε παίρνει τυχαία οποιαδήποτε τιμή ανάμεσα στην κατώτατη και την ανώτατη τιμή που μπορεί να πάρει το γονίδιο. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Ενώνουμε όλα τα άτομα του πληθυσμού, σχηματίζοντας έτσι μια ακολουθία γονιδίων και θέτουμε i=1.

Βήμα 2ο: Παράγουμε έναν τυχαίο αριθμό $r \in [0,1)$. Αν $r < P_m$ το γονίδιο που βρίσκεται στη θέση i παίρνει τυχαία οποιαδήποτε τιμή ανάμεσα στην κατώτατη και την ανώτατη τιμή που μπορεί να πάρει το γονίδιο.

Βήμα 3ο: Αυξάνουμε το i κατά 1 i=i+1. Αν i>N όπου N το μήκος της ακολουθίας γονιδίων, ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.



Εικόνα 151 Ομοιόμορφη / τυχαία μετάλλαξη για κωδικοποίηση με ακέραιους αριθμούς

Όταν ο γενετικός αλγόριθμος χρησιμοποιεί κωδικοποίηση με ακέραιους αριθμούς η ομοιόμορφη μετάλλαξη ονομάζεται “τυχαία μετάλλαξη” (random mutation). Ο αλγόριθμος random mutation χρησιμοποιείται όταν όλοι οι ακέραιοι έχουν την ίδια πιθανότητα εμφάνισης (cardinal attributes). Για παράδειγμα αν ένα ρομπότ μπορεί να κινηθεί (μπροστά, πίσω, αριστερά, δεξιά) = (0,1,2,3) δεν υπάρχει καμία συσχέτιση ανάμεσα στους ακεραίους. Αντίθετα όταν υπάρχει συσχέτιση (ordinal attributes) π χ (πολύ μικρό, μικρό, μεσαίο, μεγάλο, πολύ μεγάλο) = (0,1,2,3,4), συνήθως χρησιμοποιείται ο αλγόριθμος creep mutation.

Η υλοποίηση του αλγόριθμου σε php βρίσκεται εδώ.

8.4 Μη ομοιόμορφη μετάλλαξη - Non-uniform mutation

Ο αλγόριθμος της μη ομοιόμορφης μετάλλαξης (Michalewicz, 1992) δημιουργήθηκε για τους γενετικούς αλγορίθμους οι οποίοι χρησιμοποιούν κωδικοποίηση αριθμών κινητής υποδιαστολής, είτε κωδικοποίηση ακεραίων αριθμών.

Στις πρώτες γενιές του γενετικού αλγορίθμου το γονίδιο που μεταλλάσσεται μπορεί να πάρει όλες τις τιμές ανάμεσα στο κατώτατο και το ανώτατο όριο με την ίδια πιθανότητα. Όσο πλησιάζουμε προς το τέλος του γενετικού αλγορίθμου η μετάλλαξη ενός γονιδίου προκαλεί πολύ μικρές αλλαγές στην τιμή του. Όταν ένα γονίδιο k επιλεγεί για μετάλλαξη η τιμή που θα πάρει δίνεται με την ίδια πιθανότητα χρησιμοποιώντας έναν από τους δύο τύπους.

$$x_k^m = x_k + (U_k - x_k) \cdot [1 - r^{(1-t/T)^b}] \quad (1)$$

$$x_k^m = x_k - (x_k - L_k) \cdot [1 - r^{(1-t/T)^b}] \quad (2)$$

- U_k η ανώτατη τιμή που μπορεί να πάρει το γονίδιο.
- L_k η κατώτατη τιμή που μπορεί να πάρει το γονίδιο.
- x_k η τιμή του γονιδίου πριν τη μετάλλαξη.
- r ένας τυχαίος αριθμός με $r \in [0, 1]$.
- T ο μέγιστος αριθμός γενεών.
- t η τρέχουσα γενιά.
- b σταθερά που συνήθως παίρνει τιμές $b \in [2, 5]$

Εικόνα 152 Τύποι ομοιόμορφης μετάλλαξης

Βήμα 1ο: Ενώνουμε όλα τα άτομα του πληθυσμού, σχηματίζοντας έτσι μια ακολουθία γονιδίων και θέτουμε i=1.

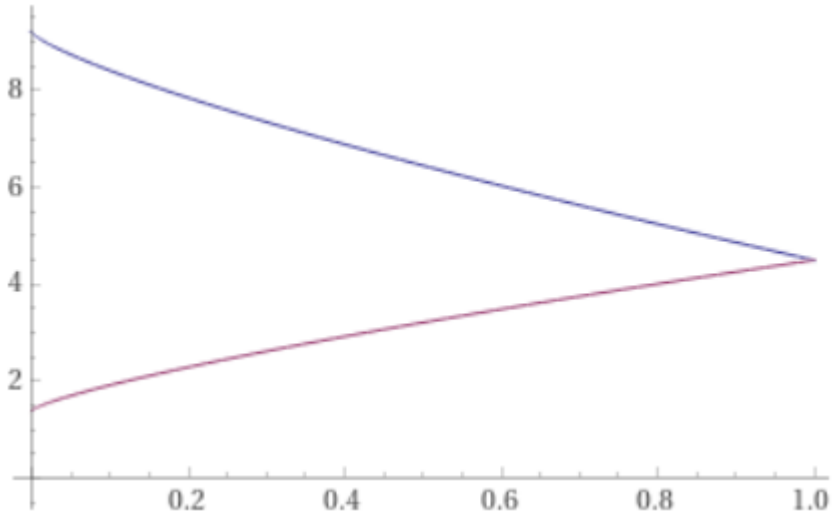
Βήμα 2ο: Παράγουμε έναν τυχαίο αριθμό $r \in [0,1)$. Αν $r < P_m$ πηγαίνουμε στο Βήμα 3. Αν $r \geq P_m$ πηγαίνουμε στο Βήμα 4.

Βήμα 3ο: Παράγουμε έναν τυχαίο αριθμό k που μπορεί να πάρει τις τιμές 0 ή 1. Αν k=0 το γονίδιο που βρίσκεται στη θέση i μεταλλάσσεται σύμφωνα με τη σχέση (1) . Αν k=1 το γονίδιο που βρίσκεται στη θέση i μεταλλάσσεται σύμφωνα με τη σχέση (2).

Βήμα 4ο: Αυξάνουμε το i κατά 1 i=i+1. Αν i>N όπου N το μήκος της ακολουθίας γονιδίων, ο αλγόριθμος τερματίζει. Επιστρέφουμε στο Βήμα 2.

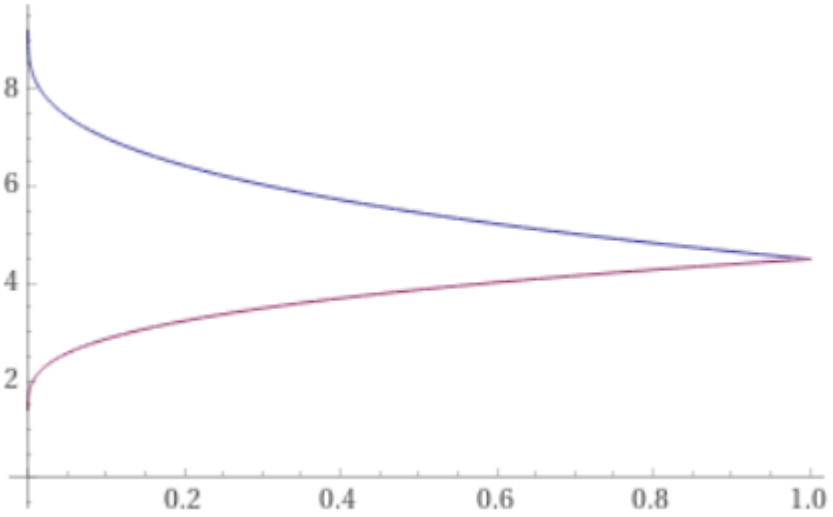
Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα $x = [x_1, x_1, \dots, x_k = 4.5, \dots, x_n]$ και επιλέγεται για μετάλλαξη το γονίδιο x_k με τιμή 4.5, ανώτερη τιμή $U_k = 9.2$ και κατώτερη τιμή $L_k = 1.4$. Ο μέγιστος αριθμός γενεών είναι 100.

- Για t=5, b=5 και r=0.5 το x_k μπορεί να πάρει τις τιμές
  - $x_k^m = 4.5 + (9.2 - 4.5) * \left[1 - 0.5^{(1-5/100)^5}\right] = 6.45105$
  - $x_k^m = 4.5 - (4.5 - 1.4) * \left[1 - 0.5^{(1-5/100)^5}\right] = 3.21314$



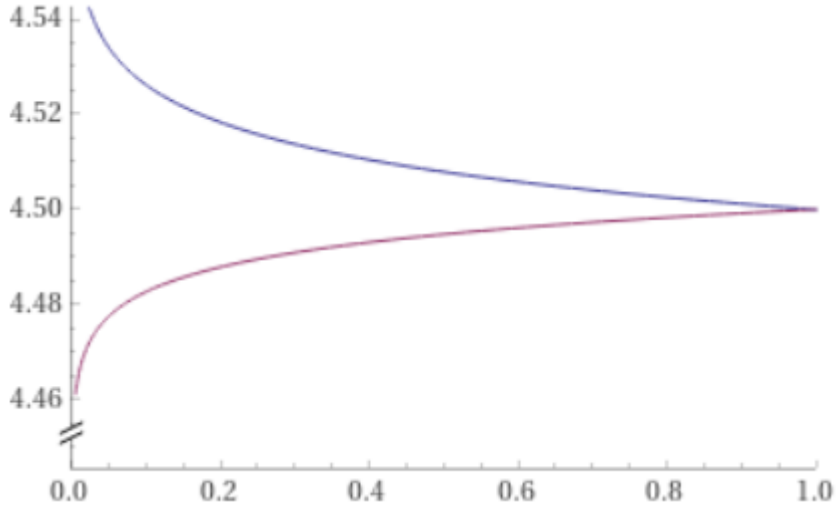
Εικόνα 153 Τιμές που μπορεί να πάρει το μεταλλαγμένο γονίδιο με t=5 για διάφορες τιμές του r

- Για t=20, b=5 και r=0.5 το x_k μπορεί να πάρει τις τιμές
  - $x_k^m = 4.5 + (9.2 - 4.5) * \left[1 - 0.5^{(1-20/100)^5}\right] = 5.45496$
  - $x_k^m = 4.5 - (4.5 - 1.4) * \left[1 - 0.5^{(1-20/100)^5}\right] = 3.87013$



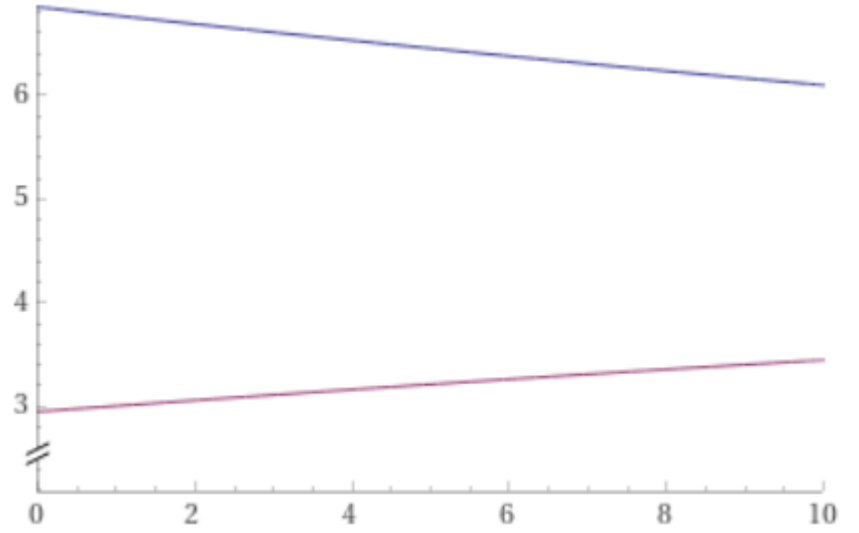
Εικόνα 154 Τιμές που μπορεί να πάρει το μεταλλαγμένο γονίδιο με t=20 για διάφορες τιμές του r

- Για t=70, b=5 και r=0.5 το x_k μπορεί να πάρει τις τιμές
  - $x_k^m = 4.5 + (9.2 - 4.5) * \left[1 - 0.5^{(1-70/100)^5}\right] = 4.50791$
  - $x_k^m = 4.5 - (4.5 - 1.4) * \left[1 - 0.5^{(1-70/100)^5}\right] = 4.49478$



Εικόνα 155 Τιμές που μπορεί να πάρει το μεταλλαγμένο γονίδιο με t=70 για διάφορες τιμές του r

- Για t=5, b=3 και r=0.5 το x_k μπορεί να πάρει τις τιμές: (Όταν αυξάνουμε την τιμή του b, μειώνεται το εύρος των τιμών που μπορεί να πάρει το x_k^m).
  - $x_k^m = 4.5 + (9.2 - 4.5) * \left[1 - 0.5^{(1-5/100)^3}\right] = 6.60581$
  - $x_k^m = 4.5 - (4.5 - 1.4) * \left[1 - 0.5^{(1-5/100)^3}\right] = 3.11106$



Εικόνα 156 Τιμές που μπορεί να πάρει το μεταλλαγμένο γονίδιο με t=5, r=0.5 για διάφορες τιμές του b

Σημείωση: Αν χρησιμοποιήσουμε τη μη ομοιόμορφη μετάλλαξη όταν τα γονίδια παίρνουν μόνο ακέραιες τιμές, παίρνουμε τυχαία τον πλησιέστερο προς τα πάνω ή προς τα κάτω ακέραιο.

Η υλοποίηση του αλγόριθμου σε php βρίσκεται εδώ.

8.5 Creep mutation

Ο αλγόριθμος creep mutation (S.N.Deepa, 2008) δημιουργήθηκε για τους γενετικούς αλγορίθμους οι οποίοι χρησιμοποιούν κωδικοποίηση αριθμών κινητής υποδιαστολής, είτε κωδικοποίηση ακεραίων αριθμών.

Μπορεί να θεωρηθεί ως ένας αλγόριθμος μη ομοιόμορφης μετάλλαξης καθώς το γονίδιο που μεταλλάσσεται δεν παίρνει όλες τις πιθανές τιμές με την ίδια πιθανότητα, αλλά ακολουθεί κάποια κατανομή (εκθετική, κανονική-γκαουσιανή κατανομή, κατανομή Cauchy, διωνυμική κατανομή κτλ.)

