ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναζήτηση σε δίκτυα ομότιμων κόμβων με χρήση γενετικού προγραμματισμού

Ιωάννης Χ. Αλαγιάννης

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2008

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναζήτηση σε δίκτυα ομότιμων κόμβων με χρήση γενετικού προγραμματισμού

Ιωάννης Χ. Αλαγιάννης

Α.Μ.: Μ799

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:
Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ
Μανόλης Κουμπαράκης, Αναπληρωτής Καθηγητής ΕΚΠΑ

Απρίλιος 2008

ΠΕΡΙΛΗΨΗ

Ο γενετικός προγραμματισμός είναι ένας εξελικτικός αλγόριθμος ο οποίος χρησιμοποιεί τις αρχές της φυσικής εξέλιξης για τη δημιουργία προγραμμάτων που μπορούν να επιλύουν ένα δοθέν πρόβλημα. Στα πλαίσια αυτής της εργασίας, ένας τέτοιος αλγόριθμος ενσωματώνεται στο μηχανισμό αναζήτησης δικτύων ομότιμων κόμβων (Peer to Peer - P2P). Πιο συγκεκριμένα, με τη βοήθεια ενός γενετικού προγράμματος τροποποιείται ο αλγόριθμος αναζήτησης στα πρωτόκολλα, Chord και S-Chord, ώστε να μειωθεί το μονοπάτι αναζήτησης για την εύρεση κάποιου δεδομένου.

Η αναπαράσταση των γενετικών προγραμμάτων πραγματοποιείται με τη βοήθεια μιας γενετικής γλώσσα που αναπτύχθηκε για τις ανάγκες της εργασίας, ενώ για την αποτίμηση των προγραμμάτων αναπτύχθηκε ένας διερμηνευτής για την εκτέλεση της γενετικής γλώσσας. Ακόμα, μελετήθηκαν και υλοποιήθηκαν διάφοροι αλγόριθμοι για την αρχικοποίηση και εξέλιξη γενετικών πληθυσμών καθώς και για την τροποποίηση γενετικών προγραμμάτων. Επιπλέον, αναπτύχθηκε η υποδομή για την προσομοίωση των πρωτοκόλλων Chord και S-Chord, με και χωρίς το γενετικό μηχανισμό ώστε να είναι δυνατή η σύγκριση ανάμεσα στις δύο προσεγγίσεις.

Η προσθήκη του γενετικού μηχανισμού οδήγησε σε βελτίωση στην αναζήτηση στο δίκτυο, χωρίς να χρειαστεί να μεταβάλλει τους αλγόριθμους των πρωτοκόλλων. Τέλος, παρουσιάζονται μια σειρά πειραμάτων που αφορούν τόσο την απόδοση στον υπολογισμό των γενετικών προγραμμάτων όσο και το πως επηρεάζει η προσθήκη του γενετικού μηχανισμού την απόδοση στα δύο πρωτόκολλα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γενετικός Προγραμματισμός, Κατανεμημένα Συστήματα.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Γενετικές πράξεις, εξελικτικοί υπολογισμοί, υπολογισμός καταλληλότητας, δίκτυα ομότιμων κόμβων, πρωτόκολλα κατανεμημένων πινάκων κατακερματισμού

ABSTRACT

Genetic programming (GP) is an evolutionary algorithm based on Darwinian natural selection, applied to the problem of creating a program that enables a computer to solve a specified problem. This thesis discusses the integration of a genetic mechanism in the search process of a peer-to-peer system. Moreover, search algorithms in Chord and S-Chord are modified, in order to improve the lookup efficiency in these protocols.

Representation of genetic programs is a key issue, so a LISP-like genetic language capable of expressing general and hierarchical genetic programs was introduced. An interpreter was developed to execute genetic programs and evaluate genetic expressions. Furthermore, different algorithms for initializing tree structures in GP populations, different selection methods for fitter individuals and different genetic operations were studied and implemented. Apart from that, the appropriate infrastructure for simulating Chord and S-Chord network protocols was developed, in order to compare the performance of search algorithms using the genetic mechanism against classic algorithms.

The employ of genetic mechanism improved the lookup efficiency, without interfering with the basic characteristics of DHT protocols. Finally, simulation results from computing genetic programs and the performance of the aforementioned protocols using the genetic mechanism are presented.

SUBJECT AREA: Genetic Programming, Distributed Systems.

KEYWORDS: Genetic operations, evolutionary computation, fitness computation, peer-to-peer networks, DHT (Distributed Hash Tables) protocols

ΕΥΧΑΡΙΣΤΙΕΣ

Πριν ξεκινήσει η παρουσίαση της εργασίας θα ήταν παράλειψη να μην ευχαριστήσω όλους όσους βοήθησαν και στήριξαν την εκπόνηση αυτής της διπλωματικής εργασίας.

Συγκεκριμένα, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ.Χατζηευθυμιάδη για τη συνεργασία, την καθοδήγηση, τις χρήσιμες συμβουλές και τη στήριξη κατά τη διάρκεια της εκπόνησης αυτής της εργασίας. Ακόμα, θέλω να ευχαριστήσω όλα τα παιδιά από την ερευνητική ομάδα διάχυτου υπολογισμού (P-comp), όπου εδώ και αρκετό καιρό μοιραζόμαστε το ίδιο γραφείο, για το θερμό και παραγωγικό κλίμα ανταλλαγής απόψεων και συνεργασίας.

Τέλος, ακόμα μια φορά θέλω να ευχαριστήσω την οικογένεια μου που με στήριξε και με ανέχτηκε σε όλη τη διάρκεια της προσπάθειας για την ολοκλήρωση αυτής της εργασίας.

Απρίλης 2008

Ιωάννης Αλαγιάννης

ΠΕΡΙΕΧΟΜΕΝΑ

Μέρος Γ΄: Εφαρμογή Γενετικού Προγραμματισμού στα Πρωτόκολλα Chord και S-Chord

ΠΡΟΛΟΓΟΣ

Η διπλωματική αυτή εργασία εκπονήθηκε στα πλαίσια του Μεταπτυχιακού Διπλώματος Ειδίκευσης (ΜΔΕ) με ειδίκευση στην Τεχνολογία Συστημάτων Υπολογιστών του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.

Το παρόν κείμενο έχει οργανωθεί σε τρία μέρη. Στο Μέρος Α πραγματοποιείται μια ανασκόπηση στο πεδίο του γενετικού προγραμματισμού, δίνοντας έμφαση στα στοιχεία που ενσωματώνονται στο διερμηνευτή. Στο Μέρος Β παρουσιάζονται, σύντομα, τα πρωτόκολλα κατανεμημένων πινάκων κατακερματισμού δίνοντας έμφαση στους αλγορίθμους αναζήτησης που θα τροποποιήσουμε. Τέλος, στο Μέρος Γ παρουσιάζεται η γενετική γλώσσα, ο διερμηνευτής που υλοποιήθηκε για τη γλώσσα αυτή, η προσθήκη του γενετικού μηχανισμού στους αλγορίθμους αναζήτησης των πρωτοκόλλων Chord και S-Chord καθώς και τα αποτελέσματα από την προσθήκη αυτή.

Το Μέρος Α, αποτελείται από πέντε κεφάλαια. Στο κεφάλαιο 1, γίνεται μια εισαγωγή για το γενετικό προγραμματισμό, δίνοντας έμφαση στα τερματικά και συναρτησιακά σύμβολα που απαρτίζουν ένα γενετικό πρόγραμμα. Στο τέλος του κεφαλαίου παρουσιάζονται παραδείγματα προβλημάτων όπου εφαρμόζονται αλγόριθμοι γενετικού προγραμματισμού. Στο κεφάλαιο 2, παρουσιάζονται οι βασικοί γενετικοί τελεστές οι οποίοι χρησιμοποιούνται στην εξελικτική διαδικασία και οι οποίοι χρησιμοποιούνται και σε αυτή την εργασία. Στο κεφάλαιο 3, παρουσιάζεται η έννοια της συνάρτησης καταλληλότητας σε σχέση με τα προβλήματα γενετικού προγραμματισμού και δίνονται ορισμοί για τις πιο συνηθισμένες περιπτώσεις όπου έχουμε σαφή υπολογισμός της καταλληλότητας. Στο κεφάλαιο 4, παρουσιάζονται οι διαφορετικές μέθοδοι για την αρχικοποίηση του πληθυσμού των γενετικών προγραμμάτων στην περίπτωση όπου τα γενετικά προγράμματα αναπαρίστανται με δεντρική δομή. Το τελευταίο κεφάλαιο του πρώτου μέρους παρουσιάζονται τα σχήματα επιλογής των γενετικών προγραμμάτων που θα συμμετάσχουν στο σχηματισμό του νέου πληθυσμού (με την εφαρμογή σε αυτά των γενετικών τελεστών), καθώς και τα σχήματα με τα οποία επιλέγονται τα άτομα που θα περάσουν στην επόμενη γενιά.

Το Μέρος Β αποτελείται από ένα κεφάλαιο. Στο κεφάλαιο αυτό πραγματοποιείται μια σύντομη αναφορά στα χαρακτηριστικά των δικτύων ομότιμων κόμβων, δίνοντας έμφαση στα πρωτόκολλα πινάκων κατανεμημένου κατακερματισμού. Ακόμα, γίνεται μια

σύντομη περιγραφή στα πρωτόκολλα Chord και S-Chord στα οποία θα τροποποιηθεί ο αλγόριθμος αναζήτησης με την προσθήκη του γενετικού μηχανισμού.

Το Μέρος Γ, χωρίζεται σε τέσσερα κεφάλαια. Στο κεφάλαιο 7, πραγματοποιείται περιγραφή της γενετικής γλώσσας (πράξεις που υποστηρίζει, τύποι δεδομένων κ.ά.). Επιπλέον, περιγράφεται ο τρόπος λειτουργίας του διερμηνευτή για τη γλώσσα αυτή. Στο επόμενο κεφάλαιο, γίνεται μια θεωρητική παρουσίαση της προτεινόμενης εφαρμογής του γενετικού προγραμματισμού, των διαφόρων σημείων που επηρεάζουν την απόδοση καθώς και στο πως τροποποιούνται οι αλγόριθμοι αναζήτησης. Στο κεφάλαιο 9 παρουσιάζονται τα αποτελέσματα των προσομοιώσεων της υλοποίησης και τέλος στο κεφάλαιο 10 συνοψίζονται τα βασικότερα συμπεράσματα μαζί με κάποιες προτάσεις για μελλοντική εργασία.