Στην έρευνά τους οι Yao et. al. (1999) (Xin Yao, 1999) προτείνουν το συνδυασμό της μετάλλαξης Gauss και Cauchy. Η μετάλλαξη Gauss ευνοεί την τοπική αναζήτηση, δηλαδή η μετάλλαξη ενός γονιδίου έχει μεγαλύτερη πιθανότητα να μας δώσει κοντινές τιμές, ενώ η μετάλλαξη Cauchy είναι καλύτερη όταν η τιμή του γονιδίου βρίσκεται πολύ μακριά από το ολικό βέλτιστο. Επειδή όμως τις περισσότερες φορές δε γνωρίζουμε το ολικό βέλτιστο οι Yao et. al. πρότειναν να γίνεται η μετάλλαξη και με τις δύο μεθόδους και να επιβιώνει το μεταλλαγμένο άτομο με την καλύτερη απόδοση.

Οι Chen, M.R., Lu, Y.Z (2008) (Chen, 2008) πρότειναν έναν άλλο τρόπο συνδυασμού της μετάλλαξης Gauss και Cauchy. Η μετάλλαξη ξεκινάει ακολουθώντας κατανομή Cauchy και οι δύο κατανομές εναλλάσσονται.

Μια προσεγγιστική υλοποίηση της μεθόδου με κανονική κατανομή βρίσκεται εδώ.

8.6 Μετάλλαξη με αντιστροφή - Reverse Sequence Mutation (RSM) - Simple inversion mutation (SIM)

Στην μετάλλαξη Reverse Sequence Mutation (RSM) (Holland, 1975) παράγουμε δύο τυχαίους αριθμούς M1 και M2 στο διάστημα [1,n] όπου n ο αριθμός των γονιδίων και αντιστρέφουμε τη θέση των γονιδίων που βρίσκονται σε αυτό το διάστημα. Ο αλγόριθμος μετάλλαξης RSM χρησιμοποιείται κυρίως όταν ο Γ.Α. χρησιμοποιεί κωδικοποίηση μετάθεσης. Ο αλγόριθμος μετάλλαξης RSM περιγράφεται στα παρακάτω βήματα

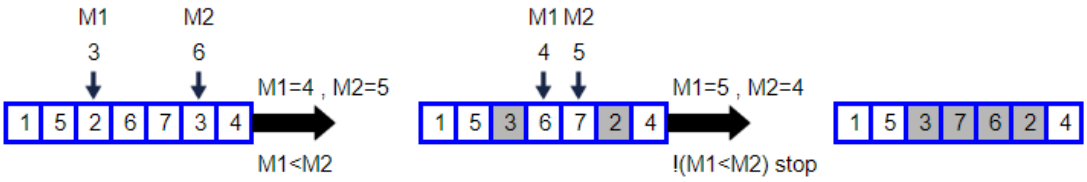
Βήμα 1ο: Παίρνουμε δύο τυχαίους αριθμούς M1 και M2 στο διάστημα [1,n] όπου n ο αριθμός των γονιδίων. Αν M2 < M1 τότε ο M1 παίρνει την τιμή του M2 και ο M2 την τιμή του M1.

Βήμα 2ο: Αλλάζουμε θέση μεταξύ των γονιδίων που βρίσκονται στις θέσεις M1 και M2.

Βήμα 3ο: Αυξάνουμε την τιμή του M1 κατά 1 και μειώνουμε την τιμή του M2 κατά 1.

Βήμα 4ο: Αν M1<M2 επιστρέφουμε στο βήμα 2. Διαφορετικά ο αλγόριθμος τερματίζει.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα (1, 5, 2, 6, 7, 3, 4) και M1=3, M2=6. Στη θέση 3 βρίσκεται το γονίδιο 2 και στη θέση 6 το γονίδιο 3. Αλλάζουν θέσεις και παίρνουμε το χρωμόσωμα (1, 5, 3, 6, 7, 2, 4). Το M1 γίνεται 4 και το M2 γίνεται 5. Στη θέση 4 βρίσκεται το γονίδιο 6 και στη θέση 5 το γονίδιο 7. Αλλάζουν θέσεις και προκύπτει το γονίδιο (1, 5, 3, 7, 6, 2, 4). Το M1 είναι 5 και το M2 είναι 4. Δεν ισχύει πλέον M1<M2 οπότε ο αλγόριθμος τερματίζει.



Εικόνα 157 Reverse sequence mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

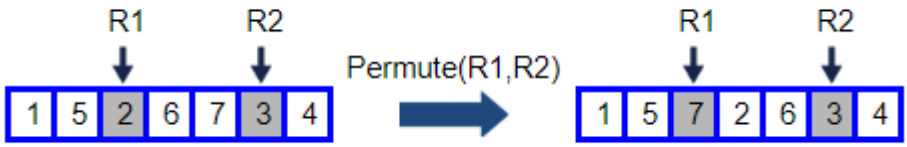
8.7 Partial shuffle mutation (PSM) - Scramble mutation

Η μέθοδος μετάλλαξης partial shuffle mutation (PSM), Syswerda 1991 (Syswerda G. , 1991) κάνει ότι ακριβώς και η μέθοδος RSM με την μόνη η διαφορά ότι το τα γονίδια που βρίσκονται στο ενδιάμεσο τμήμα που παράγεται από τους δύο τυχαίους αριθμούς δεν αντιστρέφονται, αλλά παίρνουμε μια τυχαία μετάθεσή τους. Ο αλγόριθμος της μετάλλαξης PSM περιγράφεται στα παρακάτω βήματα.

Βήμα 1: Παράγουμε δύο τυχαίους αριθμούς R1 και R2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων. Αν R1>R2 τότε ο R1 παίρνει την τιμή του R2 και ο R2 την τιμή του R1.

Βήμα 2: Παράγουμε μια τυχαία μετάθεση των γονιδίων που βρίσκονται μεταξύ R1 και R2.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] . Παίρνουμε τους δύο τυχαίους αριθμούς R1=3 και R2=6. Μεταξύ αυτών των δύο αριθμών βρίσκονται τα γονίδια 2, 6, 7, και 3. Μια τυχαία μετάθεσή τους είναι η 7, 2, 6 και 3. Έτσι προκύπτει το χρωμόσωμα [1, 5, 7, 2, 6, 3, 4].



Εικόνα 158 Partial shuffle mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

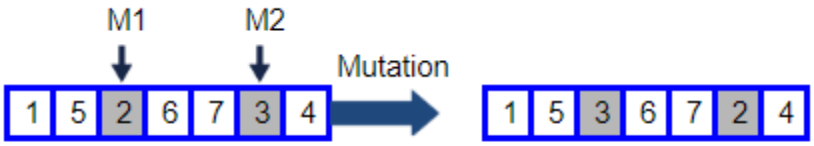
8.8 Μετάλλαξη με ανταλλαγή - Twors Mutation - Exchange Mutation (EM) - Swap Mutation - Point mutation

Τη μετάλλαξη με ανταλλαγή τη συναντάμε με πολλές ονομασίες (P. Larrañaga, 1999). Είναι ένας πολύ απλός αλγόριθμος μετάλλαξης στον οποίο δύο γονίδια ανταλλάσσουν θέση. Ο αλγόριθμος περιγράφεται αναλυτικά στα παρακάτω βήματα.

Βήμα 1ο: Επιλέγονται δύο τυχαίοι αριθμοί M1 και M2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων.

Βήμα 2ο: Τα δυο γονίδια που βρίσκονται στις θέσεις M1 και M2 αλλάζουν θέση.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα (1, 5, 2, 6, 7, 3, 4) και M1=3, M2=6. Στη θέση 3 βρίσκεται το γονίδιο 2 και στη θέση 6 βρίσκεται το γονίδιο 3. Αλλάζουν θέσεις και παίρνουμε το χρωμόσωμα (1, 5, 3, 6, 7, 2, 4).



Εικόνα 159 Swap mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.9 Thoros mutation

Η μετάλλαξη Thoros (Hsien-Pin Hsu, 2019) είναι παρόμοια με τη μετάλλαξη με ανταλλαγή. Η μόνη διαφορά είναι ότι αλλάζουν θέση τρία γονίδια αντί για δύο. Ο αλγόριθμος περιγράφεται αναλυτικά από τα παρακάτω βήματα.

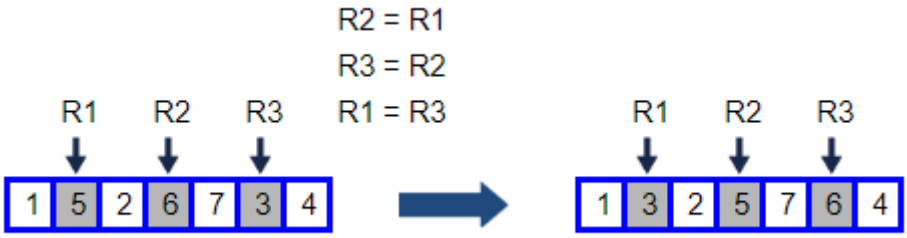
Βήμα 1ο: Παράγουμε 3 τυχαίους αριθμούς R1, R2, R3 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων του χρωμοσώματος.

Βήμα 2ο: Το γονίδιο που βρίσκεται στη θέση R2 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R1.

Βήμα 3ο: Το γονίδιο που βρίσκεται στη θέση R3 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R2.

Βήμα 4ο: Το γονίδιο που βρίσκεται στη θέση R1 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R3.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τους τυχαίους αριθμούς R1=2, R2=4 και R3=6. Το γονίδιο στην θέση 4 θα γίνει 5, το γονίδιο στην θέση 6 θα γίνει 6 και το γονίδιο στην θέση 2 θα γίνει 3 και προκύπτει το χρωμόσωμα [1, 3, 2, 5, 7, 6, 4].



Εικόνα 160 Thoros mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.10 Thoras mutation

Όπως και στη μετάλλαξη Thoros έτσι και στη μετάλλαξη Thoras (Hsien-Pin Hsu, 2019) τρία γονίδια ανταλλάσσουν θέσεις. Η μόνη διαφορά είναι ότι τα γονίδια που αλλάζουν θέση δε βρίσκονται διάσπαρτα, αλλά σε γειτονικές θέσεις. Ο αλγόριθμος περιγράφεται αναλυτικά στα παρακάτω βήματα.

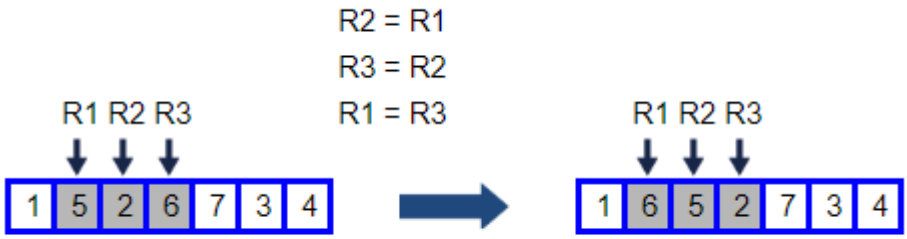
Βήμα 1ο: Παράγουμε τον τυχαίο αριθμό R1 στο διάστημα [1,N-2], όπου N ο αριθμός των γονιδίων του χρωμοσώματος και θέτουμε R2=R1+1 και R3=R2+1.

Βήμα 2ο: Το γονίδιο που βρίσκεται στη θέση R2 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R1.

Βήμα 3ο: Το γονίδιο που βρίσκεται στη θέση R3 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R2.

Βήμα 4ο: Το γονίδιο που βρίσκεται στη θέση R1 παίρνει την τιμή του γονιδίου που βρίσκεται στη θέση R3.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τον τυχαίο αριθμό R1=2, οπότε έχουμε R2=3 και R3=4. Το γονίδιο στην θέση 3 θα γίνει 5, το γονίδιο στην θέση 4 θα γίνει 2 και το γονίδιο στην θέση 2 θα γίνει 6 και προκύπτει το χρωμόσωμα [1, 6, 5, 2, 7, 3, 4].



Εικόνα 161 Thoras mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.11 Μετάλλαξη με μετατόπιση - Displacement mutation (DM)

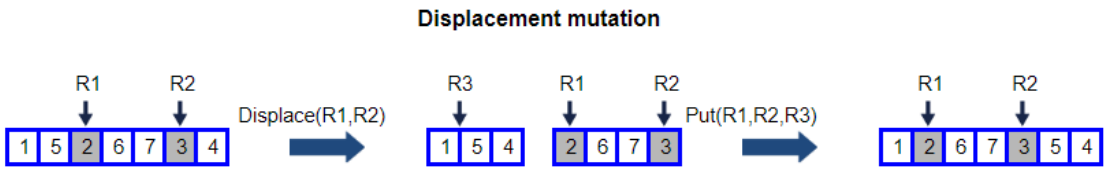
Στην μετάλλαξη με μετατόπιση (Michalewicz 1992) (Michalewicz, 1992) παίρνουμε δύο τυχαίους αριθμούς R1, R2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων. Παράγουμε έναν ακόμα τυχαίο αριθμό R3 και τοποθετούμε τα γονίδια που βρίσκονται μεταξύ R1, R2 στη θέση R3. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Παράγουμε δύο τυχαίους αριθμούς R1 και R2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων. Αν R1>R2 τότε ο R1 παίρνει την τιμή του R2 και ο R2 την τιμή του R1.

Βήμα 2ο: Αφαιρούμε τα γονίδια που βρίσκονται στο διάστημα [R1,R2] του χρωμοσώματος, τα τοποθετούμε σε μια λίστα L1, ενώνουμε τα δύο τμήματα που περιέχουν τα γονίδια που δεν βρίσκονται στο διάστημα [R1,R2] και τα τοποθετούμε σε μια λίστα L2.

Βήμα 3ο: Παράγουμε έναν τυχαίο αριθμό R3στο διάστημα [1,K+1] όπου K το μήκος της λίστας L2 και τοποθετούμε τη λίστα L1 στο σημείο R3.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τους τυχαίους αριθμούς R1=3και R2=6. Αφαιρούμε τα γονίδια που βρίσκονται μεταξύ των θέσεων 3 και 6 δηλαδή τα γονίδια 2, 6, 7 και 3 και ενώνουμε τα άλλα γονίδια παίρνοντας τη λίστα 1, 5, 4. Παράγουμε έναν ακόμα τυχαίο αριθμό στο διάστημα [1,4], έστω R3=2. Οπότε πρέπει να τοποθετήσουμε τη λίστα ανάμεσα στο 1 και στο 5 παίρνοντας το γονίδιο [1, 2, 6, 7, 3, 5, 4].