Μέρος Α΄

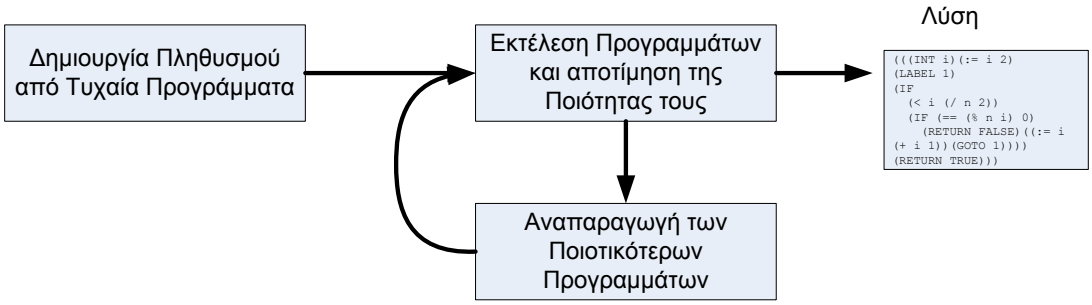
Γενετικός Προγραμματισμός

ΚΕΦΑΛΑΙΟ 1
ΓΕΝΕΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

1.1 Εισαγωγή

Ο γενετικός προγραμματισμός είναι μια προσπάθεια να απαντηθεί μια από τις βασικές ερωτήσεις στην επιστήμη της πληροφορικής (διατυπώθηκε από τον Arthur Samuel 1959): Πώς οι υπολογιστές μπορούν να λύσουν προβλήματα χωρίς να έχουν προγραμματιστεί ρητά; Πώς οι υπολογιστές μπορούν να κάνουν αυτό που χρειάζεται χωρίς να έχει σαφώς διατυπωθεί πως θα το κάνουν;

Ο γενετικός προγραμματισμός είναι μια επέκταση των γενετικών αλγορίθμων[1] όπου κάθε άτομο του πληθυσμού είναι ένα γενετικό πρόγραμμα και αποτελεί μια μέθοδο καθοδήγησης των υπολογιστών για την αυτόματη επίλυση ενός προβλήματος ξεκινώντας με την περιγραφή της λύσης σε υψηλό επίπεδο. Ο χώρος αναζήτησης του γενετικού προγραμματισμού εξαρτάται από το πεδίο του προβλήματος που επιλύεται και είναι ο χώρος όλων των προγραμμάτων που δημιουργούνται από τις συναρτήσεις και τα τερματικά σύμβολα του πεδίου του προβλήματος. Αρχικά, αναπτύσσεται ένας πληθυσμός από προγράμματα για την επίλυση του προβλήματος και διαδοχικά ο πληθυσμός αυτός μετατρέπεται σε ένα νέο πληθυσμό με την εφαρμογή σειράς από γενετικές πράξεις ανάλογες με αυτές που πραγματοποιούνται στη φύση.



Εικόνα 1.1: Επαναληπτική Ακολουθία Ανάπτυξης Γενετικών Προγραμμάτων.

Για την εφαρμογή γενετικού προγραμματισμού σε ένα πρόβλημα υπάρχουν 5 βασικά βήματα προετοιμασίας. Τα βήματα αυτά περιλαμβάνουν τον προσδιορισμό:

- Του συνόλου των τερματικών συμβόλων (terminal set),
- Του συνόλου των συναρτήσεων (primitive functions set),
- Της μετρικής καταλληλότητας (fitness measure),

[1] Οι γενετικοί αλγόριθμοι είναι προσαρμοστικοί ευρετικοί αλγόριθμοι βελτιστοποίησης και αναζήτησης, όπου βασίζονται στην ιδέα της φυσικής επιλογής και η συμπεριφορά τους απορρέει από τη μεταφορά μερικών από τους μηχανισμούς της εξέλιξης του φυσικού περιβάλλοντος.

- Τις παραμέτρους εκτέλεσης,
- Τη μέθοδο επιλογής αποτελέσματος και το κριτήριο τερματισμού.

Στη συνέχεια, θα σκιαγραφήσουμε κάθε ένα από αυτά.

1.2 Τερματικά σύμβολα και συναρτήσεις

Στο υποκεφάλαιο αυτό θα αναφερθούμε στο σύνολο των τερματικών συμβόλων *T*, στο σύνολο των συναρτήσεων *F* και σε κάποιες ιδιότητες αυτών των συνόλων.

Οι συναρτήσεις και τα τερματικά σύμβολα είναι τα βασικά συστατικά από τα οποία χτίζεται ένα πρόγραμμα γενετικού προγραμματισμού. Οι συναρτήσεις και τα τερματικά σύμβολα έχουν διαφορετικό ρόλο. Τα τερματικά σύμβολα “παρέχουν” μια τιμή στο σύστημα, ενώ οι συναρτήσεις επεξεργάζονται μια τιμή που εμφανίζεται ήδη στο σύστημα.

1.2.1 Σύνολο τερματικών συμβόλων

Το σύνολο των τερματικών συμβόλων (terminal set) μπορεί να θεωρηθεί ως είσοδο για τα υπό-ανάπτυξη γενετικά προγράμματα. Αποτελείται από τα δεδομένα εισόδου (inputs), τις σταθερές που παρέχονται στο γενετικό πρόγραμμα και τις συναρτήσεις χωρίς ορίσματα (zero argument functions) οι οποίες εκτελούνται από το γενετικό πρόγραμμα.

Τα δεδομένα εισόδου, οι σταθερές και οι κόμβοι χωρίς ορίσματα ονομάζονται τερματικά σύμβολα ή φύλλα επειδή τερματίζουν το κλαδί ενός γενετικού προγράμματος που περιγράφεται με δεντρική δομή. Ένα τερματικό σύμβολο βρίσκεται στο τέλος κάθε κλαδιού σε ένα γονίδιο με μορφή δέντρου (tree-structured) και αντιστοιχεί σε μια τιμή. Η τάξη (arity) μιας συνάρτησης είναι το πλήθος των δεδομένων εισόδου ή τα ορίσματα της συνάρτησης.

1.2.2 Σύνολο συναρτήσεων

Το σύνολο συναρτήσεων αποτελείται από τις δηλώσεις, τους τελεστές και τις συναρτήσεις που είναι διαθέσιμα στο σύστημα γενετικού προγραμματισμού.

Το σύνολο συναρτήσεων εξαρτάται από το είδος της εφαρμογής και επιλέγεται ώστε να ταιριάζει στο πεδίο του προβλήματος. Το εύρος των συναρτήσεων είναι πολύ μεγάλο (πρακτικά μπορεί να χρησιμοποιηθεί κάθε δομική μονάδα μιας γλώσσας προγραμματισμού).

1.2.3 Επιλογή συναρτήσεων και τερματικών συμβόλων

Τόσο οι συναρτήσεις όσο και τα τερματικά σύμβολα που θα επιλεγούν για το γενετικό πρόγραμμα, θα πρέπει να είναι ικανά να αναπαραστήσουν τη λύση του προβλήματος. Για παράδειγμα, αν η μόνη συνάρτηση στο σύνολο συναρτήσεων είναι η πρόσθεση τότε δεν μπορούν να λυθούν πολλά προβλήματα. Αντίστοιχα, είναι καλύτερα να μην είναι πολύ μεγάλο το σύνολο των συναρτήσεων, γιατί σε αυτή την περίπτωση μεγαλώνει ο χώρος αναζήτησης και κατ' επέκταση η εύρεση λύσης δυσκολεύει. Η λογική της εξοικονόμησης επιλέγεται και στην περίπτωση των σταθερών. Σε πολλές περιπτώσεις έχει αποδειχθεί πως οι σταθερές μπορούν να επιλύσουν δύσκολα προβλήματα, αφού ο γενετικός προγραμματισμός έχει την ικανότητα να συνδυάζει τις σταθερές που έχει στη διάθεση του, ώστε να δημιουργηθούν νέες σταθερές. Σε κάθε περίπτωση, στα πρώτα στάδια σχεδίασης ενός γενετικού προγράμματος δεν είναι απαραίτητο να σχεδιαστούν συγκεκριμένες συναρτήσεις και τερματικά σύμβολα, τα οποία να λύνουν με ακρίβεια το πρόβλημα.

1.2.4 Κλειστότητα

Γενικά, μια σημαντική ιδιότητα που πρέπει να έχει το σύνολο των συναρτήσεων είναι η κλειστότητα.

Η ιδιότητα της κλειστότητας απαιτεί κάθε συνάρτηση από το σύνολο των συναρτήσεων να μπορεί να δεχθεί, ως ορίσματα, κάθε τιμή και τύπο δεδομένων ο οποίος μπορεί να επιστραφεί από κάποια συνάρτηση του συνόλου των συναρτήσεων και κάθε τιμή και τύπο δεδομένων ο οποίος αντιστοιχεί σε τερματικό σύμβολο.

Δηλαδή, κάθε συνάρτηση θα πρέπει να είναι καλά ορισμένη και κλειστή σε κάθε συνδυασμό ορισμάτων που μπορεί να προκύψουν. Στην απλή περίπτωση όπου οι συναρτήσεις τις οποίες χειριζόμαστε είναι οι λογικές (boolean) συναρτήσεις AND, OR και NOT και το σύνολο των τερματικών συμβόλων είναι λογικές μεταβλητές (που δέχονται τιμές true και false), η κλειστότητα ικανοποιείται εύκολα. Κάτι τέτοιο όμως δεν είναι η συνηθισμένη περίπτωση, αφού τα προγράμματα αποτελούνται από μεταβλητές διαφορετικών τύπων δεδομένων (π.χ. ακέραιο, πραγματικό, λογικό, συμβολοσειρά), ενώ διαθέτουν τελεστές σύγκρισης και υπό συνθήκη διακλάδωσης.

Σε πολλά προγράμματα, οι αριθμητικές πράξεις οι οποίοι εφαρμόζονται σε αριθμητικές μεταβλητές δεν είναι δυνατές (π.χ. διαίρεση με το μηδέν, λογάριθμος του μηδέν). Επιπλέον, η τιμή που επιστρέφουν μαθηματικές συναρτήσεις είναι τύποι δεδομένων, οι οποίοι δεν είναι αποδεκτοί σε συγκεκριμένα προγράμματα (π.χ. τετραγωνική ρίζα

αρνητικού αριθμού). Ακόμα, μια λογική μεταβλητή δεν μπορεί να αποτελέσει όρισμα σε μια αριθμητική πράξη.

Ένα παράδειγμα, που δεν τηρεί την ιδιότητα της κλειστότητας είναι ο τελεστής της διαίρεσης. Ο τελεστής της διαίρεσης δεν μπορεί να λάβει ως είσοδο μηδέν, αφού κάτι τέτοιο θα οδηγήσει σε σφάλμα με αποτέλεσμα τον τερματισμό του γενετικού προγράμματος το οποίο δεν είναι επιθυμητό. Για να λυθεί αυτό το πρόβλημα με τον κλασικό τελεστή της διαίρεσης, μπορεί να οριστεί ένας νέος τελεστής της διαίρεσης ο οποίος επιστρέφει κανονικά το αποτέλεσμα σε όλες τις τιμές όπως ο κλασικός τελεστής αλλά για είσοδο μηδέν επιστρέφει κάποιο μεγάλο αριθμό, μηδέν, ή κάποια σταθερά που προσδιορίζει ασάφεια. Ένας τέτοιος εναλλακτικός ορισμός του τελεστή της διαίρεσης όπου στην περίπτωση όπου ο παρανομαστής είναι μηδέν επιστρέφει 1, φαίνεται παρακάτω:

Με den(ominator) τον παρανομαστή και num(erator) τον αριθμητή
((IF (== den 0) (1) (/ num den)))

Σε περίπτωση όπου διαθέτουμε τελεστή σύγκρισης και η λογική τιμή που προκύπτει ως αποτέλεσμα δεν είναι κατάλληλη, ο τελεστή μπορεί να τροποποιηθεί είτε χρησιμοποιώντας αριθμητική λογική είτε επαναορίζοντας κατάλληλα τον τελεστή συνθήκης. Εφαρμόζοντας αριθμητική λογική, αντί να επιστρέφεται αληθές ή ψευδές μπορεί να επιστρέφεται 1 ή -1. Αντίστοιχα, μπορούν να τροποποιηθούν οι τελεστές σύγκρισης ώστε να χειρίζονται αντί για λογικούς τύπους αριθμητικούς τύπους. Δηλαδή, αντί για IF συνθήκη αληθής, να γίνει IF συνθήκη θετική.

1.2.5 Επάρκεια

Μια ακόμα ιδιότητα που αφορά το σύνολο των συναρτήσεων και τα τερματικά σύμβολα είναι η επάρκεια (sufficiency).

Η ιδιότητα της επάρκειας απαιτεί το σύνολο των τερματικών συμβόλων και οι βασικές συναρτήσεις να μπορούν να εκφράσουν τη λύση του προβλήματος.

Ο χρήστης του γενετικού προγράμματος πρέπει να ξέρει πως κάποια σύνθεση των τερματικών συμβόλων και των συναρτήσεων μπορούν να παρέχουν μια λύση για το πρόβλημα. Το βήμα για να αναγνωριστούν οι μεταβλητές που επαρκούν για την επίλυση ενός προγράμματος είναι πολύ συχνό πρόβλημα. Ανάλογα με το πρόβλημα, μπορεί να είναι εύκολο να εντοπιστούν οι μεταβλητές ή να είναι αρκετά δύσκολο.

Σε κάποιος τομείς, οι απαιτήσεις για επάρκεια στις βασικές συναρτήσεις είναι γνωστές. Για παράδειγμα, στο πεδίο των λογικών συναρτήσεων, το σύνολο συναρτήσεων F=

(AND, OR, NOT) είναι γνωστό πως επαρκούν για να εκφραστεί κάθε λογική συνάρτηση. Αν αφαιρέσουμε τη συνάρτηση OR η ιδιότητα αυτή παραμένει, ενώ αν αφαιρέσουμε τη συνάρτηση NOT, οι εναπομείναντες συναρτήσεις δεν επαρκούν για να εκφράσουν πλέον όλες τις λογικές συναρτήσεις (π.χ. το XOR δεν μπορεί να εκφραστεί). Αντίθετα με το παραπάνω παράδειγμα, σε πολλές περιπτώσεις δεν είναι σαφές ποιες συναρτήσεις απαιτούνται (π.χ. μόνο πρόσθεση και αφαίρεση δεν επαρκούν για να εκφραστεί ο τρίτος νόμος του Kepler).

1.3 Συνάρτηση Καταλληλότητας

Στο γενετικό προγραμματισμό, ο πληθυσμός αποτελείται από γενετικά προγράμματα τα οποία αναπαράγονται γενετικά, ακολουθώντας την αρχή της επιβίωσης και της αναπαραγωγής του Δαρβίνου. Το πόσο καλά συμπεριφέρεται ένα άτομο του πληθυσμού μετριέται με τη συνάρτηση καταλληλότητας (fitness function). Η φύση της συνάρτησης καταλληλότητας ποικίλει ανάλογα με το πρόβλημα που αντιμετωπίζεται. Περισσότερα στοιχεία για τη συνάρτηση καταλληλότητας και τα χαρακτηριστικά της δίνονται στο κεφάλαιο που αναφέρεται στη συνάρτηση καταλληλότητας.

1.4 Παράμετροι Εκτέλεσης και Ολοκλήρωση Προσομοίωσης

Για την προσομοίωση των γενετικών προγραμμάτων, πρέπει να επιλεγούν αρκετές παράμετροι. Τις παραμέτρους αυτές μπορούμε να τις χωρίσουμε σε δύο κατηγορίες τις πιο σημαντικές (major parameters) και τις λιγότερο σημαντικές (minor parameters). Στις πιο σημαντικές παραμέτρους ανήκει το μέγεθος του πληθυσμού και μέγιστο πλήθος γενεών προσομοίωσης. Στις λιγότερες σημαντικές παραμέτρους ανήκει η πιθανότητα να εφαρμοστεί κάποιος γενετικός τελεστής (π.χ. διασταύρωση, μετάλλαξη, αναπαραγωγή), η πιθανότητα επιλογής εσωτερικού σημείου για διασταύρωση, μέγιστο βάθος δέντρων αναπαράστασης προγραμμάτων, μέγιστο μέγεθος αρχικών προγραμμάτων, το κριτήριο επιλογής των ατόμων στα οποία θα εφαρμοστούν οι γενετικοί τελεστές, το κριτήριο αναπαραγωγής στην επόμενη γενιά κ.ά.

Η προσομοίωση μπορεί να ολοκληρώνεται είτε μετά από κάποιο αριθμό γενεών είτε όταν βρεθεί ένα άτομο που δίνει βέλτιστη λύση (ή λύση που πληρεί κάποιο κριτήριο πρόωρου τερματισμού). Μια συνηθισμένη μέθοδο για την ανάδειξη αποτελέσματος (result designation) είναι η ανάδειξη ως λύση, του καλύτερου ατόμου της τρέχουσας γενιάς τη στιγμή του τερματισμού.

1.5 Παραδείγματα χρήσης γενετικών αλγορίθμων

Υπάρχουν αρκετά πεδία όπου μπορεί να βρει εφαρμογή ο γενετικός προγραμματισμός όπως: βέλτιστος έλεγχος (optimal control – εύρεση στρατηγικής ελέγχου η οποία χρησιμοποιεί τις μεταβλητές κατάστασης του συστήματος για την επιλογή τιμής σε κάποια μεταβλητή ελέγχου), σχεδιασμός (planning – εύρεση σχεδίου όπου με βάση πληροφορίες από το περιβάλλον επιλέγει ενέργειες που μεταβάλλουν την κατάσταση), εύρεση στρατηγικών παιγνίων (discovering game-playing strategies), αυτόματος προγραμματισμός (automatic programming – εύρεση σχέσης που λύνει συγκεκριμένο πρόβλημα ξεκινώντας από συγκεκριμένες τιμές και παράγοντας την επιθυμητή), εμπειρική εύρεση και πρόβλεψη (empirical discovery and forecasting), αντιστροφή προβλήματος (inverse problem – εύρεση αντίστροφης συνάρτησης), εύρεση μαθηματικών εκφράσεων (discovering mathematical identities) κ.ά. Στη συνέχεια θα γίνει αναφορά σε κάποια αντιπροσωπευτικά προβλήματα τα οποία επιλύονται με γενετικό προγραμματισμό.

Συμβολική Παλινδρόμηση (Symbolic Regression): το πρόβλημα αυτό περιλαμβάνει την εύρεση μιας μαθηματικής έκφρασης, με τη μορφή συμβόλων, η οποία θα παρέχει ταίριασμα όσο το δυνατόν καλύτερο, ανάμεσα σε ένα πεπερασμένο σύνολο τιμών δειγματοληψίας για τις ανεξάρτητες μεταβλητές και τις συσχετιζόμενες τιμές, για αυτές, των εξαρτημένων μεταβλητών.

Πολυπλέκτης 6-Bit και 11-Bit, και Ισοτιμία: Ο στόχος του προβλήματος 6-Bit και 11-Bit πολυπλέκτη, είναι ο εντοπισμός μιας λογικής συνάρτησης η οποία να εκτελεί πολυπλεξία για διευθύνσεις 2 και 3 bits. Στον πολυπλέκτη 6-Bit υπάρχουν δύο λογικές μεταβλητές διεύθυνσης και τέσσερις λογικές μεταβλητές δεδομένων. Ο πολυπλέκτης επιστρέφει την τιμή της μεταβλητής δεδομένων που περιγράφεται από τις μεταβλητές διεύθυνσης. Δεδομένου πως έχουμε 6 μεταβλητές (2 εισόδου και 4 εξόδου) το σύνολο των δυνατών περιπτώσεων είναι 64. Αντίστοιχη λογική έχει και η περίπτωση του 11-Bit πολυπλέκτη.

Το πρόβλημα της Ισοτιμίας είναι και αυτό ένα λογικό πρόβλημα που εφαρμόζεται σε ένα σύνολο από n μεταβλητές δεδομένων. Στο πρόβλημα αυτό, επιστρέφεται αληθές αν για τις μεταβλητές που εξετάζουμε, το πλήθος των μεταβλητών που έχει τιμή αληθής είναι άρτιο (ή περιττό). Για ένα πρόβλημα n-Ισοτιμίας υπάρχουν 2^N περιπτώσεις δοκιμών. Ο χώρος λύσεων στο πρόβλημα αυτό είναι διακριτός και γι' αυτό υπάρχει πεπερασμένος αριθμός από τιμές για τη συνάρτηση καταλληλότητας. Έτσι, υπάρχουν περιπτώσεις όπου η εφαρμογή γενετικών τελεστών (μετάλλαξη, διασταύρωση) μπορεί να μεταβάλλει

ριζικά τη δομή του γενετικού προγράμματος, αλλά όχι την τιμή της συνάρτησης καταλληλότητας.

Τεχνητό Μυρμήγκι (Artificial Ant): Το πρόβλημα του τεχνητού μυρμηγκιού είναι ένα σχετικά δύσκολο πρόβλημα για το γενετικό προγραμματισμό. Στόχος στο πρόβλημα αυτό είναι να βρεθεί ένας αυτοματοποιημένος αλγόριθμος (robotic planning) ο οποίος θα βρίσκει και θα καταναλώνει όσο το δυνατόν περισσότερη τροφή για το μυρμήγκι. Το μυρμήγκι μπορεί να κινηθεί ευθεία, να στρίψει δεξιά και να στρίψει αριστερά, ενώ όταν κινείται ευθεία, αν συναντήσει τροφή την καταναλώνει. Ο χώρος που κινείται το μυρμήγκι είναι ένα πλέγμα. Το πείραμα ολοκληρώνεται είτε μετά από κάποιο αριθμό βημάτων είτε αν καταναλωθεί όλη η διαθέσιμη τροφή. Η καταλληλότητα υπολογίζεται με βάση το πλήθος της τροφής που δεν έχει καταναλωθεί. Στο πρόβλημα αυτό δεν έχει σημασία η τιμή που επιστρέφεται από την εκτέλεση ενός κόμβου, αλλά πως μεταβάλλεται η κατάσταση του κόσμου του μυρμηγκιού από τις ενέργειες του μυρμηγκιού. Η μεταβολή στην κατάσταση του κόσμου εξαρτάται από το μονοπάτι στο πρόγραμμα που ακολουθείται κατά την εκτέλεση.