Εικόνα 162 Displacement mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.12 Μετάλλαξη με εισαγωγή - Insertion Mutation - Position Based Mutation Operator

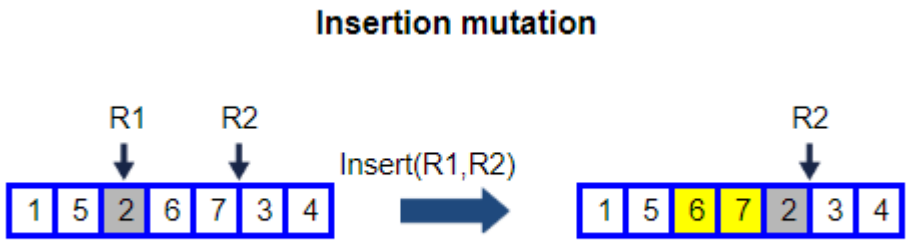
Η μετάλλαξη με εισαγωγή (Erna Budhiarti Nab, Genetic Algorithms Dynamic Population Size with Cloning in Solving Traveling Salesman Problem, 2018) μπορεί να θεωρηθεί ως μια ειδική περίπτωση της μετάλλαξης με μετατόπιση. Παράγουμε δύο τυχαίους αριθμούς $R_1 \neq R_2$ στο διάστημα [1,N] όπου N ο

αριθμός των γονιδίων και μεταφέρουμε το γονίδιο που βρίσκεται στη θέση R_1 στη θέση R_2. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Παράγουμε δύο τυχαίους αριθμούς $R_1 \neq R_2$ στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων.

Βήμα 2ο: Μεταφέρουμε το γονίδιο που βρίσκεται στη θέση R_1 στη θέση R_2.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τους τυχαίους αριθμούς $R_1 = 3$ και $R_2 = 5$. Μεταφέρουμε το χρωμόσωμα που βρίσκεται στη θέση 3 δηλαδή το 2 στη θέση 5.



Εικόνα 163 Insertion mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

8.13 Cut-inverse mutation operator - Inverted Displacement mutation (IDM)

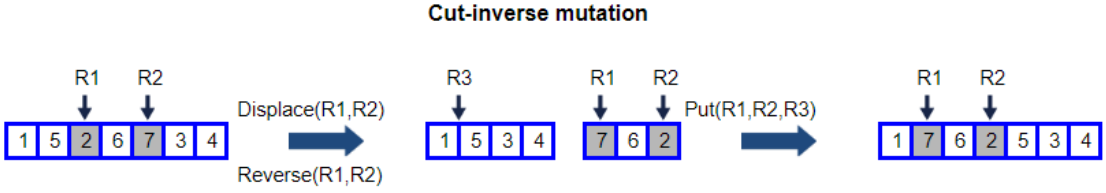
Η μέθοδος Cut-inverse mutation (Banzhaf, 1990) αποτελεί συνδυασμό των μεθόδων simple inversion mutation και displacement mutation και περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Παράγουμε δύο τυχαίους αριθμούς R1, R2 στο διάστημα [1,N] όπου N ο αριθμός των γονιδίων.

Βήμα 2ο: Αντιστρέφουμε τα γονίδια που βρίσκονται στο διάστημα [R1,R2].

Βήμα 3ο: Ακολουθούμε τα βήματα της μετάλλαξης με μετατόπιση από το Βήμα 2.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τους τυχαίους αριθμούς R1=3 και R2=5. Μεταξύ των R1, R2 βρίσκονται τα γονίδια 2, 6 και 7. Τα αντιστρέφουμε 7, 6, 2. Παράγουμε τον τυχαίο αριθμό 2 στο διάστημα 1,5 και τοποθετούμε τα γονίδια 7, 6, 2 στη θέση 2 της λίστας [1, 5, 3, 4] παίρνοντας το χρωμόσωμα [1, 7, 6, 2, 5, 3, 4].



Εικόνα 164 Cut-Inverse mutation

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

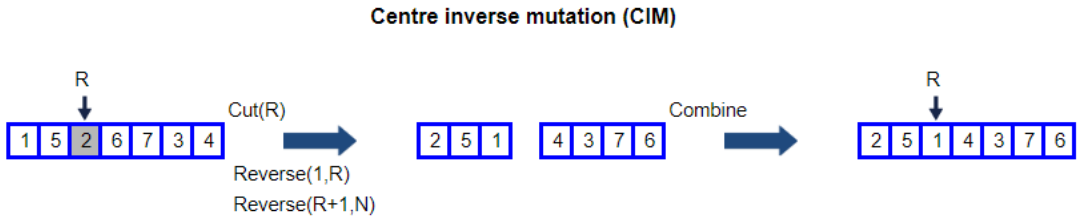
8.14 Centre inverse mutation (CIM)

Ο αλγόριθμος μετάλλαξης centre inverse mutation (Hsien-Pin Hsu, 2019) χωρίζει το χρωμόσωμα σε δύο μέρη και αντιστρέφει τη σειρά των γονιδίων. Ο αλγόριθμος περιγράφεται στα παρακάτω βήματα.

Βήμα 1ο: Επιλέγουμε ένα τυχαίο αριθμό R στο διάστημα 1,N όπου N ο αριθμός των γονιδίων και χωρίζουμε το χρωμόσωμα σε 2 μέρη A, B στο σημείο R.

Βήμα 2ο: Αντιστρέφουμε τα γονίδια που βρίσκονται στα δύο μέρη A,B και ενώνουμε τα δύο μέρη για πάρουμε το μεταλλαγμένο χρωμόσωμα.

Παράδειγμα: Έστω ότι έχουμε το χρωμόσωμα [1, 5, 2, 6, 7, 3, 4] και τον τυχαίο αριθμό R=3. Χωρίζουμε το χρωμόσωμα σε δύο μέρη τα 1, 5, 2 και 6, 7, 3, 4. Τα αντιστρέφουμε 2, 5, 1 και 4, 3, 7, 6. Τα ενώνουμε και παίρνουμε το χρωμόσωμα [2, 5, 1, 4, 3, 7, 6].



Εικόνα 165 Centre inverse mutation (CIM)

Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ και εδώ.

Η πρώτη υλοποίηση έγινε χωρίζοντας το array σε δύο στο σημείο R, με την array_reverse αντιστρέφουμε τα δύο μέρη και τα ενώνουμε με την array_merge. Στην δεύτερη υλοποίηση κάναμε δύο for loop αλλάζοντας θέση στα στοιχεία στις θέσεις 0+i και R-i στο πρώτο και R+i+1 και size-i-1 στο δεύτερο. Η δεύτερη υλοποίηση εκτελείται πιο γρήγορα.

9 Δυναμική πιθανότητα μετάλλαξης και διασταύρωσης

Αν και οι περισσότεροι Γ.Α. χρησιμοποιούν στατικές τιμές για την πιθανότητα μετάλλαξης και διασταύρωσης, έχουν γίνει αρκετές προτάσεις ώστε η πιθανότητα να αλλάζει δυναμικά ανάλογα με την απόδοση των ατόμων. Στη συνέχεια του κεφαλαίου θα δούμε δύο τρόπους με τους οποίους μπορεί να γίνει η αλλαγή.

Ο πρώτος τρόπος προτάθηκε από τους Min Dong, Yan Wu (Min Dong, 2009). Η πιθανότητα να γίνει διασταύρωση μεταξύ δύο χρωμοσωμάτων a και b είναι:

$$P_c = \frac{|f(a) - f(b)|}{\max(f) - \min(f)}$$

Στον τύπο f(a) είναι η απόδοση του χρωμοσώματος a, f(b) είναι η απόδοση του χρωμοσώματος b, max(f) είναι η μέγιστη απόδοση του πληθυσμού και min(f) η ελάχιστη.

Τα άτομα με πολύ μικρή ή πολύ μεγάλη απόδοση έχουν μεγαλύτερη πιθανότητα διασταύρωσης για να αποφευχθεί ο έντονος ανταγωνισμός στο στάδιο της επιλογής του επόμενου γύρου. Επίσης τα άτομα με παρόμοια απόδοση, επειδή έχουν παρόμοια γονίδια, έχουν πολύ μικρή πιθανότητα διασταύρωσης.

Η αντίστοιχη πιθανότητα μετάλλαξης ενός χρωμοσώματος δίνεται από τον τύπο:

$$P_m = k * \left(\frac{f(a)}{\max(f)}\right)^2$$

Στον τύπο f(a) είναι η απόδοση του χρωμοσώματος a, max(f) είναι η μέγιστη τιμή της απόδοσης του πληθυσμού και k μια σταθερά που παίρνει τιμές στο διάστημα (0,1). Στην έρευνά τους οι Min Dong, Yan Wu (Min Dong, 2009) παρατήρησαν ότι ο αλγόριθμος αποδίδει καλύτερα για μεγάλες τιμές του k.

Τα άτομα με καλύτερη απόδοση έχουν μεγαλύτερη πιθανότητα μετάλλαξης αποτρέποντας τον πληθυσμό να γεμίσει από αντίγραφα των καλύτερων ατόμων, αυξάνοντας έτσι την ποικιλομορφία.

Ο δεύτερος τρόπος προτάθηκε από τους S.N.Sivanandam, S.N.Deepa (S.N.Deepa, 2008). Η πιθανότητα διασταύρωσης δίνεται από τον τύπο:

$$P_c = \frac{(P_{c1} - P_{c2}) * (f' - f_{avg})}{f_{max} - f_{avg}}, f' \geq f_{avg}$$

$$P_c = P_{c1}, f' < f_{avg}$$

Στον τύπο f_{max} είναι μέγιστη τιμή της απόδοσης του πληθυσμού, f_{avg} είναι η μέση τιμή της απόδοσης του πληθυσμού, f' είναι η μέγιστη τιμή της απόδοσης ανάμεσα στα άτομα που θέλουμε να διασταυρώσουμε και P_{c1}, P_{c2}, σταθερές που παίρνουν συνήθως τιμές $P_{c1} = 0.9$, $P_{c2} = 0.6$.

Η αντίστοιχη πιθανότητα μετάλλαξης είναι:

$$P_m = \frac{(P_{m1} - P_{m2}) * (f - f_{avg})}{f_{max} - f_{avg}}, f \geq f_{avg}$$

$$P_m = P_{m1}, f < f_{avg}$$

Στον τύπο f_{max} είναι μέγιστη τιμή της απόδοσης του πληθυσμού, f_{avg} είναι η μέση τιμή της απόδοσης του πληθυσμού, f είναι η απόδοση του ατόμου προς μετάλλαξη και P_{m1}, P_{m1} σταθερές που συνήθως παίρνουν τιμές $P_{m1} = 0.1$, $P_{m2} = 0.001$.

Τα άτομα με απόδοση μικρότερη του μέσου όρου απόδοσης του πληθυσμού, έχουν μεγαλύτερη πιθανότητα μετάλλαξης, Αντίθετα τα άτομα με απόδοση μεγαλύτερη του μέσου όρου απόδοσης του πληθυσμού προστατεύονται.

10 Κριτήριο τερματισμού (Termination Condition)

Ο γενετικός αλγόριθμος επαναλαμβάνει τα βήματα αξιολόγησης, επιλογής, διασταύρωσης, μετάλλαξης μέχρι να εκπληρωθεί μια συνθήκη τερματισμού. Επειδή οι GA είναι αλγόριθμοι στοχαστικής φύσης δεν είναι βέβαιο ότι θα βρουν την βέλτιστη λύση. Επίσης είναι πολύ πιθανό να μη γνωρίζουμε τη βέλτιστη λύση ώστε να την χρησιμοποιήσουμε ως κριτήριο τερματισμού.

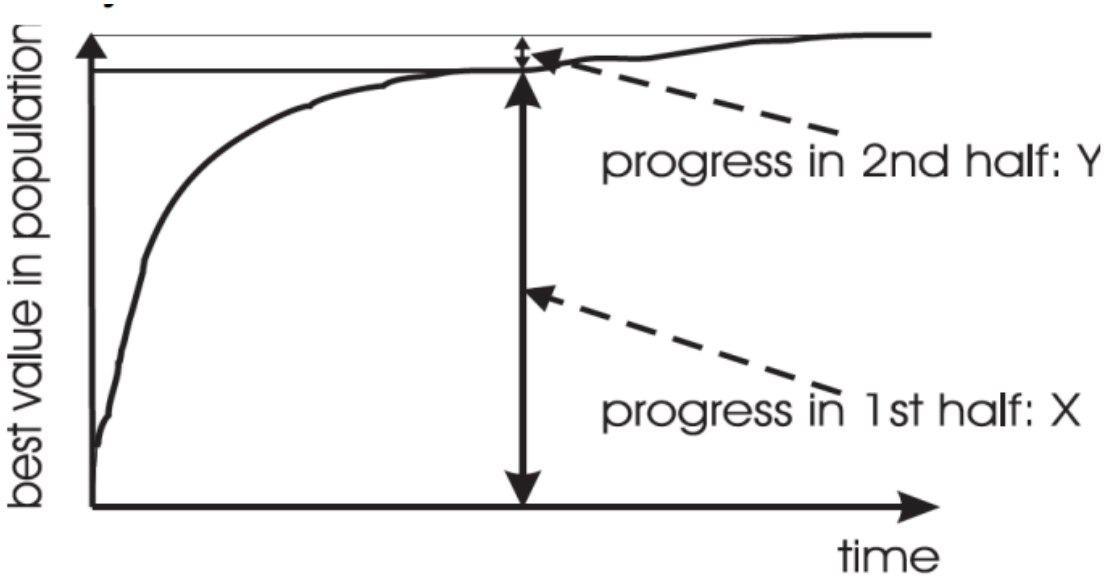
Αν ο γενετικός αλγόριθμος δεν μεταφέρει το καλύτερο άτομο στην επόμενη γενιά (elitism), είναι πιθανό η καλύτερη λύση να μη βρίσκεται στην τελευταία γενιά. Σε αυτή την περίπτωση καλό είναι να υπάρχει μια μεταβλητή fittest στην οποία θα αποθηκεύουμε το άτομο με την καλύτερη απόδοση.