ΚΕΦΑΛΑΙΟ 2
ΓΕΝΕΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Τα άτομα που προκύπτουν κατά την αρχικοποίηση του πληθυσμού έχουν συνήθως χαμηλή καταλληλότητα. Τα άτομα αυτά μεταβάλλονται με την εφαρμογή των γενετικών τελεστών (εξελικτική διαδικασία). Οι βασικοί γενετικοί τελεστές παρουσιάζονται στη συνέχεια.

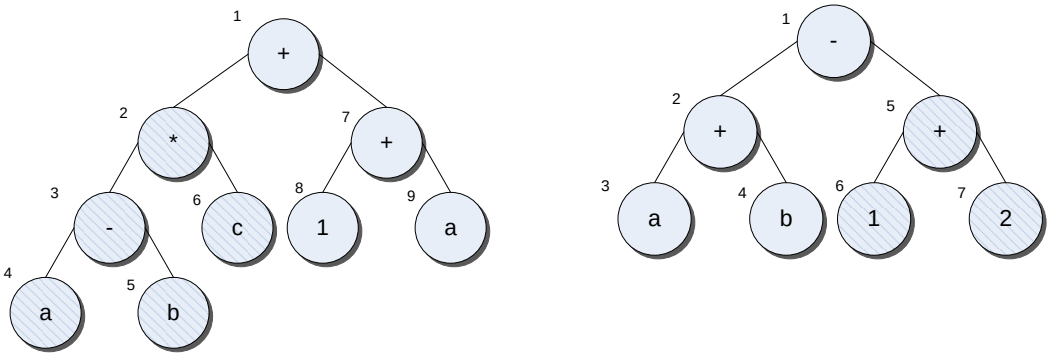
2.1 Διασταύρωση (Crossover)

Η πράξη της διασταύρωσης στα γενετικά προγράμματα οδηγεί στη δημιουργία ποικιλίας στον πληθυσμό, δημιουργώντας νέους απογόνους, συνδυάζοντας γενετικό υλικό από δύο γονείς, αλλάζοντας ένα κομμάτι από τον ένα με ένα κομμάτι από τον άλλο. Η διασταύρωση ξεκινά με δύο γονείς και ολοκληρώνεται με τη δημιουργία δύο απογόνων. Τα βήματα για την εφαρμογή του τελεστή στην περίπτωση της δεντρικής αναπαράστασης είναι τα παρακάτω:

- Αρχικά, επιλέγονται δύο γονείς από τον πληθυσμό για ζευγάρωμα, με βάση κάποια πολιτική επιλογής (οι γονείς, συνήθως, θα έχουν διαφορετικό μέγεθος και σχήμα).
- Στη συνέχεια, επιλέγεται ανεξάρτητα ένα τυχαίο σημείο σε κάθε γονέα για να αποτελέσει το σημείο διασταύρωσης. Η επιλογή σημείων διασταύρωσης, μπορεί να γίνει ώστε υποδέντρα τα οποία αποτελούνται από τερματικά σύμβολα, να επιλέγονται με μικρότερη πιθανότητα ή να λαμβάνεται υπόψη το μέγεθος του υποδέντρου και η απόσταση από τη ρίζα του γονέα[2].
- Ανταλλαγή των επιλεγμένων υποδέντρων στους γονείς. Δηλαδή, διαγραφή στον πρώτο γονέα του υποδέντρου κάτω από το σημείο διασταύρωσης και εισαγωγή εκεί του υποδέντρου που έχει προέλθει από το δεύτερο γονέα και αντίστοιχα για το δεύτερο γονέα.

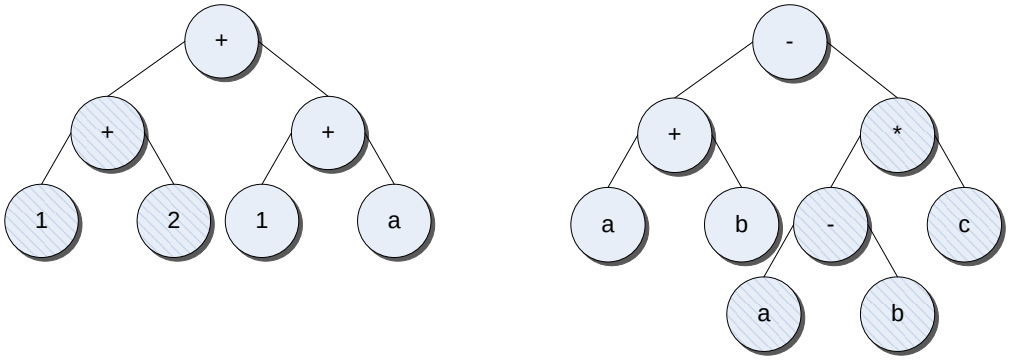
Στη συνέχεια ακολουθεί ένα παράδειγμα διασταύρωσης ανάμεσα σε δύο προγράμματα. Ας θεωρήσουμε πως το σύνολο των συναρτήσεων περιέχει τις αριθμητικές συναρτήσεις +, -, * και το σύνολο των τερματικών συμβόλων περιέχει τις μεταβλητές a, b, c και τις σταθερές 0, 1, 2, 3.

[2] Σε περίπτωση γραμμικής αναπαράστασης, ένα τυχαίο πλήθος από εντολές επιλέγεται από τον ένα πατέρα για ανταλλαγή, ενώ στην αναπαράσταση με γράφο, επιλέγονται ομάδες από κόμβους για ανταλλαγή.



Εικόνα 2.1: Παράδειγμα διασταύρωσης (πριν).

Οι γονείς που επιλέχθηκαν για διασταύρωση φαίνονται παραπάνω και θεωρούμε πως η αρίθμηση των κόμβων και στα δύο δέντρα είναι πρώτα κατά βάθος από αριστερά προς τα δεξιά. Ως σημεία διασταύρωσης επιλέγονται για τον πρώτο γονέα το σημείο με αριθμό 2 (κόμβος με χαρακτηριστικό *) και για το δεύτερο γονέα ο κόμβος με αριθμό 5 (κόμβος με χαρακτηριστικό +). Οι κόμβοι των υποδέντρων που πρόκειται να μετακινηθούν εμφανίζονται γραμμοσκιασμένοι. Τα γραμμοσκιασμένα κομμάτια ονομάζονται κομμάτια διασταύρωσης (crossover fragments), ενώ οι άλλοι κόμβοι υπόλοιπα (remainders). Οι δύο απόγονοι που προκύπτουν μετά την ανταλλαγή των υποδέντρων φαίνονται παρακάτω, με τα κομμάτια που ανταλλάχθηκαν να είναι γραμμοσκιασμένα.



Εικόνα 2.2: Παράδειγμα διασταύρωσης (μετά).

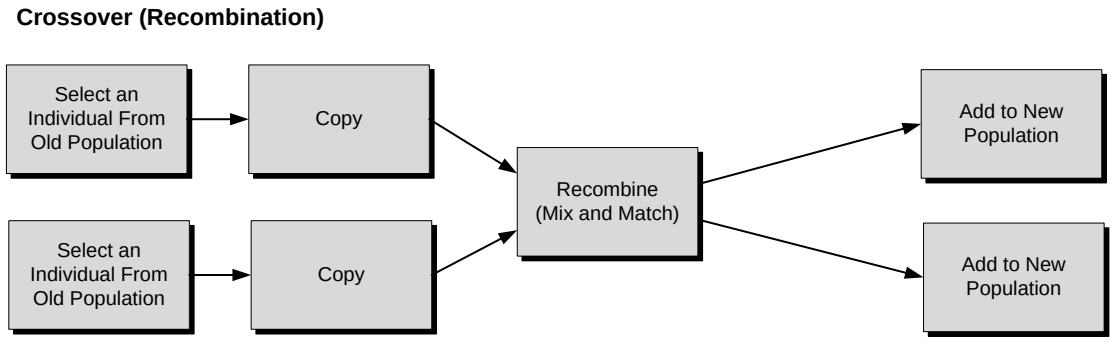
Κατά την εφαρμογή του τελεστή της διασταύρωσης φροντίζεται ώστε τα προγράμματα που παράγονται να είναι συντακτικά σωστά (κλειστότητα στους τελεστές αν απαιτείται, κατάλληλη επιλογή σημείων διασταύρωσης στους γονείς).

Οι περιπτώσεις όπου το σημείο διασταύρωσης που επιλέγεται σε ένα γονέα αντιστοιχεί σε τερματικό σύμβολο, οδηγούν στη δημιουργία απογόνου με μεγαλύτερο βάθος. Αν υπάρχει τερματικό σύμβολο στα σημεία διασταύρωσης που επιλέγονται και στους δύο γονείς, τότε η διασταύρωση οδηγεί στην ανταλλαγή αυτών των τερματικών συμβόλων

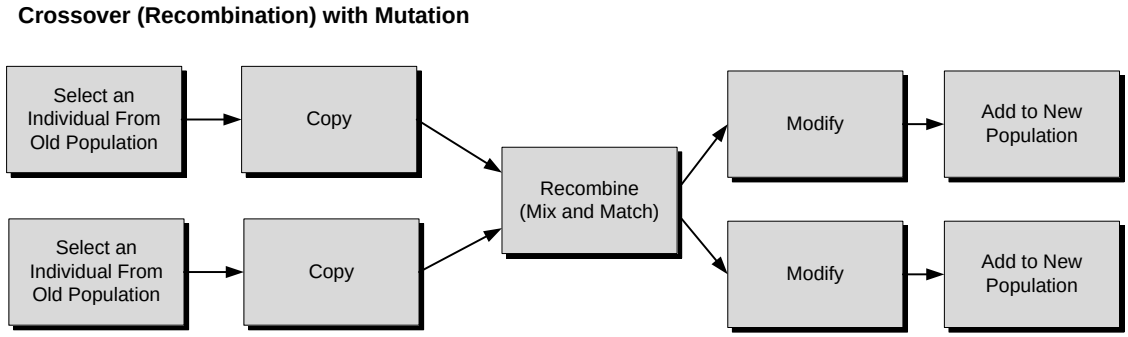
από δέντρο σε δέντρο. Το αποτέλεσμα από μια τέτοια διασταύρωση είναι παρόμοιο με μετάλλαξη σε κάποιο σημείο. Στην περίπτωση που σε κάποιο γονέα επιλεγεί ως σημείο διασταύρωσης η ρίζα, τότε κάτι τέτοιο, θα οδηγήσει πάλι στη δημιουργία απογόνου με μεγάλο βάθος. Αν οι ρίζες και των δύο γονέων επιλεγούν ως σημεία διασταύρωσης, τότε απλά οι απόγονοι που προκύπτουν θα είναι οι ίδιοι με τους γονείς. Όταν ένα άτομο από τον πληθυσμό ζευγαρώνει με τον εαυτό του ή αν δύο όμοια άτομα ζευγαρώνουν μαζί τότε, γενικά, οι δύο απόγονοι που θα προκύψουν θα είναι διαφορετικά άτομα (δεδομένου πως τα σημεία διασταύρωσης θα είναι διαφορετικά στους δύο γονείς).