Μερικές από τις πιο συνηθισμένες συνθήκες τερματισμού είναι η εξής:

- **Μέγιστος αριθμός γενεών**. Αποτελεί την πιο συνηθισμένη συνθήκη τερματισμού. Ο αλγόριθμος θα τερματίσει όταν περάσουν n γενιές με το n να δίνεται στην αρχή από τον χρήστη.
- **Χ γενιές χωρίς βελτίωση στην απόδοση** (με το Χ να καθορίζεται στην αρχή από τον χρήση). Ως απόδοση του πληθυσμού μπορεί να θεωρηθεί είτε η μέση απόδοση του πληθυσμού, είτε η καλύτερη λύση.
- **Εύρεση λύσης που πληροί κάποια κριτήρια**. Για παράδειγμα στο TSP μπορεί να μας αρκεί να βρούμε μια διαδρομή μήκους Χ μέτρων. Για να εφαρμοστεί αυτό το κριτήριο τερματισμού πρέπει να γνωρίζουμε τη βέλτιστη λύση, ή κάποιο όριο της (δηλαδή ότι υπάρχει λύση μήκους Χ μέτρων).
- **Χρονικός τερματισμός**: Τερματισμός όταν περάσουν Χ δευτερόλεπτα - λεπτά - ώρες. Είναι η δεύτερη πιο διαδεδομένη συνθήκη τερματισμού.
- **Παρεμβολή χρήστη**: Ο αλγόριθμος συνεχίζει να κάνει κύκλους μέχρι να τον διακόψει ο χρήστης.

- Σε κάποιες περιπτώσεις ο υπολογισμός της συνάρτησης καταλληλότητας είναι χρονοβόρος, οπότε ως κριτήριο τερματισμού μπορεί να τεθεί ο **υπολογισμός της καταλληλότητας Ν φορές**.
- **Σύγκλιση πληθυσμού**: Όταν η μέση τιμή της απόδοσης των ατόμων του πληθυσμού είναι Χ% φορές χειρότερη από την τιμή του ατόμου με την καλύτερη απόδοση. Σε αυτό το σημείο θεωρούμε ότι ο αλγόριθμος έχει συγκλίνει.
- **Μείωση ποικιλομορφίας**: Ο αλγόριθμος τερματίζει όταν ένα ποσοστό των ατόμων του πληθυσμού είναι ίδια. Σε αυτό το σημείο θεωρούμε ότι η μοναδική βελτίωση που μπορεί να έχουμε στην απόδοση μπορεί να προέλθει από τη μετάλλαξη.

Επειδή στα περισσότερα προβλήματα δε γνωρίζουμε τη βέλτιστη λύση, είναι δύσκολο να υπολογίσουμε πότε πρέπει να τερματίσουμε το γενετικό αλγόριθμο. Στις πρώτες γενιές η βελτίωση στην απόδοση του πληθυσμού είναι μεγάλη, αλλά όσο περνάει ο χρόνος η βελτίωση που παρατηρείται από γενιά σε γενιά είναι πολύ μικρή.



Εικόνα 166 Όσο περνάει ο χρόνος η βελτίωση στην απόδοση μειώνεται

Το για πόση ώρα θα εκτελέσουμε τον γενετικό αλγόριθμο εξαρτάται από πολλούς παράγοντες. Αν στόχος μας είναι η ακρίβεια και δεν υπάρχουν άλλοι ευρετικοί αλγόριθμοι για το πρόβλημα, τότε μπορεί να θέλουμε να αφήσουμε το γενετικό αλγόριθμο να τρέξει, για όσο περισσότερο χρόνο γίνεται. Αν υπάρχουν ευρετικοί αλγόριθμοι βελτιστοποίησης για το πρόβλημα ή αν στόχος μας είναι η ταχύτητα, τότε είναι καλύτερα να τον διακόψουμε νωρίτερα. Επειδή οι γενετικοί αλγόριθμοι είναι στοχαστικοί, ίσως να είναι καλύτερα για το πρόβλημά μας να κάνουμε πολλές μικρές επαναλήψεις.

11 Ολοκληρωμένο παράδειγμα Γ.Α.

Σε αυτό το κεφάλαιο θα δούμε ένα παράδειγμα του πρώτου κύκλου ενός γενετικού αλγορίθμου για το πρόβλημα του πλανόδιου πωλητή με τα εξής δεδομένα:

- Το μέγεθος του αρχικού πληθυσμού είναι 7 και παράγεται τυχαία.
- Η επιλογή γίνεται με τη μέθοδο τουρνουά με επανατοποθέτηση με K=2. Τα τουρνουά αυτού του είδους είναι τα πιο συνηθισμένα και ονομάζονται δυαδικά τουρνουά (binary tournaments).
- Το άτομο με την καλύτερη απόδοση σε κάθε γενιά μεταφέρεται στην επόμενη γενιά. Σε αυτή την περίπτωση λέμε ότι ο γενετικός αλγόριθμος χρησιμοποιεί τη μέθοδο του ελιτισμού (elitism).
- Η πιθανότητα διασταύρωσης είναι 100%.

- Η διασταύρωση γίνεται με τον αλγόριθμο modified crossover,
- Η πιθανότητα μετάλλαξης είναι 10%.
- Η μετάλλαξη γίνεται με τον αλγόριθμο μετάλλαξης με αντιστροφή.
- Η αναπαράσταση των λύσεων γίνεται με την κωδικοποίηση μετάθεσης.
- Η συνάρτηση καταλληλότητας είναι η f(X)=1/distance(X).
- Το πρόβλημα έχει 6 πόλεις.
- Ο πίνακας κόστους του προβλήματος φαίνεται στο σχήμα που ακολουθεί.

0	5	8	2	3	1
5	0	9	1	4	2
8	9	0	3	6	7
2	1	3	0	2	8
3	4	6	2	0	5
1	2	7	8	5	0

Εικόνα 167 Πίνακας κόστους

Στην αρχή δημιουργούμε τυχαία έναν αρχικό πληθυσμό. Στη συνέχεια επιλέγουμε 6 άτομα με τη μέθοδο της επιλογής τουρνουά με επανατοποθέτηση. Επιλέγουμε ένα άτομο λιγότερο από το μέγεθος του πληθυσμού καθώς θέλουμε να μεταφέρουμε το άτομο με την καλύτερη απόδοση της κάθε γενιάς στο νέο πληθυσμό. Η διαδικασία φαίνεται στο σχήμα που ακολουθεί.

	Πληθυσμός						Κόστος	Τουρνουά		Ενδιάμεσος πληθυσμός						Κόστος
A	2	1	3	5	4	6	31	E + F	E	3	5	4	2	1	6	22
B	5	3	1	6	2	4	20	B + E	B	5	3	1	6	2	4	20
C	4	2	1	3	6	5	28	A + D	A	2	1	3	5	4	6	31
D	1	2	5	4	6	3	34	D + D	D	1	2	5	4	6	3	34
E	3	5	4	2	1	6	22	C + B	B	5	3	1	6	2	4	20
F	5	4	1	6	3	2	25	F + C	F	5	4	1	6	3	2	25
G	6	2	5	3	4	1	18									

Εικόνα 168 Επιλογή δυαδικού τουρνουά με επανατοποθέτηση

Το πρόβλημα αυτής της μεθόδου επιλογής είναι ότι κάποια άτομα μπορεί να μη λάβουν μέρος σε κανένα τουρνουά. Στο παράδειγμά μας το άτομο G είναι το άτομο με την μεγαλύτερη καταλληλότητα αλλά δεν επιλέχθηκε γιατί δεν έλαβε μέρος σε κανένα τουρνουά.

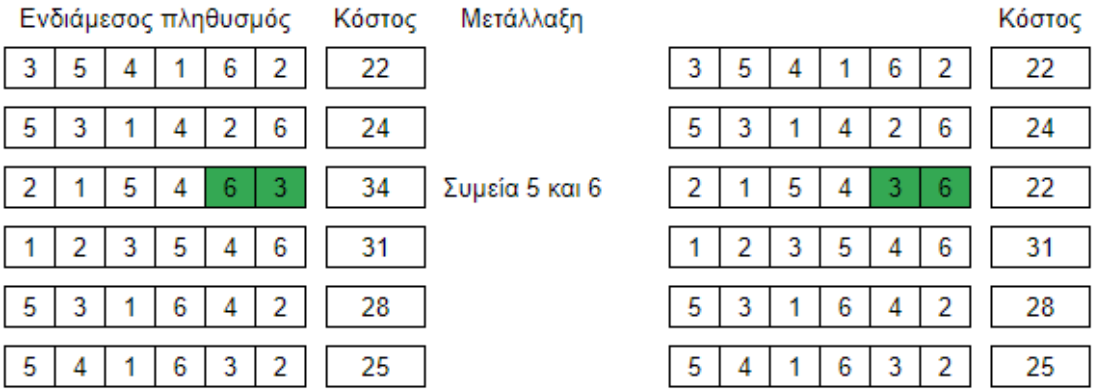
Προφανώς εφόσον έχουμε επανατοποθέτηση κάποια άτομα θα λάβουν μέρος σε περισσότερα από ένα τουρνουά και ένα άτομο μπορεί να λάβει μέρος στο ίδιο τουρνουά περισσότερες από μία φορές. Αυτό έχει ως αποτέλεσμα το άτομο με τη μικρότερη καταλληλότητα να μπορεί να επιλεγεί όπως φαίνεται και στο παράδειγμα. Το άτομο D έχει τη μικρότερη καταλληλότητα, αλλά επειδή δημιουργεί ένα δυαδικό τουρνουά με τον εαυτό του, επιλέγεται.

Μετά την επιλογή γίνεται η διασταύρωση. Η διασταύρωση γίνεται με πιθανότητα 100% επομένως όλα τα άτομα του ενδιάμεσου πληθυσμού θα προκύψουν από την διασταύρωση. Στόχος της διασταύρωσης είναι, μέσα από την ανταλλαγή γενετικού υλικού των γονέων, να προκύψουν άτομα με μεγαλύτερη καταλληλότητα, δηλαδή καλύτερες λύσεις. Επειδή ο τελεστής διασταύρωσης είναι ένας στοχαστικός τελεστής, δε μας δίνει πάντα άτομα με μεγαλύτερη καταλληλότητα. Για το πρόβλημα του πλανόδιου πωλητή υπάρχουν καλύτεροι αλγόριθμοι διασταύρωσης αλλά χρησιμοποιήθηκε ο

αλγόριθμος modified crossover επειδή είναι ο πιο απλός. Η διαδικασία της διασταύρωσης φαίνεται στο σχήμα που ακολουθεί.

Εικόνα 169 Διασταύρωση ενός σημείου για το TSP

Μετά τη διασταύρωση με πιθανότητα 10% στα άτομα που προέκυψαν από τη διασταύρωση γίνεται μετάλλαξη. Στόχος της μετάλλαξης είναι να δημιουργήσει νέες λύσεις, βοηθώντας έτσι τον γενετικό αλγόριθμο να ξεφύγει από τα τοπικά ακρότατα. Στο παράδειγμα η μετάλλαξη συμβαίνει σε ένα άτομο και το αποτέλεσμα της μετάλλαξης φαίνεται στο σχήμα που ακολουθεί.



Εικόνα 170 Μετάλλαξη με αντιστροφή για το TSP

Στον πληθυσμό που προέκυψε, επειδή έχουμε ελιτισμό, προσθέτουμε το άτομο με την μεγαλύτερη καταλληλότητα και έτσι δημιουργείται ο αρχικός πληθυσμός της νέας γενιάς. Η διαδικασία φαίνεται στο σχήμα που ακολουθεί.

Εικόνα 171 Πληθυσμός νέας γενιάς

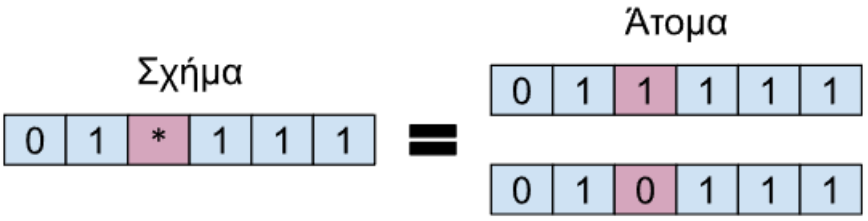
Η υλοποίηση του αλγορίθμου σε php βρίσκεται εδώ.

12 Θεωρία σχημάτων - Schema theory

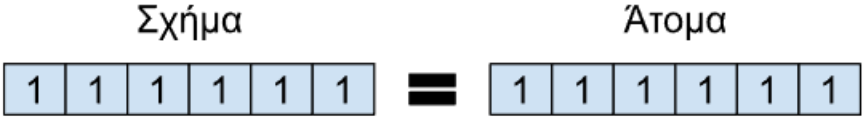
Στην προσπάθειά του να εξηγήσει γιατί δουλεύει ένας γενετικός αλγόριθμος, ο Holland διατύπωσε το θεώρημα των σχημάτων. Πριν δούμε το θεώρημα θα πρέπει να κατανοήσουμε τι είναι σχήμα και με πιο τρόπο επιδρούν οι γενετικοί τελεστές στα σχήματα.

Σύμφωνα με τον Holland σχήμα είναι ένα πρότυπο (template) που περιγράφει ένα υποσύνολο ατόμων του πληθυσμού, τα οποία έχουν κάποιους όμοιους χαρακτήρες σε συγκεκριμένες θέσεις στη συμβολοσειρά.

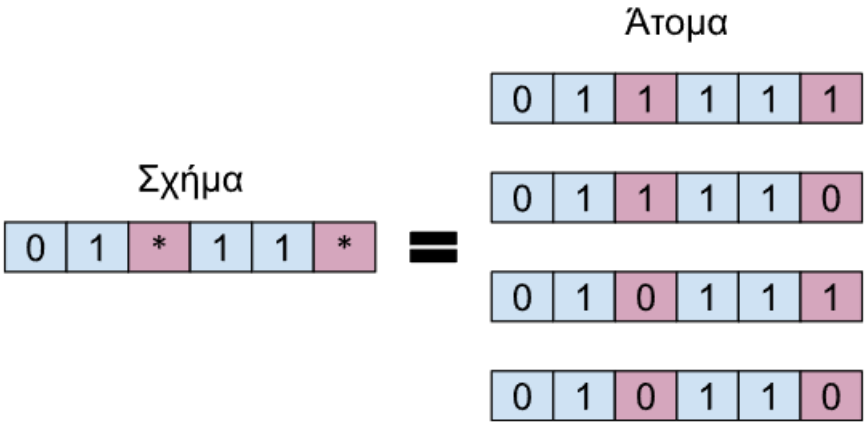
Για την κατασκευή ενός σχήματος εισάγουμε στο αλφάβητο S τον χαρακτήρα μπαλαντέρ (wildcard) *, ο οποίος μπορεί να πάρει τιμή οποιουδήποτε χαρακτήρα του αλφαβήτου S. Για παράδειγμα αν χρησιμοποιούμε το δυαδικό αλφάβητο το * μπορεί να πάρει τις τιμές 0 και 1. Στη γενική περίπτωση κατά την οποία το αλφάβητο S αποτελείται από C χαρακτήρες ο μπαλαντέρ μπορεί να πάρει C διαφορετικές τιμές. Ακολουθούν μερικά παραδείγματα σχημάτων για έναν Γ.Α.. Σημείωση: Σε όλα τα παραδείγματα για τη θεωρία σχημάτων υποθέτουμε ότι χρησιμοποιούμε τη δυαδική κωδικοποίηση.



Εικόνα 172 Άτομα που αντιπροσωπεύει το σχήμα – παράδειγμα 1



Εικόνα 173 Άτομα που αντιπροσωπεύει το σχήμα – παράδειγμα 2



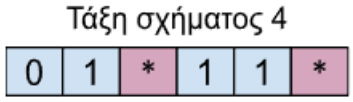
Εικόνα 174 Άτομα που αντιπροσωπεύει το σχήμα – παράδειγμα 3

Προφανώς το σχήμα ****** αντιπροσωπεύει όλες τις συμβολοσειρές μήκους έξι. Αν έχουμε ένα σχήμα με m μπαλαντέρ, το αλφάβητο αποτελείται από C χαρακτήρες και το κάθε χρωμόσωμα έχει k γονίδια τότε από την συνδυαστική ισχύουν τα εξής.

- Κάθε σχήμα αντιπροσωπεύει C^m χρωμοσώματα. Για τη δυαδική κωδικοποίηση 2^m.
- Κάθε χρωμόσωμα ταιριάζει σε C^k. Για τη δυαδική κωδικοποίηση 2^k.
- Σε κάθε αλφάβητο υπάρχουν $(C+1)^k$ σχήματα [(D.E., 1989) σελ. 19] (το +1 προκύπτει από την προσθήκη του χαρακτήρα μπαλαντέρ). Για τη δυαδική κωδικοποίηση 3^k.

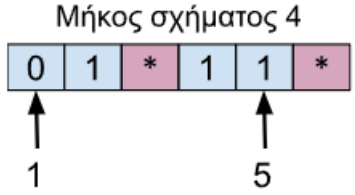
Κάθε σχήμα το χαρακτηρίζουν δύο μεγέθη, η **τάξη** και το **αντιπροσωπευτικό μήκος** του σχήματος.

Ως **τάξη (order)** ενός σχήματος Σ, η οποία συμβολίζεται με ο(Σ), ορίζεται ο αριθμός των σταθερών θέσεων του σχήματος, δηλαδή των θέσεων που δεν περιέχουν τον μπαλαντέρ (wildcard) *. Προφανώς όσο μικρότερη είναι η τάξη ενός σχήματος τόσο πιο γενικό είναι, αφού αντιστοιχεί σε περισσότερα χρωμοσώματα, ενώ αντίστροφα όσο πιο μεγάλη είναι η τάξη ενός σχήματος τόσο πιο ειδικό - συγκεκριμένο είναι. Για παράδειγμα το σχήμα (*1*1) έχει τάξη ο(*1*1)=2 και αντιστοιχεί σε τέσσερις συμβολοσειρές [(0,1,1,1) (0,1,0,1) (1,1,1,1) (1,1,0,1)] ενώ το σχήμα (0 1*1) έχει τάξη ο(0 1*1) = 3 και αντιστοιχεί σε δύο συμβολοσειρές [(0 1 1 1) (0 1 0 1)].



Εικόνα 175 Τάξη σχήματος

Ως **αντιπροσωπευτικό μήκος (defining length)** ενός σχήματος Σ, το οποίο συμβολίζεται με δ(Σ), ορίζεται η απόσταση της πρώτης και της τελευταίας σταθερής θέσης. Το μήκος του σχήματος προσδιορίζει την πυκνότητα της πληροφορίας που περιέχεται στο σχήμα. Για παράδειγμα το σχήμα (0 1 *1 1 *) έχει μήκος δ(0 1 *1 1 *)=4, το σχήμα (*1 *1 1 1) έχει επίσης μήκος δ(*1 *1 1 1) = 4, ενώ το σχήμα (* * * 1 * *) έχει μήκος δ(* * * 1 * *) = 1.



Εικόνα 176 Μήκος σχήματος

Στη συνέχεια θα δούμε πως επηρεάζουν οι γενετικοί τελεστές τον αριθμό και το είδος των σχημάτων που περιέχονται στον πληθυσμό.

12.1 Επίδραση της επιλογής

Θεωρούμε m(Σ, t) = r **(1)** τον αριθμό των χρωμοσωμάτων στη γενιά t που αντιστοιχούν στο σχήμα Σ. Ως απόδοση του σχήματος Σ τη στιγμή t ορίζεται η μέση απόδοση των χρωμοσωμάτων $\{u_1, u_2, \dots, u_r\}$, του πληθυσμού που αντιστοιχούν στο σχήμα Σ τη στιγμή t δηλαδή:

$$f(\Sigma, t) = \sum_{i=1}^{r} eval(u_i)/r \ (\mathbf{2})$$

Επίσης ορίζουμε ως μέση απόδοση $\overline{f(t)}$ του πληθυσμού τη στιγμή t τη μέση τιμή της συνάρτησης κόστους όλων των ατόμων του πληθυσμού τη στιγμή t , δηλαδή:

$$\overline{f(t)} = \sum_{i=1}^{N} eval(u_i)/N \ (\mathbf{3}), \qquad \text{με N το μέγεθος του πληθυσμού}$$

Αν η επιλογή γίνεται με τη μέθοδο της ρουλέτας η πιθανότητα να επιλεγεί η συμβολοσειρά u_i είναι $p_i = eval(u_i)/F_N$ **(4)**, όπου F_N το άθροισμα των αποδόσεων του πληθυσμού τη στιγμή t.

Από τη σχέση (2) προκύπτει ότι η πιθανότητα να επιλεγεί τουλάχιστον μία συμβολοσειρά που αντιστοιχεί στο σχήμα Σ είναι:

$$p_\Sigma = \frac{f(\Sigma, t)}{F_N} \ (\mathbf{5})$$

Από τις σχέσεις (5) και (1) προκύπτει ότι ο αριθμός των συμβολοσειρών που αντιστοιχούν στο σχήμα Σ μετά τη διαδικασία της επιλογής θα είναι:

$$m(\Sigma, t+1) = m(\Sigma, t) \cdot N \cdot \frac{f(\Sigma, t)}{F_N} \ (\mathbf{6})$$

Από τις σχέσεις (6) και (3) προκύπτει:

$$m(\Sigma, t+1) = m(\Sigma, t) \cdot \frac{f(\Sigma, t)}{\overline{f(t)}} \ (\mathbf{7})$$

Από τον τύπο (7) συμπεραίνουμε ότι ο αριθμός των ατόμων που ταιριάζει σε ένα σχήμα Σ αυξάνεται όταν η απόδοση f_Σ του σχήματος είναι μεγαλύτερη από τη μέση απόδοση του πληθυσμού $\overline{f(t)}$ και μειώνεται όταν είναι μικρότερη. Αν υποθέσουμε ότι η απόδοση του σχήματος διαφέρει από τη μέση απόδοση του πληθυσμού κατά ε%, δηλαδή $f(\Sigma, t) = \overline{f(t)} \pm \varepsilon \cdot \overline{f(t)}$ **(8)**, από τις σχέσεις (7) και (8) προκύπτουν οι σχέσεις:

$$m(\Sigma, t) = m(\Sigma, 0) \cdot (1+\varepsilon)^t \ (\mathbf{9})$$

$$m(\Sigma, t+1) = m(\Sigma, t) \cdot (1+\varepsilon) \ (\mathbf{10})$$

Στις προηγούμενες σχέσεις m(Σ,0) είναι ο αριθμός των συμβολοσειρών του αρχικού πληθυσμού που αντιστοιχούν στο σχήμα Σ. Έτσι αν ε>0 δηλαδή αν $f(\Sigma, t) > \overline{f(t)}$ ο αριθμός των ατόμων αυξάνεται γεωμετρικά ενώ αν $f(\Sigma, t) < \overline{f(t)}$ ο αριθμός μειώνεται.

Παράδειγμα: Έστω ότι έχουμε έναν Γ.Α. με δυαδική κωδικοποίηση, αριθμό γονιδίων σε κάθε χρωμόσωμα 6, συνάρτηση καταλληλότητας τον αριθμό τον άσσων, το σχήμα Σ=(1 0*1*1), μέγεθος αρχικού πληθυσμού 6 και αρχικό πληθυσμό αυτόν που φαίνεται στην παρακάτω εικόνα:

1	1	0	1	0	1
0	0	0	0	0	0
1	0	1	1	1	1
1	0	0	1	1	1
0	1	0	0	0	1
1	0	0	0	0	1

Εικόνα 177 Αρχικός πληθυσμός

Στο σχήμα Σ αντιστοιχούν τα χρωμοσώματα (1 0 0 1 1 1) και (1 0 1 1 1 1) άρα m(Σ,0)=2. Από τη σχέση (2) προκύπτει ότι η απόδοση του σχήματος Σ είναι f(Σ,0)=(4+5)/2=4.5. Από τη σχέση (3) προκύπτει ότι η μέση απόδοση του πληθυσμού είναι $\overline{f(0)} = \frac{4+5+4+2+2}{6} = 2.83$. Επειδή $f(\Sigma,0) > \overline{f(0)}$ περιμένουμε αύξηση των συμβολοσειρών που αντιστοιχούν στο σχήμα Σ. Έστω ότι μετά την επιλογή με τη μέθοδο της ρουλέτας προέκυψε ο ακόλουθος πληθυσμός.

1	1	0	1	0	1
1	0	1	1	1	1
1	0	1	1	1	1
1	0	0	1	1	1
0	1	0	0	0	1
1	0	1	1	1	1

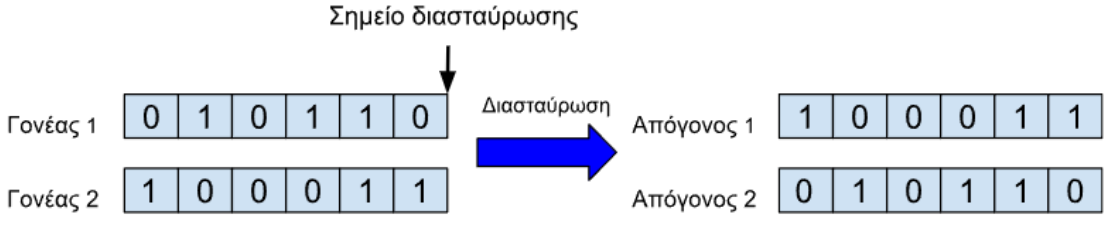
Εικόνα 178 Πληθυσμός μετά την επιλογή

Στον πληθυσμό που προέκυψε μετά την επιλογή το σχήμα Σ αντιστοιχεί σε τέσσερις συμβολοσειρές.

12.2 Επίδραση της διασταύρωσης

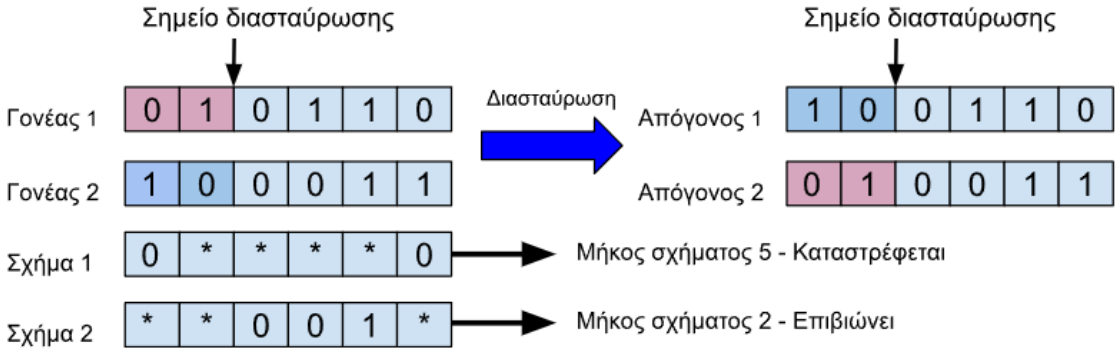
Έστω ένας γενετικός αλγόριθμος για τον οποίο χρησιμοποιήθηκε η δυαδική κωδικοποίηση, κάθε χρωμόσωμα αποτελείται από k γονίδια, η διασταύρωση έγινε με τη χρήση του αλγορίθμου διασταύρωση ενός σημείου και η πιθανότητα διασταύρωσης είναι p_c.

Για το σημείο διασταύρωσης έχουμε k-1 επιλογές αφού αν επιλέξουμε για σημείο διασταύρωσης το k οι γονείς δεν ανταλλάσσουν γενετικό υλικό. Απλά ο γονέας ένα γίνεται δύο και αντίστροφα.



Εικόνα 179 Διασταύρωση ενός σημείο στο σημείο k

Όπως είδαμε νωρίτερα το αντιπροσωπευτικό μήκος δ(Σ) μας δίνει την πληροφορία για την πυκνότητα του σχήματος. Όσο μικρότερο είναι τόσο πιο "μαζεμένες" είναι οι σταθερές θέσεις του σχήματος και τόσο μεγαλύτερη είναι η πιθανότητα να επιβιώσει το σχήμα μετά την εφαρμογή του τελεστή διασταύρωσης. Για να επιβιώσει ένα σχήμα θα πρέπει το σημείο διασταύρωσης που θα επιλέξουμε να μη βρίσκεται ανάμεσα στο πρώτο και στο τελευταίο σταθερό σημείο του σχήματος.



Εικόνα 180 Πιθανότητα επιβίωσης σχήματος ανάλογα με το μήκος του

Προφανώς η πιθανότητα καταστροφής ενός σχήματος Σ είναι $p_d(\Sigma) = \frac{\delta(\Sigma)}{k-1}$, δηλαδή από τις k-1 επιλογές που έχουμε για το σημείο διασταύρωσης, να επιλέξουμε ένα σημείο που να βρίσκεται ανάμεσα στο πρώτο και το τελευταίο σταθερό σημείο του σχήματος. Επομένως η πιθανότητα επιβίωσης ενός σχήματος Σ είναι $p_s(\Sigma) = 1 - \frac{\delta(\Sigma)}{k-1}$ $(\mathbf{11})$. Επειδή η πιθανότητα διασταύρωσης είναι p_c η σχέση (11) γίνεται $p_s(\Sigma) = 1 - p_c \cdot \frac{\delta(\Sigma)}{k-1}$ $(\mathbf{12})$.