Ο τρόπος με τον οποίο ζευγαρώνουν τα άτομα στον γενετικό προγραμματισμό έχει ως αποτέλεσμα, αν ένα άτομο έχει καλή καταλληλότητα σε σχέση με άλλα άτομα του πληθυσμού, η διασταύρωση θα οδηγήσει στην εμφάνιση πολλών αντιγράφων αυτού του “καλού” ατόμου. Έτσι, δημιουργείται η τάση να συγκλίνει ο πληθυσμός. Σε περίπτωση που η σύγκλιση αυτή δεν οδηγήσει σε ένα καλό αποτέλεσμα, ονομάζεται πρόωρη σύγκλιση (premature convergence). Κάτι τέτοιο μπορεί να συμβεί εάν υποβέλτιστα (suboptimal) άτομα εμφανίσουν πολύ καλή καταλληλότητα σε σχέση με τα υπόλοιπα άτομα του πληθυσμού, οπότε και ο γενετικός αλγόριθμος αποτυγχάνει να εντοπίσει το καθολικό βέλτιστο (global optimum). Γενικά, η φύση της διασταύρωσης είναι τέτοια που δεν είναι πιθανό να οδηγήσει σε σύγκλιση τον πληθυσμό.

Για τους απογόνους που προκύπτουν από τη διασταύρωση, ορίζεται ένα μέγιστο μέγεθος (είτε βάθος δέντρου είτε πλήθος κόμβων). Αυτό εμποδίζει τη σπατάλη πόρων σε πολύ μεγάλα προγράμματα. Αν μια διασταύρωση ανάμεσα σε δύο γονείς δημιουργήσει έναν απόγονο που ξεπερνά το όριο που ορίστηκε, τότε η διασταύρωση ματαιώνεται και ένας από τους δύο γονείς εισάγεται στην επόμενη γενιά. Αν και οι δύο απόγονοι ξεπερνούν το όριο τότε και οι δύο γονείς θα αποτελούν μέρος της επόμενης γενιάς.



Εικόνα 2.3: Μέθοδος διασταύρωσης.



Εικόνα 2.4: Μέθοδος διασταύρωσης με μετάλλαξη.

2.2 Μετάλλαξη (Mutation)

Η μετάλλαξη εφαρμόζεται σε ένα άτομο του πληθυσμού. Συνήθως, μετά τη διασταύρωση, σε κάθε απόγονο που δημιουργήθηκε, εφαρμόζεται μετάλλαξη με μικρή πιθανότητα (η πιθανότητα μετάλλαξης είναι παράμετρος της εκτέλεσης), χωρίς όμως να αποκλείεται διασταύρωση και μετάλλαξη να εφαρμόζονται ανεξάρτητα. Όταν κάποιο άτομο του πληθυσμού έχει επιλεγεί για μετάλλαξη, ένας τύπος μετάλλαξης επιλέγει έναν κόμβο στο δέντρο και αντικαθιστά το επιλεγμένο υποδέντρο με ένα καινούργιο τυχαία δημιουργημένο υποδέντρο. Το υποδέντρο αυτό έχει δημιουργηθεί με τους ίδιους κανόνες και τους ίδιους περιορισμούς με τους οποίους δημιουργήθηκαν τα αρχικά άτομα του πληθυσμού. Στη συνέχεια το αλλαγμένο άτομο εισάγεται πάλι στον πληθυσμό. Η μετάλλαξη εισάγει ποικιλία στον πληθυσμό (π.χ. ένα καινούργιο τελεστή), ο οποίος τείνει να συγκλίνει πρόωρα.

Εικόνα 2.5: Μέθοδος μετάλλαξης

Εικόνα 2.6: Μέθοδος αναπαραγωγής

2.3 Αναπαραγωγή (Reproduction)

Η πράξη της αναπαραγωγής είναι πολύ απλή και εφαρμόζεται σε ένα άτομο. Ένα άτομο από τον πληθυσμό επιλέγεται με βάση κάποιο κριτήριο επιλογής, δημιουργείται ένα

αντίγραφο χωρίς αλλαγές και εισάγεται στον πληθυσμό της νέας γενιάς. Έτσι υπάρχουν στον πληθυσμό δύο εκδοχές του συγκεκριμένου ατόμου.

2.4 Άλλοι Τελεστές

Στις προηγούμενες παραγράφους πραγματοποιήθηκε αναφορά στους βασικούς γενετικούς τελεστές, οι οποίοι και χρησιμοποιήθηκαν στα πλαίσια αυτής της εργασίας. Εκτός από αυτούς τους βασικούς τελεστές υπάρχουν και κάποιοι δευτερεύοντες.

Μετάθεση (permutation): Ο τελεστής αυτός εφαρμόζεται σε ένα άτομο και το αποτέλεσμα είναι ένας καινούργιος απόγονος. Αρχικά, επιλέγεται ένα εσωτερικό σημείο (συνάρτηση) στη δεντρική αναπαράσταση του προγράμματος. Αν η συνάρτηση έχει k ορίσματα τότε, επιλέγεται μια τυχαία μετάθεση από το σύνολο των $k!$ δυνατών μεταθέσεων. Στη συνέχεια, τα ορίσματα της συνάρτησης, στο σημείο επιλογής, μετατίθενται σύμφωνα με τη μετάθεση που επιλέχθηκε. Ανάλογα με τη φύση της συνάρτησης είναι πιθανό η μετάθεση των ορισμάτων να μην οδηγήσει σε αλλαγή.

Επεξεργασία (editing): Πρόκειται για τελεστή που εφαρμόζεται σε ένα άτομο. Ένα σύνολο από προκαθορισμένους κανόνες επεξεργασίας (editing rules) εφαρμόζεται σε κάθε άτομο του πληθυσμού. Οι κανόνες αυτοί έχουν σκοπό να απλοποιήσουν ένα άτομο (π.χ. η αριθμητική έκφραση (+ 1 3) μπορεί να αντικατασταθεί από το αποτέλεσμα της 4).

Ενθυλάκωση (encapsulation): Η πράξη αυτή εφαρμόζεται σε ένα άτομο και αποσκοπεί στην αυτόματη αναγνώριση ενός εν δυνάμει χρήσιμου υποδέντρου. Το υποδέντρο αυτό μπορεί να χρησιμοποιηθεί ανεξάρτητα στη συνέχεια.

ΚΕΦΑΛΑΙΟ 3
ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ-ΑΞΙΟΛΟΓΗΣΗΣ (FITNESS FUNCTION)

3.1 Εισαγωγή

Η συνάρτηση καταλληλότητας ή αξιολόγησης (fitness function) είναι ζωτικής σημασίας, αφού αυτή είναι που καθοδηγεί την εξέλιξη του πληθυσμού σε ένα γενετικό πρόβλημα και σε μεγάλο βαθμό εξαρτάται από το πρόβλημα που επιλύεται.

Στη φύση, η καταλληλότητα ενός ατόμου είναι η πιθανότητα να επιβιώσει μέχρι την ηλικία αναπαραγωγής και να αναπαραχθεί με επιτυχία. Στον κόσμο των αλγορίθμων υπάρχουν διάφοροι τρόποι για να μετρηθεί η καταλληλότητα και οι μετρήσεις αυτές κατευθύνουν τις ενέργειες που θα εφαρμοστούν στον πληθυσμό. Είναι σημαντικό, η συνάρτηση αυτή, όχι μόνο να ανταμείβει τις σωστές λύσεις αλλά κατά προτίμηση να ανταμείβει τη βελτίωση κατά την εκτέλεση του γενετικού προγράμματος από τη δημιουργία του αρχικού πληθυσμού έως τον τερματισμό της προσομοίωσης. Μια συνάρτηση καταλληλότητας πρέπει να μπορεί να δίνει συνεχή και κλιμακούμενα αποτελέσματα για το πόσο καλά λειτουργεί ένα πρόγραμμα.

Η συνάρτηση καταλληλότητας, επηρεάζει σημαντικά την εξέλιξη του προβλήματος, γι' αυτό και πρέπει να είναι προσεκτικά σχεδιασμένη με αρκετά επίπεδα διάκρισης στα αποτελέσματα που δίνει για μια εκτέλεση. Δυαδικές απαντήσεις, όπως ταιριάζει ή δεν ταιριάζει, δε θα δουλέψουν ικανοποιητικά. Η επιλογή για το ποια άτομα του πληθυσμού θα ζευγαρώσουν πρέπει να επιλέγεται με βάση πόσο κοντά είναι στη λύση. Διαφορετικά, οι απόγονοι που θα προκύψουν το πιο πιθανό είναι να μην έχουν μεγαλύτερη καταλληλότητα από τα άτομα που αποτελούσαν τους προγόνους τους.

Ο γενετικός προγραμματισμός δε ψάχνει ούτε ένα μονοπάτι στο χώρο αναζήτησης ούτε ψάχνει, εξαντλητικά, όλο το χώρο των γενετικών προγραμμάτων. Είναι περισσότερο, μια πρώτα κατά πλάτος αναζήτηση, η οποία προχωρά σε κάθε βήμα προς τα καλύτερα μονοπάτια, επιλέγοντας τους καλύτερους κόμβους και απορρίπτοντας τους υπόλοιπους. Έτσι, το πλήθος των κόμβων που διερευνάται παραμένει ελεγχόμενο, ακόμα και αν υπάρχουν πολλά μονοπάτια και η αναζήτηση φτάνει σε μεγάλο βάθος.

Στις περισσότερες περιπτώσεις ο υπολογισμός της συνάρτησης καταλληλότητας απορροφά μεγάλο ποσοστό από το χρόνο εκτέλεσης γι' αυτό και η αποτελεσματικότητα της συνάρτησης είναι πολύ σημαντικός παράγοντας. Ο χρόνος εκτέλεσης μπορεί να

γίνει υπερβολικός σε περιπτώσεις όπου τα προγράμματα χρησιμοποιούν επαναληπτικές δομές ή αναδρομή.

Ορισμός: *Η συνάρτηση καταλληλότητας είναι η μετρική που χρησιμοποιείται από το γενετικό πρόγραμμα κατά τη διαδικασία της προσομοίωσης της εξέλιξης και εξετάζει πόσο καλά έχει υπολογίσει το πρόγραμμα την έξοδο, από εισόδους που αποτελούν χαρακτηριστικά του πεδίου εκμάθησης.*

Ο στόχος της ύπαρξης μιας συνάρτησης καταλληλότητας είναι να παρέχει ανάδραση στον “εκπαιδευόμενο” αλγόριθμο αναφορικά με τα ποια άτομα θα πρέπει να έχουν μεγαλύτερη πιθανότητα να πολλαπλασιαστούν, να αναπαραχθούν και ποια άτομα έχουν μεγαλύτερη πιθανότητα να απομακρυνθούν από τον πληθυσμό.