Ακόμα και αν το σημείο διασταύρωσης είναι ανάμεσα στην πρώτη και την τελευταία σταθερή θέση ενός σχήματος, υπάρχει περίπτωση το σχήμα να επιβιώσει.

Εικόνα 181 Το σχήμα 2 επιβιώνει παρόλο που το σημείο διασταύρωσης βρίσκεται ανάμεσα στην πρώτη και την τελευταία σταθερή θέση

Επομένως η σχέση (12) γράφεται:

$$p_s(\Sigma) \geq 1 - p_c \cdot \frac{\delta(\Sigma)}{k-1} \quad (\mathbf{13})$$

Από τις σχέσεις (7) και (13) προκύπτει:

$$m(\Sigma, t+1) \geq m(\Sigma, t) \cdot \frac{f(\Sigma, t)}{f(t)} \cdot \left[1 - p_c \cdot \frac{\delta(\Sigma)}{k-1}\right] (\mathbf{14})$$

Υπενθυμίζουμε ότι m(Σ, t) είναι ο αριθμός των συμβολοσειρών που αντιστοιχούν στο σχήμα τη στιγμή t, f(Σ, t) είναι η απόδοση του σχήματος τη στιγμή t δηλαδή η μέση απόδοση των συμβολοσειρών που αντιστοιχούν στο σχήμα, $\overline{f(t)}$ είναι η μέση απόδοση όλων των συμβολοσειρών του πληθυσμού, p_c η πιθανότητα διασταύρωσης, δ(Σ) το μήκος του σχήματος και k ο αριθμός των γονιδίων.

Από τη σχέση (14) συμπεραίνουμε ότι στα σχήματα με μικρό μήκος και απόδοση μεγαλύτερη από το μέσο όρο, θα αντιστοιχούν περισσότερες συμβολοσειρές από γενιά σε γενιά.

12.3 Επίδραση της μετάλλαξης

Έστω ένας γενετικός αλγόριθμος για τον οποίο χρησιμοποιήθηκε η δυαδική κωδικοποίηση και κάθε γονίδιο έχει πιθανότητα μετάλλαξης p_m.

Για να επιβιώσει ένα σχήμα κατά τη μετάλλαξη, θα πρέπει η μετάλλαξη να μη γίνει σε κάποια από τις σταθερές θέσεις του. Η πιθανότητα να γίνει μετάλλαξη σε ένα γονίδιο είναι p_m και η πιθανότητα να μη γίνει είναι $1 - p_m$. Επειδή η πιθανότητα να γίνει μετάλλαξη σε κάποιο γονίδιο είναι ανεξάρτητη από την πιθανότητα να γίνει μετάλλαξη σε ένα άλλο γονίδιο, η πιθανότητα να επιβιώσει ένα σχήμα με τάξη o(Σ) δίνεται από τον τύπο:

$$p_s(\Sigma) = (1 - p_m)^{o(\Sigma)} \mathbf{(15)}$$

Συνήθως η πιθανότητα μετάλλαξης είναι ένας πολύ μικρός αριθμός οπότε από τη γνωστή σχέση $(1 + x)^{\alpha} \cong 1 + \alpha \cdot x$ *όταν* $|x| < 1$ *και* $|\alpha \cdot x| \ll 1$ η σχέση (15) γίνεται:

$$p_s(\Sigma) = 1 - o(\Sigma) \cdot p_m \mathbf{(16)}$$

Από τις σχέσεις (14) και (16) προκύπτει η σχέση:

$$m(\Sigma, t+1) \geq m(\Sigma, t) \cdot \frac{f(\Sigma, t)}{f(t)} \cdot \left[1 - p_c \cdot \frac{\delta(\Sigma)}{k-1} - o(\Sigma) \cdot p_m\right] \mathbf{(17)}$$

Από τη σχέση (17) συμπεραίνουμε ότι ο αριθμός των συμβολοσειρών που αντιστοιχούν σε σχήματα με απόδοση μεγαλύτερη από το μέσο όρο απόδοσης του πληθυσμού, έχουν μικρό αντιπροσωπευτικό μήκος και μικρή τάξη, αυξάνεται εκθετικά από γενιά σε γενιά. Αυτή η παρατήρηση είναι γνωστή ως θεώρημα σχημάτων (schema theorem). *"Short, low-order schemata with above-average fitness increase exponentially in frequency in successive generations."*

12.4 Αριθμός σχημάτων που επεξεργάζεται ο Γ.Α.

Είδαμε νωρίτερα ότι όταν το αλφάβητο αποτελείται από C χαρακτήρες και το κάθε χρωμόσωμα έχει μήκος k, το χρωμόσωμα θα ταιριάζει σε C^k σχήματα. Επομένως αν το μέγεθος του πληθυσμού είναι N ο αριθμός των σχημάτων Τ(Σ) που περιέχονται σε αυτόν τον πληθυσμό θα είναι $C^k \leq T(\Sigma) \leq N \cdot C^k \mathbf{(18)}$, αφού κάποια σχήματα μπορεί να αντιπροσωπεύουν περισσότερα από ένα άτομα. Για την δυαδική κωδικοποίηση η σχέση (18) γίνεται $2^k \leq T(\Sigma) \leq N \cdot 2^k \mathbf{(19)}$.

Όπως είδαμε νωρίτερα όλα τα σχήματα δεν έχουν την ίδια πιθανότητα να επιβιώσουν. Ο Holland απέδειξε ότι τελικά ο αριθμός των σχημάτων που επεξεργάζεται σε κάθε κύκλο ο γενετικός αλγόριθμος είναι N^3. (Η απόδειξη υπάρχει καλύτερα γραμμένη και στο βιβλίο του Goldberg [(D.E., 1989) σελ. 40]).

Ο Holland ονόμασε αυτή την ιδιότητα τον γενετικών αλγορίθμων έμμεσος παραλληλισμός (implicit parallelism) και έχει πολύ μεγάλη σημασία. Παρόλο που σε κάθε γενιά οι υπολογιστικές πράξεις που εκτελεί ο γενετικός αλγόριθμος είναι ανάλογες του μεγέθους του πληθυσμού N, επεξεργάζεται N^3 σχήματα. Η επεξεργασία αυτή δε φαίνεται πουθενά στον κώδικα. Δε χρειάζεται να δεσμευτεί επιπλέον μνήμη, ή να αποθηκευτούν κάποια δεδομένα σε ένα αρχείο. Η επεξεργασία των σχημάτων γίνεται με έμμεσο τρόπο μέσω των ατόμων που αντιπροσωπεύουν τα σχήματα, χωρίς να επιβαρύνεται ο γενετικός αλγόριθμος.

12.5 The building block hypothesis

Σύμφωνα με την υπόθεση των δομικών στοιχείων (Building Block Hypothesis), ένας γενετικός αλγόριθμος αναζητώντας τη βέλτιστη λύση, επιλέγει, διασταυρώνει και μεταλλάσσει, σχήματα μικρού αντιπροσωπευτικού μήκους, χαμηλής τάξης και άνω του μέσου όρου απόδοσης, στην προσπάθειά του να δημιουργήσει χρωμοσώματα με μεγαλύτερη απόδοση. Αυτά τα σχήματα ονομάζονται building blocks. Χρησιμοποιώντας τα building blocks ο γενετικός αλγόριθμος μειώνει την πολυπλοκότητα του

προβλήματος. Αντί να δοκιμάσει κάθε δυνατό συνδυασμό των λύσεων, κατασκευάζει σε κάθε κύκλο καλύτερες λύσεις, παίρνοντας το καλύτερο μέρος των λύσεων των προηγούμενων γενεών.

"Short, low order, and highly fit schemata are sampled, recombined [crossed over], and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata [the building blocks], we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings." - Goldberg 1989.

Όλα αυτά τα χρόνια έχουν γίνει πολλές προσπάθειες για την απόδειξη της υπόθεσης των δομικών στοιχείων, αλλά καμία από αυτές δεν είχε επιτυχία. Αν και έχουν αναφερθεί καλά αποτελέσματα για ορισμένες κατηγορίες προβλημάτων, εξακολουθεί να υπάρχει σκεπτικισμός σχετικά με τη γενικότητα ή/και την πρακτικότητα της υπόθεσης των δομικών στοιχείων ως εξήγηση για την αποτελεσματικότητα των γενετικών αλγορίθμων.

Βασιζόμενος στην υπόθεση των δοκιμών στοιχείων ο Goldberg δημιούργησε δύο αρχές για την επιλογή της σωστής κωδικοποίησης [(D.E., 1989) σελ. 80].

12.5.1 Αρχή των σημαντικών δομικών στοιχείων - Principle of meaningful building blocks

Η κωδικοποίηση που θα επιλεγεί πρέπει να είναι τέτοια ώστε τα σχήματα μικρής τάξης και μικρού μήκους, να είναι σχετικά με το βασικό πρόβλημα και να σχετίζονται όσο το δυνατό λιγότερο με τα σχήματα στις άλλες σταθερές θέσεις. Αν και στην πράξη η εφαρμογή αυτής της αρχής είναι δύσκολη, θα πρέπει να ελέγχουμε την απόσταση των γονιδίων που περιέχουν πληροφορίες που έχουν σχέση μεταξύ τους. Τα γονίδια αυτά θα πρέπει να βρίσκονται το ένα δίπλα στο άλλο μέσα στο χρωμόσωμα. Όταν συμβαίνει αυτό, μειώνεται η πιθανότητα να χωριστούν από τον τελεστή διασταύρωσης.

12.5.2 Αρχή του ελάχιστου αλφαβήτου - Principle of minimal alphabets

Η αρχή αυτή, η οποία στην ουσία προωθεί τη δυαδική κωδικοποίηση, αναφέρει ότι πρέπει να επιλέγουμε την κωδικοποίηση με τον μικρότερο αριθμό συμβόλων, καθώς ένα αλφάβητο με λιγότερους χαρακτήρες, περιέχει μεγαλύτερο αριθμό σχημάτων.

Έστω ότι θέλουμε να κωδικοποιήσουμε ένα διάστημα X τιμών. Ο αριθμός των ψηφίων που θα χρειαστούμε για μια τέτοια κωδικοποίηση είναι $\mathrm{k} = \lfloor \log_c X \rfloor + 1$. Είδαμε νωρίτερα ότι αν το αλφάβητο έχει C χαρακτήρες και το μήκος του χρωμοσώματος είναι k, τότε υπάρχουν $(C+1)^k$ σχήματα. Το μέγιστο της συνάρτησης $f(C) = (C+1)^k$ προκύπτει για C=2. Έτσι η δυαδική κωδικοποίηση η οποία έχει τον ελάχιστο αριθμό συμβόλων, θα έχει τον μέγιστο αριθμό σχημάτων.

Ο κύριος λόγος που εισάγαμε την έννοια των σχημάτων, ήταν για να βρούμε ομοιότητες ανάμεσα σε συμβολοσειρές με υψηλή απόδοση. Για παράδειγμα έστω ότι θέλουμε να μεγιστοποιήσουμε τη συνάρτηση $f(x) = x^2$ με το x να είναι ένας ακέραιος που παίρνει τιμές στο διάστημα [0,31]. Αν χρησιμοποιήσουμε δυαδική κωδικοποίηση τα χρωμοσώματα θα είναι της μορφής (0 0 0 0 0)(0 0 0 0 1)...(1 1 1 1 0)(1 1 1 1 1). Παρατηρούμε ότι τα χρωμοσώματα που έχουν το πρώτο γονίδιο 1, δηλαδή αυτά που αντιστοιχούν στο σχήμα (1 * * * *) έχουν μεγαλύτερη απόδοση. Αυτή την πληροφορία μπορεί να την εκμεταλλευτεί ο γενετικός αλγόριθμος. Αν αντί για την δυαδική κωδικοποίηση χρησιμοποιήσουμε τα 26 γράμματα του Αγγλικού αλφαβήτου και τους αριθμούς 1 έως 6, τότε το χρωμόσωμα θα αποτελείται από ένα γονίδιο (A) (B) … (5) (6). Τα χρωμοσώματα (5) και (6) είναι οι αριθμοί 30 και 31 αντίστοιχα και για την συνάρτηση $f(x) = x^2$ έχουν την μεγαλύτερη απόδοση, όμως δεν έχουν καμία ομοιότητα. Δεν υπάρχει κάποιο σχήμα το οποίο μπορεί να εκμεταλλευτεί ο γενετικός αλγόριθμος.

Η αρχή του ελάχιστου αλφαβήτου αμφισβητήθηκε από τον Jim Antonisse (Antonisse, 1989), η έρευνα του οποίου καταλήγει στο συμπέρασμα ότι η κωδικοποίηση με μεγάλο αριθμό χαρακτήρων δίνει καλύτερα αποτελέσματα. Αυτό που φάνηκε από μετέπειτα θεωρητικές μελέτες και πρακτικές

εφαρμογές, είναι δεν υπάρχει αναπαράσταση που να είναι ιδανική για όλα τα προβλήματα και για κάθε πρόβλημα μια αναπαράσταση θα υπερέχει.

13 Συμπεράσματα

Στην εργασία αυτή αναλύσαμε τους γενετικούς αλγορίθμους και είδαμε πως αυτοί μπορούν να χρησιμοποιηθούν για την επίλυση του προβλήματος του πλανόδιου πωλητή (ή πρόβλημα του μετακινούμενου πωλητή ή πρόβλημα του περιπλανώμενου πωλητή ή στα αγγλικά travelling salesman problem ή travelling salesperson problem ή για συντομία TSP).

Κατά τη διάρκεια της εργασίας αναλύσαμε πολλούς τρόπους επιλογής, διασταύρωσης και μετάλλαξης και είδαμε πόσο πολύ μπορεί να επηρεάσουν την ταχύτητα και την απόδοση του γενετικού αλγορίθμου.

Αν και οι γενετικοί αλγόριθμοι δεν δημιουργήθηκαν για την επίλυση τέτοιων προβλημάτων, με τη χρήση κατάλληλων τελεστών διασταύρωσης και μετάλλαξης, μπορούν να δώσουν μια καλή λύση στο πρόβλημα αρκετά γρήγορα, αρκεί να γίνει ο σωστός σχεδιασμός του γενετικού αλγορίθμου.