Η καταλληλότητα μπορεί να μετρηθεί με διάφορους τρόπους, κάποιους σαφής και κάποιους ασαφής. Η πιο συνηθισμένη προσέγγιση για τη μέτρηση της καταλληλότητας είναι η δημιουργία μιας σαφούς συνάρτησης μέτρησης για τα άτομα του πληθυσμού. Η προσέγγιση αυτή χρησιμοποιείται στην πλειοψηφία των εφαρμογών των τυπικών γενετικών αλγορίθμων. Για κάθε άτομο υπολογίζεται μια τιμή καταλληλότητας που έχει υπολογιστεί από μια καλώς ορισμένη, σαφή εξελικτική διαδικασία. Στη συνέχεια, θα δοθεί ένας ορισμός της καταλληλότητας, θα γίνει αναφορά σε τέσσερις μετρικές σαφούς υπολογισμού της καταλληλότητας και θα παρουσιαστεί ένα παράδειγμα υπολογισμού της καταλληλότητας με τη βοήθεια δειγμάτων εκπαίδευσης. Οι μετρικές αυτές είναι: ακατέργαστη καταλληλότητα (raw fitness), τυποποιημένη καταλληλότητα (standardized fitness), προσαρμοσμένη καταλληλότητα (adjusted fitness) και κανονικοποιημένη καταλληλότητα (normalized fitness).

3.2 Ακατέργαστη Καταλληλότητα (Raw Fitness)

Ορισμός: *Η ακατέργαστη καταλληλότητα είναι η μέτρηση της καταλληλότητας που δηλώνεται από τη φυσική ορολογία του προβλήματος.*

Για παράδειγμα, η ακατέργαστη καταλληλότητα στο πρόβλημα της τεχνητής νοημοσύνης με το μυρμήγκι που αναζητά τροφή, είναι ο αριθμός κομματιών φαγητού τα οποία φαγώθηκαν από το μυρμήγκι. Όσο περισσότερη τροφή, τόσο καλύτερα.

Ένας συνηθισμένος ορισμός της ακατέργαστης καταλληλότητας είναι ο ορισμός της ως το σφάλμα, με τη βοήθεια δειγμάτων εκπαίδευσης (fitness cases). Η ακατέργαστη καταλληλότητα ενός ατόμου είναι το άθροισμα των σφαλμάτων, για όλα τα δείγματα εκπαίδευσης, ανάμεσα στην τιμή που επιστρέφει το πρόγραμμα για το σύνολο των ορισμάτων που σχετίζονται με το συγκεκριμένο δείγμα εκπαίδευσης και στη σωστή τιμή

που αντιστοιχεί στο συγκεκριμένο δείγμα εκπαίδευσης. Στην περίπτωση όπου η ακατέργαστη καταλληλότητα είναι το σφάλμα, τότε η ακατέργαστη καταλληλότητα $r(i,t)$ ενός προγράμματος i σε μια γενιά t δίνεται από τη σχέση:

$$r(i,t) = \sum_{j=1}^{M} \left| S(i,j) - C(i,j) \right|$$

με $S(i,j)$ την τιμή που επιστρέφει το πρόγραμμα i για το δείγμα εκπαίδευσης j και $C(i,j)$ η σωστή για το ίδιο δείγμα.

Στην περίπτωση αυτή, μικρές βελτιώσεις στο πόσο καλά έχει μάθει ένα πρόγραμμα το πεδίο εκμάθησης σχετίζονται με μικρές βελτιώσεις στην καταλληλότητα που υπολογίζεται για το πρόγραμμα, ενώ μεγάλες βελτιώσεις στο πόσο καλά έχει μάθει ένα πρόγραμμα το πεδίο εκμάθησης σχετίζονται με μεγάλες βελτιώσεις στη καταλληλότητα που υπολογίζεται για το πρόγραμμα.

3.3 Τυποποιημένη καταλληλότητα (Standardized Fitness)

Ορισμός: *Η τυποποιημένη καταλληλότητα s(i,t) επαναδιατυπώνει την ακατέργαστη καταλληλότητα έτσι ώστε μια μικρότερη αριθμητικά τιμή να υποδηλώνει ένα καλύτερο άτομο (και η τιμή μηδέν το καλύτερο άτομο).*

Εάν η ακατέργαστη καταλληλότητα αυξάνεται δεδομένου ότι ένα άτομο βελτιώνεται, η τυποποιημένη καταλληλότητα ενός ατόμου είναι η μέγιστη ακατέργαστη καταλληλότητα (δηλαδή η ικανότητα του καλύτερου ατόμου πληθυσμός) μείον την ακατέργαστη καταλληλότητα του ατόμου. Για παράδειγμα, αν σε κάποιο πρόβλημα σκοπός είναι να ελαχιστοποιηθεί το σφάλμα, τότε μια μικρότερη τιμή ακατέργαστης καταλληλότητας είναι καλύτερη (με 0 να είναι η καλύτερη). Αν μια μικρότερη τιμή ακατέργαστης καταλληλότητας είναι καλύτερη, τότε η τυποποιημένη καταλληλότητα (*std*) είναι ίση με την ακατέργαστη καταλληλότητα (*raw*) για το πρόβλημα αυτό, δηλαδή $std(i,t) = raw(i,t)$. Αν θέλουμε η καλύτερη τιμή της τυποποιημένης καταλληλότητας να είναι μηδέν, και κάτι τέτοιο δεν ισχύει, τότε αυτό μπορεί να συμβεί με τη βοήθεια μιας σταθεράς, δηλαδή $std(i,t) = constant - raw(i,t)$.

3.4 Προσαρμοσμένη Καταλληλότητα (Adjusted Fitness)

Η προσαρμοσμένη καταλληλότητα υπολογίζεται από την τυποποιημένη καταλληλότητα ως εξής:

$$adj(i,t) = \frac{1}{1 + std(i,t)}$$

Με $std(i,t)$ την τυποποιημένη καταλληλότητα για κάθε άτομο i τη στιγμή t.

Η προσαρμοσμένη καταλληλότητα είναι μεταξύ 0 και 1, είναι μεγαλύτερη για καλύτερα άτομα στον πληθυσμό και έχει το πλεονέκτημα πως τονίζει τη σημασία των μικρών διαφορών στη τιμή της τυποποιημένης καταλληλότητας όσο αυτή τείνει προς το 0 (διαχωρισμός ατόμων με τυποποιημένη καταλληλότητα κοντά στο μηδέν). Έτσι, όσο ο πληθυσμός βελτιώνεται, δίνεται μεγαλύτερη έμφαση στις μικρές διαφορές που διαχωρίζουν ένα καλό άτομο από ένα πολύ καλό.

3.5 Κανονικοποιημένη Καταλληλότητα (Normalized Fitness)

Η κανονικοποιημένη καταλληλότητα $norm(i,t)$ υπολογίζεται από την προσαρμοσμένη καταλληλότητα $adj(i,t)$ ως εξής:

$$norm(i,t) = \frac{adj(i,t)}{\sum_{k=1}^{M} adj(k,t)}$$

Η κανονικοποιημένη καταλληλότητα έχει τρία επιθυμητά χαρακτηριστικά:

- Έχει τιμή μεταξύ 0 και 1
- Είναι μεγαλύτερη για καλύτερα άτομα του πληθυσμού
- Το άθροισμα των τιμών της είναι 1.

3.6 Υπολογισμός Καταλληλότητας με Δείγμα Εκπαίδευσης

Ο προσδιορισμός της συνάρτησης καταλληλότητας, εξαρτάται σε μεγάλο βαθμό από το ίδιο το πρόβλημα που εξετάζεται. Σε πολλά προβλήματα η συνάρτηση καταλληλότητας ορίζεται σαφώς με τη βοήθεια του σφάλματος που προκύπτει από την εκτέλεση των προγραμμάτων. Όσο πιο κοντά στο μηδέν είναι το σφάλμα τόσο καλύτερο είναι το πρόγραμμα. Σε ένα πρόβλημα βέλτιστου ελέγχου, η καταλληλότητα μπορεί να υπολογίζεται με βάση το χρόνο που απαιτείται ώστε να φτάσει το σύστημα στην επιθυμητή κατάσταση. Όσο λιγότερος χρόνος τόσο καλύτερα. Σε περιπτώσεις όπου στόχος είναι η αναγνώριση ή ταξινόμηση προτύπων, η καταλληλότητα μπορεί να μετριέται με βάση το πλήθος των περιπτώσεων το οποίο χειρίζεται σωστά ή λανθασμένα. Η συσχέτιση (correlation) χρησιμοποιείται συχνά ως μετρική καταλληλότητας. Ακόμα, η καταλληλότητα, ανάλογα το πρόβλημα, μπορεί αν μετρηθεί με τη βοήθεια εντροπίας, την ικανοποίησης δοκιμών αλλά και με συνδυασμός παραπάνω από ενός χαρακτηριστικών (multi-objective fitness) όπως ορθότητα, αποτελεσματικότητα, ταχύτητα κ.ά.

Στην περίπτωση μας, θα θεωρήσουμε πως η συνάρτηση καταλληλότητας υπολογίζεται με βάση κάποιο δείγμα εκπαίδευσης. Τα δείγματα εκπαίδευσης, χρησιμοποιούνται για την αποτίμηση των προγραμμάτων του πληθυσμού σε σχέση με ένα πλήθος από αντιπροσωπευτικές περιπτώσεις, επαρκής, ώστε να προκύπτει ένα εύρος από διαφορετικές αριθμητικές τιμές καταλληλότητας.

Το δείγματα εκπαίδευσης είναι, συνήθως, ένα μικρό και πεπερασμένο δείγμα από το πεδίο ορισμού (που συνήθως είναι πολύ μεγάλο ή και άπειρο). Οι τιμές (δείγματα διαφορετικών τιμών μιας ανεξάρτητης μεταβλητής) μπορεί να επιλέγονται τυχαία ή με κάποιο προκαθορισμένο τρόπο, αλλά πρέπει να είναι αντιπροσωπευτικές ως προς το πεδίο ορισμού, αφού αποτελούν τη βάση για τη γενίκευση των αποτελεσμάτων σε όλο το πεδίο ορισμού. Είναι δυνατόν, να περιοριστεί η επίδραση της επιλογής συγκεκριμένων δειγμάτων εκπαίδευσης με τον υπολογισμό της καταλληλότητας χρησιμοποιώντας διαφορετικά δείγματα εκπαίδευσης σε κάθε γενιά. Το πιθανό όφελος από μια τέτοια προσέγγιση αντισταθμίζεται από τη δυσκολία για σύγκριση της απόδοσης ενός ατόμου με το πέρας των γενεών. Γι’ αυτό, τα δείγματα εκπαίδευσης επιλέγονται στην αρχή της εκτέλεσης και δεν ποικίλουν από γενιά σε γενιά.