Το αν οι γενετικοί αλγόριθμοι είναι η καλύτερη ευρετική λύση για την επίλυση του προβλήματος του πλανόδιου πωλητή παραμένει ένα αναπάντητο ερώτημα. Τα τελευταία χρόνια δημιουργήθηκαν κάποιοι μεταευρετικοί αλγόριθμοι (όπως ο ant colony optimization) οι οποίοι έδωσαν καλύτερα αποτελέσματα (είτε σε απόδοση είτε σε χρόνο).

Παρόλα αυτά επειδή ακόμα και σήμερα δε γνωρίζουμε γιατί ακριβώς δουλεύει ένας γενετικός αλγόριθμος, υπάρχουν πολλά περιθώρια βελτίωσης και δεν αποκλείεται με έναν καλύτερο τελεστή διασταύρωσης ή και μετάλλαξης, να έχουμε πολύ καλύτερα αποτελέσματα.

14 Κώδικας αλγορίθμων

Σε αυτό το κεφάλαιο υπάρχει στο github ο κώδικας για όλους τους αλγορίθμους που χρησιμοποιήσαμε.

- Cycle crossover
- PMX Crossover
- One point crossover path representation
- One point crossover binary
- Two point crossover path representation
- Two point crossover binary
- Uniform crossover
- Shuffle crossover
- Single arithmetic crossover
- Simple arithmetic crossover
- Whole arithmetic crossover
- Reduced surrogate crossover
- SCX Version 1
- SCX Version 2
- Order crossover
- Order based crossover - OBX
- Position Crossover
- Sinusoidal Motion Crossover (SMX)
- Modified Partially-Mapped Crossover (MPMX)
- Enhanced Sequential Constructive Crossover (ESCX) Version 1 - Backup
- Enhanced Sequential Constructive Crossover (ESCX) Version 2 - Backup
- Reverse Sequence Mutation (RSM)
- Twors Mutation
- Edge recombination crossover (ERX ή ER)

- Fitness proportionate selection (RWS)
- Fitness proportionate selection (Binary)
- Fitness proportionate selection (Alias)
- Fitness proportionate selection (Stochastic)
- Linear rank selection
- Stochastic universal sampling (SUS)
- Random tournament selection with replacement
- Random tournament selection without replacement
- Unbiased tournament selection
- Parallel unbiased tournament selection
- Centre inverse mutation (CIM) με array reverse
- Centre inverse mutation (CIM) με for loops
- Partial shuffle mutation (PSM) - Scramble mutation
- Thoros mutation
- Thoras mutation
- Μετάλλαξη με μετατόπιση - Displacement mutation (DM)
- Μετάλλαξη με εισαγωγή - Insertion Mutation - Position Based Mutation Operator
- Cut-inverse mutation operator - Inverted Displacement mutation (IDM)
- Truncation selection
- Bidirectional Circular Sequential Constructive Crossover
- Greedy Subtour Crossover - GSX-0
- Greedy Subtour Crossover - GSX-1
- Greedy Subtour Crossover - GSX-1 from parent 2
- Greedy Subtour Crossover - GSX-2 from parent 2
- Alternating-position crossover (AP)
- Boundary mutation
- Uniform mutation
- Non-uniform mutation
- Linear crossover
- Discrete crossover
- Heuristic crossover
- Bit flip mutation
- Gaussian mutation
- Parent-centric BLX-α (PBX-α)
- Blend alpha crossover (BLX-α)
- Simulated binary crossover (SBX)
- Γενετικός αλγόριθμος
- Knapsack problem

15 Παράρτημα

15.1 Knapsack problem

Το πρόβλημα του σακιδίου είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης και διατυπώνεται ως εξής:

Έχουμε N αντικείμενα και το κάθε ένα από αυτά έχει βάρος w_i και αξία v_i με $i \in [1, N]$. Έχουμε ένα σακίδιο στο οποίο μπορούμε να βάλουμε αντικείμενα συνολικού βάρους W. Ζητάμε ένα σύνολο από τα παραπάνω αντικείμενα που το συνολικό του βάρος να είναι μικρότερο από W ενώ ταυτόχρονα η συνολική του αξία να γίνει όσο το δυνατόν μεγαλύτερη.

Υπάρχουν πολλές παραλλαγές του Knapsack problem. Η πιο διαδεδομένη είναι το 0-1 knapsack problem. Σε αυτή την παραλλαγή τα αντικείμενα μπορούν να πάρουν δύο τιμές 0 ή 1. Δεν μπορούμε να πάρουμε αντίγραφα από το ίδιο αντικείμενο και δεν μπορούμε να σπάσουμε το αντικείμενο σε κομμάτια.

Υπάρχουν πολλοί αλγόριθμοι επίλυσης του knapsack problem. Κάποιοι από αυτούς δίνουν την ακριβή λύση (με δυναμικό προγραμματισμό, με τον αλγόριθμο branch and bound είτε με συνδυασμό αυτών των δύο) ενώ άλλοι όπως οι γενετικοί αλγόριθμοι, μια προσεγγιστική λύση.

Στο σύνδεσμο που ακολουθεί, δίνεται η λύση του 0-1 knapsack σε php με δυναμικό προγραμματισμό.

15.2 Δυαδικοί αριθμοί και bitwise operators

Όταν κάνουμε αναπαράσταση ενός δεκαδικού αριθμού σε δυαδικό, ο αριθμός των bit που θα χρειαστούμε εξαρτάται από την ακρίβεια που θέλουμε να έχουμε. Αν ε η ακρίβεια τότε ο αριθμός των bit δίνεται από τον τύπο:

$$n = \left\lfloor \log_2\left(\frac{X_U - X_L}{\varepsilon}\right) \right\rfloor + 1, \mu\varepsilon\ \varepsilon \in (0,1)$$

Πολλές φορές η ακρίβεια ζητείται ως ο αριθμός κάποιον δεκαδικών ψηφίων. Για παράδειγμα αν ζητείται ακρίβεια 2 δεκαδικών ψηφίων ε=0.01. Αν γνωρίζουμε τον αριθμό των bit και θέλουμε να βρούμε την ακρίβεια χρησιμοποιούμε τον τύπο:

$$\varepsilon = \frac{X_U - X_L}{2^n}$$

Ένας εναλλακτικός τρόπος για να υπολογίσουμε τα bit είναι η χρήση των λεγόμενων τελεστών επιπέδου bit (bitwise operators). Υπάρχουν έξι bitwise operators.

~ NOT. Αντιστρέφει τα bit . ~a Αντιστρέφει τα bit της μεταβλητής a δηλαδή τα 0 γίνονται 1 και τα 1 γίνονται 0.

& AND. Σύζευξη σε επίπεδο bit. Ένα bit είναι 1, αν τα αντίστοιχα bits των μεταβλητών είναι και τα δύο 1. a & b = 0 αν a = 2 και b = 5 γιατί a = 010 και b = 101.

| OR. Διάζευξη σε επίπεδο bit. Ένα bit είναι 1, αν τουλάχιστον ένα από τα αντίστοιχα bits των μεταβλητών είναι 1. a | b = 7 αν a = 2 και b = 5 γιατί a = 010 και b = 101 οπότε a | b = 111.

^ XOR. Αποκλειστική διάζευξη σε επίπεδο bit. Ένα bit είναι 1, αν ακριβώς ένα από τα αντίστοιχα bits των μεταβλητών είναι 1. a ^ b = 7 αν a = 2 και b = 5 γιατί a = 010 και b = 101 οπότε a ^ b = 111.

<< Left shift. Ολίσθηση προς τα αριστερά. Μετατοπίζει τα bit του αριστερού τελεστέου τόσες θέσεις αριστερά όσες μας λέει ο δεξιός τελεστέος. Τα κενά bit που προκύπτουν δεξιά συμπληρώνονται με 0 ενώ τα αριστερά που βγαίνουν έξω από το όριο χάνονται. Για παράδειγμα αν έχουμε τον αριθμό 9 δηλαδή a = 1 0 0 1 και αποθηκεύουμε μέχρι 4 bit, αν κάνουμε a << 2 θα πάρουμε ως αποτέλεσμα 0 1 0 0 . Ο πρώτος άσσος χάθηκε, ο δεύτερος μετατοπίστηκε 2 θέσεις αριστερά και συμπληρώνουμε με μηδενικά.

>> Right shift. Ολίσθηση προς τα δεξιά. Μετατοπίζει τα bit του αριστερού τελεστέου τόσες θέσεις δεξιά όσες μας λέει ο δεξιός τελεστέος. Τα κενά bit που προκύπτουν αριστερά συμπληρώνονται με 0 ενώ τα δεξιά που βγαίνουν έξω από το όριο χάνονται. Για παράδειγμα αν έχουμε a = 9 = 1 0 0 1 και κάνουμε a >> 2 θα πάρουμε ως αποτέλεσμα 0 0 1 0 δηλαδή 2.

Ένας εύκολος τρόπος για να μετρήσουμε τα bit που χρειαζόμαστε είναι να κάνουμε δεξιά ολίσθηση ένα bit τη φορά μέχρι ο αριθμός να γίνει 0. Οι φορές που κάναμε ολίσθηση μέχρι να γίνει ο αριθμός 0 είναι και τα bit που χρειαζόμαστε. Η υλοποίηση σε php βρίσκεται εδώ.

Για να μετατρέψουμε έναν δυαδικό αριθμό $\Delta(\alpha_1, \alpha_2, \dots, \alpha_L)$ που αναπαριστά έναν δεκαδικό x ο οποίος παίρνει τιμές σε ένα διάστημα [A,B] χρησιμοποιούμε τον τύπο $x = A + C \cdot \frac{D}{Z}$ όπου:

- C ο δυαδικός αριθμός αν ήταν ακέραιος δηλαδή $C = \sum_{j=0}^{L-1} \alpha_{L-j} \cdot 2^j$.
- D το μήκος του διαστήματος δηλαδή D = B – A.

- Z ο μέγιστος αριθμός που μπορούμε να πάρουμε με τα bit που χρησιμοποιούμε δηλαδή $Z = 2^L - 1$.

Παράδειγμα 1: Αναπαράσταση του αριθμού 4.2 στο διάστημα [-3,4.5] με ακρίβεια δύο δεκαδικών ψηφίων. Η αναπαράσταση της μεταβλητής $-3 \leq x \leq 4.5$ με ακρίβεια δύο δεκαδικών ψηφίων χρειάζεται $n = \left\lfloor \log_2(\frac{4.5-(-3)}{0.01}) \right\rfloor + 1 = 10bits$. Ο αριθμός 4.2 γράφεται σε δυαδική μορφή:

$$4.2_2 = (\left\lfloor (4.2 - (-3)) \cdot \frac{2^{10} - 1}{4.5 - (-3)} \right\rfloor)_2 = (\left\lfloor 7.2 \cdot \frac{1023}{7.5} \right\rfloor)_2 = 982_2 = 1111010110$$

Παράδειγμα 2: Θέλουμε να κωδικοποιήσουμε τη μεταβλητή $x \in [1,10]$. Αν x=3.14 και έχουμε διαθέσιμα 8 bit τότε $x_2 = (\lfloor (3.14 - 1) \cdot 255/9 \rfloor)_2$. Ο δυαδικός αριθμός 00111100 μας δίνει τον αριθμό $x = 1 + (9/255) \cdot 60 = 3.12$. Αν θέλουμε μεγαλύτερη ακρίβεια πρέπει να χρησιμοποιήσουμε περισσότερα bit.

16 Βιβλιογραφία

A. dos Santos-Paulino, J.-C. N.-R. (2014). *Evolutionary algorithm for dense pixel matching in presence of distortions.*

A. E. Eiben Vrije, P.-E. R. (1994). *Genetic algorithms with multi-parent recombination.*

A. Homaifar, S. G. (1993). A new approach to the traveling salesman problem by genetic algorithms. Στο *Proc. 5th Int. Conf on Genetic Algorithms (1CGA '93), University of Illinois at Urbana-Champaign, Champaign, IL* (σσ. 460-466).

Abdullah Konak, D. W. (2006). *Multi-objective optimization using genetic algorithms: A tutoria.*

Adam Lipowski, D. L. (2011). *Roulette-wheel selection via stochastic acceptance.*

Adarsh Sehgal, H. M. (2019). *Deep Reinforcement Learning Using Genetic Algorithm for Parameter Optimization.*

Ahmad B. A. Hassanat, E. A. (2017). *On Enhancing Genetic Algorithms Using New Crossovers.*

Ahmad B. Hassanat, V. B.-Q. (2018). *An Improved Genetic Algorithm with a New Initialization Mechanism Based on Regression Techniques.*

Ahmad Hassanat, K. A. (2019). *Choosing Mutation and Crossover Ratios for Genetic Algorithms-A Review with a New Dynamic Approach.*

Ahmed, Z. H. (2010). *Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator.*

Ahmed, Z. H. (2011). *An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem.*

Alanzi, H. B. (2017). *Genetic Algorithm For The Travelling Salesman Problem using Enhanced Sequential Constructive Crossover Operator.*

Ali Yadav Nikravesh, S. A.-H. (2018). *Using genetic algorithms to find optimal solution in a search space for a cloud predictive cost-driven decision maker.*

Anca Andreica, C. C. (2014). *Best-order crossover for permutation-based evolutionary algorithms.*

Antonisse, J. (1989). *A new interpretation of schema notation that overturns the binary encoding constraint.*

Anupriya Shukla, H. M. (2015). *Comparative Review of Selection Techniques in Genetic Algorithm.*

B.R.Rajakumara, A. G. (2013). *APOGA: An Adaptive Population Pool Size Based Genetic Algorithm.*

Bagley, J. D. (1967). *The behavior of adaptive systems which employ genetic and correlation algorithms : technical report.*

Baker, J. E. (1985). *Adaptive selection methods for genetic algorithms.*

Baker, J. E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. Στο *Proceedings of the Second International Conference on Genetic Algorithms and Their Application. Hillsdale, New Jersey: L. Erlbaum Associates* (σσ. 14-21).

Banzhaf, W. (1990). *The "molecular" traveling salesman.*

Barricelli, N. A. (1954). *Esempi numerici di processi di evoluzione.*

Barricelli, N. A. (1957). *Symbiogenetic evolution processes realized by artificial methods.*

Booker, L. (1987). Improving Search in Genetic Algorithms. Στο *Algorithms and Simulated Annealing* (σσ. 61-73).