Στη συνέχεια ακολουθεί ένα παράδειγμα όπου αναζητείται μια συνάρτηση η οποία να ικανοποιεί τα δείγματα εκπαίδευσης του παρακάτω Πίνακα. Κάθε ζευγάρι εισόδου/εξόδου αντιστοιχεί σε ένα στιγμιότυπο εκπαίδευσης. Όλα τα δείγματα εκπαίδευσης αποτελούν το σύνολο εκπαίδευσης.

Πίνακας 3.1: Δείγματος εκπαίδευσης

Δείγματα Εκπαίδευσης	**Είσοδος**	**Έξοδος**
Δείγμα 1	1	2
Δείγμα 2	2	6
Δείγμα 3	4	20
Δείγμα 4	7	56
Δείγμα 5	9	90

Αν θεωρήσουμε πως το γενετικό πρόγραμμα έχει ως σκοπό να αναπτύξει ένα πρόγραμμα που ακολουθεί το πρότυπο του παραπάνω Πίνακα, δηλαδή, ένα πρόγραμμα το οποίο θα προβλέπει τις τιμές που αντιστοιχούν στη στήλη της εξόδου γνωρίζοντας μόνο τις τιμές της στήλης εισόδου. Στη συγκεκριμένη περίπτωση, ένα πρόγραμμα που ταιριάζει ακριβώς σε αυτό το σύνολο εκπαίδευσης είναι το πρόγραμμα που αντιστοιχεί στη συνάρτηση $f(x) = x^2 + x$.

ΚΕΦΑΛΑΙΟ 4
ΑΡΧΙΚΟΠΟΙΗΣΗ ΠΛΗΘΥΣΜΟΥ ΓΕΝΕΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Το πρώτο βήμα πριν πραγματοποιηθούν δοκιμές με γενετικό προγραμματισμό είναι η αρχικοποίηση του πληθυσμού, δηλαδή η δημιουργία μιας ποικιλίας γενετικών προγραμμάτων τα οποία θα χρησιμοποιηθούν στην εξελικτική διαδικασία. Η διαδικασία αυτή εξαρτάται από τη δομή αναπαράστασης που χρησιμοποιείται για το γενετικό πρόγραμμα.

Μια από τις παραμέτρους που πρέπει να ληφθεί υπόψη κατά τις δοκιμές είναι το μέγιστο επιτρεπτό μέγεθος ενός προγράμματος. Στην περίπτωση της δεντρικής αναπαράστασης, η παράμετρος αυτή εκφράζεται από το μέγιστο βάθος του δέντρου ή από το μέγιστο αριθμό κόμβων στο δέντρο.

Ορισμός: *Το βάθος ενός κόμβου είναι ο ελάχιστος αριθμός κόμβων τον οποίο πρέπει να διασχίσουμε για να διατρέξουμε τη διαδρομή από τη ρίζα προς τον επιλεγμένο κόμβο.*

Η παράμετρος μέγιστου βάθους (max_depth) είναι η μεγαλύτερη απόσταση η οποία επιτρέπεται να υπάρχει ανάμεσα στον κόμβο ρίζα και στους πιο απομακρυσμένους τερματικούς κόμβους σε ένα άτομο. Για κόμβους με δύο παιδιά, το μέγεθος του δέντρου θα έχει μέγιστο πλήθος κόμβων $2^{\max\_depth}$ (Για γραμμική αναπαράσταση γενετικών προγραμμάτων, η παράμετρος αυτή ονομάζεται μέγιστο μήκος και είναι ο μέγιστος αριθμός εντολών που επιτρέπεται να υπάρχουν σε ένα πρόγραμμα. Στην αναπαράσταση με γράφο, το μέγιστο πλήθος κόμβων αποτελεί το μέγεθος του προγράμματος).

Γενικά, όμοια άτομα στον αρχικό πληθυσμό είναι μη παραγωγικά, αφού σπαταλούν υπολογιστικούς πόρους και μειώνουν τη γενετική ποικιλία στον πληθυσμό (είναι επιθυμητό να αποφεύγονται τα όμοια άτομα αλλά δεν είναι απαραίτητο). Στο γενετικό προγραμματισμό, διπλά άτομα είναι αρκετά πιθανό να εμφανιστούν στην αρχική (τυχαία) γενιά, όταν τα δέντρα είναι μικρά (στις μεθόδους ramped half-and-half και grow, που θα αναφέρουμε παρακάτω, για κάποια ποσοστά του πληθυσμού). Γι' αυτό, κάθε αρχικό πρόγραμμα ελέγχεται κατά πόσο είναι μοναδικό πριν εισαχθεί στον αρχικό πληθυσμό. Εφόσον παραχθεί διπλό άτομο, η γενετική διαδικασία επαναλαμβάνεται μέχρι να παραχθεί μοναδικό άτομο.

Η ποικιλία (variety) ενός πληθυσμού είναι το ποσοστό των ατόμων για τα οποία δεν εμφανίζονται ίδια στον πληθυσμό. Είναι επιθυμητό, να υπάρχει όσο γίνεται μεγαλύτερη

ποικιλία στον αρχικό πληθυσμό (περισσότερο γενετικό υλικό). Αν πραγματοποιείται έλεγχος για διπλότυπα, η ποικιλία σε ένα τυχαίο πληθυσμό είναι 100%. Σε επόμενες γενιές, η δημιουργία όμοιων ατόμων είναι έμφυτο κομμάτι της γενετικής διαδικασίας.

Ακόμα, η εισαγωγή ικανών ατόμων στον αρχικό πληθυσμό τυχαίων ατόμων (τα οποία όπως είναι λογικό έχουν χαμηλή καταλληλότητα), δεν αποτελεί καλή επιλογή γιατί θα οδηγήσει στην επόμενη γενιά σε επικράτηση στον πληθυσμό, αντιγράφων και απογόνων των πιο ικανών ατόμων. Έτσι, σύντομα, ο πληθυσμός θα μοιάζει σε μεγάλο ποσοστό με το να αρχίζαμε με ένα πληθυσμό από τα ικανά άτομα. Για να δοκιμαστεί κάτι τέτοιο, θα πρέπει το 100% του αρχικού πληθυσμού να αποτελείται από άτομα με σχετικά παρόμοια καταλληλότητα.

4.1 Αρχικοποίηση δεντρικής δομής

Η αρχικοποίηση μιας δεντρικής δομής είναι απλή. Κάθε δέντρο αποτελείται από βασικές μονάδες, τις συναρτήσεις και τα τερματικά σύμβολα. Αν θεωρήσουμε πως έχουμε ήδη επιλέξει το σύνολο των τερματικών συμβόλων και το σύνολο των συναρτήσεων, οι τρόποι για την αρχικοποίηση δεντρικών δομών είναι η πλήρης (full), η εξέλιξη (grow) και η ramped half-and-half μέθοδοι.

Στη συνέχεια, παρουσιάζονται τα πρώτα βήματα για τη δημιουργία ενός τυχαίου δέντρου. Αν θεωρήσουμε πως διαθέτουμε το σύνολο συναρτήσεων $F = \{+, -, *\}$ και το σύνολο τερματικών συμβόλων $T = \{a, b, c, 0, 1, 2\}$. Αρχικά, θεωρούμε πως από το σύνολο των συναρτήσεων F, επιλέχθηκε η συνάρτηση + που δέχεται δύο ορίσματα. Για κάθε ένα από τα ορίσματα της συνάρτησης +, επιλέγονται στοιχεία από το σύνολο $F \cup T$, ώστε να είναι παιδιά της. Αν για κάποιο από τα παιδιά επιλεγεί στοιχείο από το σύνολο των συναρτήσεων, τότε η διαδικασία συνεχίζεται αναδρομικά όπως παραπάνω. Για παράδειγμα, από το $F \cup T$ επιλέγεται το σύμβολο – (εσωτερικό σύμβολο του δέντρου), που δέχεται δύο ορίσματα, ως αριστερό παιδί για τον κόμβο +. Στην περίπτωση που επιλεγεί κάποιο τερματικό σύμβολο, η ανάπτυξη του συγκεκριμένου κλαδιού ολοκληρώνεται και η διαδικασία τερματίζει για το σημείο αυτό. Η διαδικασία αυτή συνεχίζεται αναδρομικά από τα αριστερά προς τα δεξιά μέχρι όλα τα κλαδιά να τερματίζουν σε τερματικά σύμβολα.

Είναι δυνατόν κάποιο συναρτησιακό σύμβολο να δέχεται παραπάνω από 2 ορίσματα (έστω ν). Η περίπτωση αυτή δεν αλλάζει τη διαδικασία που παρουσιάστηκε παραπάνω, αφού για το συγκεκριμένο κόμβο θα δημιουργηθούν όσα παιδιά χρειάζονται (εδώ ν παιδιά).

4.1.1 Μέθοδος Εξέλιξης (grow)

Η μέθοδος εξέλιξης παράγει δέντρα με ακανόνιστο σχήμα-μέγεθος. Οι κόμβοι επιλέγονται τυχαία από τα σύνολα των συναρτήσεων και των τερματικών συμβόλων σε όλο το δέντρο, εκτός από τον κόμβο ρίζα ο οποίος δεν είναι τερματικό σύμβολο. Επιλέγεται η ρίζα να μην είναι τερματικό σύμβολο, ώστε να αποφευχθούν περιπτώσεις όπου προκύπτουν εκφυλισμένα δέντρα. Όταν ένα κλαδί περιέχει ένα τερματικό σύμβολο, το κλαδί αυτό έχει ολοκληρωθεί, ακόμα και αν δεν έχουμε φτάσει το μέγιστο επιτρεπτό βάθος. Ο λόγος που προκύπτουν ακανόνιστα δέντρα, είναι πως η επιλογή των τερματικών συμβόλων είναι τυχαία με αποτέλεσμα τέτοιες επιλογές να οδηγήσουν σε τερματισμό της ανάπτυξης κάποιου κλαδιού.

```
InitialiseG(node,depth)
switch depth do
        case 1
                Ομοιόμορφη τυχαία επιλογή κόμβου από το σύνολο F;
        case maxdepth
                Ομοιόμορφη τυχαία επιλογή κόμβου από το σύνολο T;
        case otherwise
                Ομοιόμορφη τυχαία επιλογή κόμβου από το σύνολο F ∪ T ;
end
if node type ∈ F then
        for n = 1 to arity of node type do
                InitialiseG(child n,depth+1);
        end
else
        return;
```

4.1.2 Μέθοδος Πλήρης (full)