Burkowski, F. (1999). *Shuffle crossover and mutual information.*

Buurman, J., Zhang, S., & Babovic, V. (2009). *Reducing risk through real options in systems design: the case of architecting a maritime domain protection system.*

C. Janikow, Z. M. (1991). *An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms.*

C.García-Martínez, M. F. (2008). *Global and local real-coded genetic algorithms based on parent-centric crossover operators.*

Chang, W.-D. (2006). *Coefficient estimation of IIR filter by a multiple crossover genetic algorithm.*

Chen, M. L. (2008). *A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization.*

Chuangyin Dang, M. L. (2007). *A floating-point genetic algorithm for solving the unit commitment problem.*

Crosby, J. L. (1973). *Computer Simulation in Genetics.*

D. Dumitrescu, B. L. (2000). *Evolutionary Computation.*

D.D.(2015, P. V. (2015). *THE OPTIMAL CROSSOVER OR MUTATION RATES IN GENETIC ALGORITHM: A REVIEW.*

D.E., G. (1989). *GENETIC ALGORITHMS in Search, Optimization and Machine Learning, Addison Wesley Publishing Company, Inc.*

Darrell Whitley, J. K. (1988). *GENITOR: A different genetic algorithm.*

Dassanayake, P. (2015). *Effect of Mutation and Effective Use of Mutation in Genetic Algorithm.*

Davis, L. (1985). *Applying adaptive algorithms to epistatic domains.*

Davis, L. (1985). *Applying Adaptive Algorithms toEpistatic Domains.*

Davis, L. (1991). *Handbook of Genetic Algorithms.*

Donald E. Brown, C. L. (1989). A Parallel Genetic Heuristic for the Quadratic Assignment Problem. Στο *Proceedings of the 3rd International Conference on Genetic Algorithms* (σσ. 406-415).

Dr. Anantkumar J. Umbarkar, P. D. (2015). *Crossover Operators In Genetic Algorithms: A Review.*

Eiben, A. S. (2003). *Introduction to Evolutionary Computing.*

Eneko Osaba. Roberto Carballedo, F. D. (2014). *On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: A first study on the TSP.*

Erna Budhiarti Nab, O. S. (2018). *Genetic Algorithms Dynamic Population Size with Cloning in Solving Traveling Salesman Problem.*

Erna Budhiarti Nab, O. S. (2018). *Genetic Algorithms Dynamic Population Size with Cloning in Solving Traveling Salesman Problem.*

Eshelman, L. a. (1993). *Real-Coded Genetic Algorithms and Interval Schemata.*

Eshelman, L. C. (1989). *Biases in the Crossover Landscape.*

F. Herrera, E. H.-v. (1994). *Fuzzy Tools to Improve Genetic Algorithms.*

F. Herrera, M. L. (1998). *Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis.*

Fisher, B. J. (2013). *Optimization of Water-level Monitoring Networks in the Eastern Snake River Plain Aquifer Using a Kriging-based Genetic Algorithm Method.*

Fogel L. J., O. A. (1966). *Artificial Intelligence through Simulated Evolution.*

Fogel, D. (2002). *n memoriam Alex S. Fraser [1923-2002].*

Fogel, D. B. (1998). *Evolutionary Computation: The Fossil Record.*

Forrest, S. (1985). *Documentation for prisoner's dilemma and norms programs that use the genetic algorithm." Unpublished report.*

Fraser, A., & Burnell, D. (1970). *Computer Models in Genetics.*

Frasher, A. (1957). *Simulation of Genetic Systems by Automatic Digital Computers I. Introduction.*

García-Hernández, L., Araúzo-Azofra, A., & Salas-Morera, L. (2009). *A REVIEW ON ENCODING SCHEMES USED BY GENETIC ALGORITHMS IN PLANT LAYOUT DESIGN .*

Genetic Algorithms for Real Parameter Optimization. (1990). *Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, G.J.E. Rawlin(Ed.)*, (σσ. 205-218).

George B. Dantzig, D. R. (1954). *Solution of a Large-Scale Traveling-Salesman Problem.*

Goldberg, D. D. (1991). A comparative analysis of selection schemes used in genetic algorithms. Στο *Foundations of Genetic Algorithms,* (σσ. 69-93).

Goldberg, D. E. (1989). Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. Στο *Complex Systems, 3* (σσ. 129- 152).

Goldberg, D. E. (1989). Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. Στο *Complex Systems 3* (σσ. 153-171).

Goldberg, D. E. (1990). *A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing.*

Goldberg, D. E. (1991). *Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking.*

Grefenstette, J. (1986). *Optimization of Control Parameters for Genetic Algorithms.*

H. Mühlenbein, D. S.-V. (1993). *Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization.*

H. Sengoku, I. Y. (1999). *A Fast TSP Solver Using GA on JAVA.*

H. Voigt, H. M. (1995). *Fuzzy Recombination for the Breeder Genetic Algorithm.*

H.D. Nguyen, I. Y. (2000). *Modified edge recombination operators of genetic algorithms for the traveling salesman problem.*

Hassan Ismkhan, K. Z. (2013). *Study of Some Recent Crossovers Effects on Speed and Accuracy of Genetic Algorithm, Using Symmetric Travelling Salesman Problem.*

Hingee, K. H. (2008). Equivalence of probabilistic tournament and polynomial ranking selection. Στο *In: Proceedings of IEEE Congress on Evolutionary Computation* (σσ. 564-571).

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.*

Hommadi, A. (2018). *Multi-Stop Routing Optimization: A Genetic Algorithm Approach.*

Hsien-Pin Hsu, T.-L. C.-N.-P.-C. (2019). *A Hybrid GA with Variable Quay Crane Assignment for Solving Berth Allocation Problem and Quay Crane Assignment Problem Simultaneously.*

Hung Dinh Nguyen, Y. I. (2003). *Greedy Genetic Algorithms for Symmetric and Asymmetric TSPs.*

J. Grefenstette, R. G. (1985). enetic algorithms for the traveling salesman problem. Στο *Proc. 1st. Int. Conf. on Genetic Algorithms (ICGA '85), Carnegie-Mellon University, Pittsburgh* (σσ. 160-168).

Janikow, U. K. (χ.χ.). *An Analysis of Gray versus Binary Encoding in Genetic Search.*

Jason Digalakis, K. G. (2001). *On benchmarking functions for genetic algorithm.*

Jong, K. A. (1975). *Analysis of the behavior of a class of genetic adaptive systems.*

Jun LI, H. T. (2001). *A Real-coded Genetic Algorithm Applied to Optimum Design of a Low Solidity Vaned Diffuser for Diffuser Pump.*

K. Deb, R. B. (1995). *Simulated Binary Crossover for Continuous Search Space.*

K. Deep, M. T. (2007). *A new crossover operator for real coded genetic algorithms.*

Kevin L. Mills, J. J. (2014). *Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters.*

Kumar, A. (2013). *ENCODING SCHEMES IN GENETIC ALGORITHM.*

Kusum Deep, H. M. (2011). *New Variations of Order Crossover for Travelling Salesman Problem.*

L. Budin, M. G. (2000). *Traditional Techniques of Genetic Algorithms Applied to Floating-Point Chromosome Representations.*

Larranaga, P. K. (1994). *Optimal decomposition of Bayesian networks by genetic algorithms.*

Lingle, D. G. (1985). *Alleles, loci, and the Traveling Salesman Problem.*

Lobo, F. M. (2000). *THE PARAMETER-LESS GENETIC ALGORITHM: RATIONAL AND AUTOMATED PARAMETER SELECTION FOR SIMPLIFIED GENETIC ALGORITHM OPERATION.*

Luke, S. (2016). *Essentials of Metaheuristics.*

Mandira Chakraborty, U. K. (1999). *Branching Process Analysis of Linear Ranking and Binary Tournament Selection in Genetic Algorithms.*

Manger, K. P. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. Στο *MATHEMATICAL COMMUNICATIONS Math. Commun. 18* (σσ. 359–375).

Manuel Lozano, F. H. (2004). *Real-Coded Memetic Algorithms with Crossover Hill-Climbing.*

Masato Takahashi, H. K. (2001). *A Crossover Operator Using Independent Component Analysis for Real-Coded Genetic Algorithms.*

McMahon, B. F. (1991). Genetic operators for sequencing problems. Στο J. Rawlins, *Foundations of Genetic Algorithms* (σσ. 284-300).

McMahon, B. F. (1991). Genetic operators for sequencing problems. Στο J. Rawlins, *Foundations of Genetic Algorithms* (σσ. 284-300).

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.*

Min Dong, Y. W. (2009). Dynamic Crossover and Mutation Genetic Algorithm Based on Expansion Sampling. Στο *Proceedings of International Conference on Artificial Intelligence and Computational Intelligence* (σσ. 141-149).

Montana, D. J. (1995). *Neural Network Weight Selection Using Genetic Algorithms.*

Mühlenbein, D. S.-V. (1994). *Strategy Adaptation by Competition.*

Nicolás García-Pedrajas, C. H.-B. (2005). *CIXL2: A Crossover Operator for Evolutionary Algorithms Based on Population Features.*

Nomura, T. (1997). *An Analysis on Crossovers for Real Number Chromosomes in an Infinite Population Size.*

Olga Yugay, I. K. (2008). *Hybrid Genetic Algorithm for Solving Traveling Salesman Problem with Sorted Population.*

Oliver, I. S. (1987). Study of permutation crossover operators on the traveling salesman problem, In: Grefenstette, J. J. (ed.) Genetic Algorithms and Their Applications, Proceedings of the Second International Conference. Hillsdale, New Jersey: Lawrence Erlbaum.

Olympia Roeva, S. F. (2013). *Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling.*

Ono I, K. S. (1997). *A real-coded genetic algorithm for functional optimization using unimodal normal distribution crossover.*

P. Larrañaga, C. K. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Στο *Artificial Intelligence Review* (σσ. 129-170).

Parashar, I. G. (2011). Study of Crossover operators in Genetic Algorithm for Travelling Salesman Problem. Στο *International Journal of Advanced Research in Computer Science.*

Pedro A. Diaz-Gomez, D. F. (2007). *Initial Population for Genetic Algorithms: A Metric Approach.*

Poli, R. (2005). *Tournament Selection, Iterated Coupon-Collection Problem, and Backward-Chaining Evolutionary Algorithms.*

Radcliffe, N. J. (1991). *Equivalence Class Analysis of Genetic Algorithms.*

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.*

Reeves, C. (1993). Using Genetic Algorithms with Small Populations. *Proc. of the Fifth Int. Conf. on Genetic Algorithms, S. Forrest (Ed.)*, (σσ. 92–99).

Risi, S., & Stanley, K. O. (2012). *An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density and Connectivity of Neurons.*

S. Gotshall, B. R. (2002). *Optimal Population Size and the Genetic Algorithm.*

S. Kang, S.-S. K.-H.-M. (χ.χ.). *Bidirectional constructive crossover for evolutionary approach to travelling salesman problem,.*

S.N.Deepa, S. ·. (2008). *Introduction to Genetic Algorithms.*

Sandi Baressi Šegota, N. A. (2020). *Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms.*

Schaffer J.D., C. R. (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. Στο *Proceeding of the 3rd International Conference on Genetic Algorithms and their applications,.*

Schaffer, J. C. (1989). *A study of control parameters affecting online performance of genetic algorithms for function optimization.*

Schwarz, K. (2011). *keithschwarz*. Ανάκτηση από Darts, Dice, and Coins: Sampling from a Discrete Distribution: https://www.keithschwarz.com/darts-dice-coins/

Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik.*

Sefiane Slimane, B. M. (2012). *Portfolio Selection Using Genetic Algorithm.*

Sefiane, S. a. (2012). *Portfolio Selection Using Genetic Algorithm.*

Shu-heng Chen, C.-H. Y. (1996). *Genetic programming learning and the cobweb model.*

Siew Mooi Lim, A. B. (2017). *Crossover and Mutation Operators of Genetic Algorithms.*

Sokolov, A. W. (2005). Unbiased tournament selection. Στο *Proceedings of Genetic and Evolutionary Computation Conference* (σσ. 1131–1138).

Stanislawska, K., Krawiec, K., & Kundzewicz, Z. W. (2012). *Modelling global temperature changes with genetic programming.*

Stjepan Picek, D. J. (2013). *On the Recombination Operator in the Real-Coded Genetic Algorithms.*

Stjepan Picek, M. G. (2020). *On the efficiency of crossover operators in genetic algorithms with binary representation.*

Sumati Jaggi, R. B. (2013). *Guidelines to Decide the Encoding Scheme Used For G.A.*

Syswerda, G. (1989). *Uniform Crossover in Genetic Algorithms.*

Syswerda, G. (1991). Schedule Optimization Using Genetic Algorithms. Στο L. Davis, *Handbook of Genetic Algorithms* (σσ. 332–349).

T., M. (2002). Calculating the expected loss of diversity of selection schemes. Στο *Evolutionary Computation, 10(4)* (σσ. 397-422).

Takeda, A. Y. (1999). Optimization of Delivery Route in a City Area using Genetic Algorithm,. Στο *Proc. 4th Int'l Symposium on Artificial Life, and Robotics* (σσ. 496-499).

Tanese, R. (1989). Distributed genetic algorithm. Στο *3rd Int. Conf. on Genetic Algorithms* (σσ. 434-440).

Ting, C.-K. (2005). *On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection.*

Tobias Blickle, L. T. (1995). *A Comparison of Selection Schemes used in Genetic Algorithms.*

Varun Kumar S G1, D. R. (2017). *A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator.*

Wall, M. B. (1996). *A Genetic Algorithm for Resource-Constrained Scheduling.*

Whitley, D., Starkweather, T., & Fuquay, D. (1989). Scheduling problems and traveling salesman: The genetic edge recombination operator. Στο *International Conference on Genetic Algorithms* (σσ. 133-140).

Wright, A. H. (1999). *Genetic Algorithms for Real Parameter Optimization.*

Xie, H. Z. (2008). An analysis of multi-sampled issue and no-replacement tournament selection. Στο *Proceedings of Genetic and Evolutionary Computation Conference* (σσ. 1323–1330).

Xie, H. Z. (2008). Is the not-sampled issue in tournament selection critical? Στο *Proceedings of IEEE Congress on Evolutionary Computation,* (σσ. 3711–3718).

Xin Yao, Y. L. (1999). *Evolutionary Programming Made Faster.*

Yang, R. (1998). *Solving large travelling salesman problems with small populations.*

Yongsheng Fang, J. L. (2010). *A Review of Tournament Selection in Genetic Programming.*