Στην περίπτωση της πλήρους μεθόδου, αντί να επιλέγονται τυχαία κόμβοι από το σύνολο των συναρτήσεων και των τερματικών συμβόλων, επιλέγονται συνέχεια συναρτήσεις μέχρι κάποιος κόμβος να φτάσει το μέγιστο βάθος. Τότε, η ανάπτυξη ολοκληρώνεται με την επιλογή τερματικών συμβόλων. Το αποτέλεσμα είναι κάθε κλαδί να φτάνει στο μέγιστο δυνατό βάθος. Αν το πλήθος των κόμβων χρησιμοποιείται ως μονάδα μεγέθους για την ανάπτυξη του δέντρου, το δέντρο σταματά να μεγαλώνει όταν φτάσει το προεπιλεγμένο μέγεθος.

```
InitialiseF(node,depth)
if depth < maxdepth then
        Ομοιόμορφη τυχαία επιλογή κόμβου από το σύνολο F;
        for n=1 to arity of node type do
                InitialiseF(child n,depth+1);
        end
else
        Ομοιόμορφη τυχαία επιλογή κόμβου από το σύνολο T;
        return;
end
```

4.1.3 Μέθοδος ramped half-and-half

Στους πληθυσμούς του γενετικού προγραμματισμού είναι πολύ σημαντική η ποικιλία. Η πλήρης μέθοδος μπορεί να παράγει ομοιόμορφες δεντρικές δομές, στον αρχικό πληθυσμό, αφού η διαδικασία που εφαρμόζεται είναι η ίδια για όλα τα άτομα. Για να αποφευχθεί κάτι τέτοιο, επινοήθηκε η μέθοδος ramped half-and-half, με σκοπό να εμπλουτίσει την ποικιλία στους πληθυσμούς. Η μέθοδος αυτή παράγει ένα εύρος από δέντρα με διάφορα μεγέθη και σχήματα. Η ιδέα είναι να συνδυάσει τις δύο παραπάνω μεθόδους. Αν θεωρήσουμε πως το μέγιστο βάθος είναι x. Τότε, ο πληθυσμός χωρίζεται ομοιόμορφα σε άτομα που έχουν βάθος x-1, x-2, x-3, x-4, κτλ. Για κάθε ομάδα με διαφορετικό βάθος, τα μισά δέντρα αρχικοποιούνται με την πλήρη μέθοδο και τα μισά με τη μέθοδο εξέλιξης. Στα δέντρα που δημιουργούνται με την πλήρη μέθοδο, τα μονοπάτια από τη ρίζα του δέντρου μέχρι τα φύλλα έχουν το ίδιο μέγεθος αλλά και το ίδιο σχήμα. Αντίθετα, στα δέντρα που έχουν δημιουργηθεί με τη μέθοδο εξέλιξης, τα δέντρα διαφέρουν στο σχήμα μεταξύ τους.

Data: maxdepth — μέγιστο βάθος ατόμου στον πληθυσμό, ≥ 2
Data: N — μέγεθος πληθυσμού, άρτιο πολλαπλάσιο του maxdepth
for i=2 to maxdepth **do**
    for 1 **to** N/(2*(maxdepth-1)) **do**
        Add *InitialiseF*(root,i) to population;
        Add *InitialiseG*(root,i) to population;
    end
end

4.2 Δεντρική δομή και εκτέλεση προγράμματος

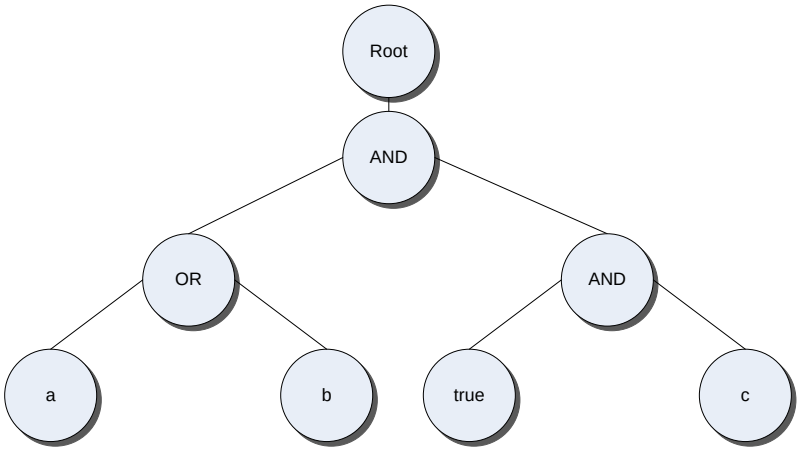
Τα βασικά συστατικά ενός γενετικού προγράμματος (συναρτήσεις και τερματικά σύμβολα) σε καμία περίπτωση δεν αποτελούν προγράμματα. Πρέπει να συνενωθούν σε μια δομή πριν τα χειριστούμε ως προγράμματα.

Τα προγράμματα είναι δομές από συναρτήσεις και τερματικά σύμβολα μαζί με κανόνες και συμβάσεις για το πώς κάθε τερματικό σύμβολο και συνάρτηση πρόκειται να εκτελεστεί.

Η δομή του προγράμματος ενός γενετικού προγράμματος επηρεάζει τη σειρά εκτέλεσης αλλά και τον τρόπο εφαρμογή των τελεστών στο πρόγραμμα. Οι βασικές δομές για την αναπαράσταση γενετικών προγραμμάτων είναι δεντρική δομή, γραμμική δομή και δομή γράφου. Στη δικιά μας περίπτωση έχει επιλεγεί η δεντρική δομή αναπαράστασης. Στη δεντρική δομή υπάρχει μια σύμβαση για το πως εκτελείται η δεντρική δομή.

Η συνήθης σύμβαση για την εκτέλεση δεντρικής δομής είναι ότι εκτελούνται οι κόμβοι πιο κοντά στη ρίζα και στη συνέχεια εκτελούνται οι τερματικοί κόμβοι., έχοντας

κατεύθυνση από τα αριστερά προς τα δεξιά. Η σειρά εκτέλεσης είναι προθεματική (prefix) γιατί οι τελεστέοι εμφανίζονται μετά τον τελεστή (+ X Y). Το πλεονέκτημα της προθεματικής σειράς εκτέλεσης είναι πως σε δέντρα που περιέχουν κόμβους τύπου IF/THEN πρώτα θα αποτιμηθεί το κομμάτι της συνθήκης εκτέλεσης του THEN και μετά εφόσον χρειάζεται, θα εκτελεστεί αυτό. Στο παρακάτω δέντρο, η σειρά εκτέλεσης των κόμβων θα είναι από τον κόμβο AND κάτω από τη ρίζα και από αριστερά προς τα δεξιά OR, a, b και στη συνέχεια στο δίπλα υποδέντρο AND, true, c.



Εικόνα 4.1: Μέθοδος αναπαραγωγής

ΚΕΦΑΛΑΙΟ 5
ΣΧΗΜΑΤΑ ΕΠΙΛΟΓΗΣ και ΕΠΑΝΑΕΙΣΑΓΩΓΗΣ

5.1 Σχήματα Επιλογής (Selection)

Ο τελεστής επιλογής είναι ένας από τους πιο βασικούς τελεστές των εξελικτικών αλγορίθμων. Χρησιμοποιείται ώστε να βελτιώσει την μέση ποιότητα του πληθυσμού δίνοντας στα καλύτερα άτομα μεγαλύτερη πιθανότητα να επιλεγούν στην επόμενη γενιά. Ο τελεστής επιλογής κατευθύνει την αναζήτηση σε πιο υποσχόμενες περιοχές στο χώρο αναζήτησης. Στο κεφάλαιο αυτό, θα γίνει αναφορά στα διαφορετικά σχήματα επιλογής που χρησιμοποιήθηκαν στα πλαίσια της εργασίας και στα βασικά χαρακτηριστικά του καθενός. Οι τελεστές αυτοί είναι: επιλογή με πρωτάθλημα (tournament selection), επιλογή με αποκοπή (truncation selection), γραμμική και εκθετική επιλογή με διαβάθμιση (linear και exponential selection ranking) και επιλογή ανάλογα με την τιμή της συνάρτησης καταλληλότητας (fitness proportional selection).

5.1.1 Ανάλογη με τη καταλληλότητα επιλογή

Η επιλογή ανάλογα με την τιμή που προκύπτει από τη συνάρτηση καταλληλότητας είναι η πρώτη μέθοδος επιλογής που προτάθηκε από τον Holland (1975) για τους γενετικούς αλγορίθμους. Η πιθανότητα κάποιο άτομο να επιλεγεί είναι ανάλογο με την τιμή καταλληλότητας που αντιστοιχεί στο άτομο αυτό, δηλαδή

$$p_i = \frac{f_i}{M * N}$$

όπου M είναι η μέση τιμή καταλληλότητας για τον πληθυσμό των N ατόμων. Αυτή η μέθοδος επιλογής, κανονικοποιεί όλες τις τιμές καταλληλότητας στον πληθυσμό. Αυτές οι κανονικοποιημένες τιμές καταλληλότητας μετατρέπονται στις πιθανότητες επιλογής για τα αντίστοιχα άτομα. Οι τιμές αυτές μπορούν να έχουν μετασχηματιστεί πριν την κανονικοποίηση (π.χ. προσαρμοσμένη για τυποποιημένη καταλληλότητα).

Η πολυπλοκότητα για τον υπολογισμό ενός τέτοιου αλγορίθμου είναι $O(N)$. Στη συνέχεια, παρουσιάζεται ο αλγόριθμος αυτός.

Είσοδος: Ο πληθυσμός $\vec{J} = (J_1, \ldots, J_N)$ και οι τιμές της συνάρτησης καταλληλότητας για τα άτομα του πληθυσμού $\vec{f} = (f_1, \ldots, f_N)$.

Έξοδος: Τα άτομα που έχουν επιλεγεί $\vec{J}' = (J_1', \ldots, J_L')$, όπου L είναι το πλήθος των ατόμων που έχουν επιλεγεί (Το L μπορεί να είναι μικρότερο, μεγαλύτερο ή ίσο του N, ανάλογα με το πλήθος των ατόμων που θέλουμε να επιλέξουμε).

$proportional(\vec{J}, \vec{f})$:

$sum_0 \leftarrow 0$
for $i \leftarrow 1$ **to** N **do**
    $sum_i \leftarrow sum_{i-1} + f_i$
end for
for $i \leftarrow 1$ **to** L **do**
    $r \leftarrow random\ 0, sum_N$
    $επιλόγη\ l, έτσι\ ώστε\ sum_{l-1} \le r < sum_l$
    $l \leftarrow υπολογισμός\ θέσης\ r$
    $J_i' \leftarrow J_l$
end for
return $\vec{J}'$

Ο αλγόριθμος αυτός, για να δουλέψει πρέπει όλες οι τιμές fitness να είναι μεγαλύτερες από μηδέν, ενώ οι πιθανότητες επιλογής εξαρτώνται από την κλιμάκωση (scaling) της συνάρτησης καταλληλότητας. Για παράδειγμα, αν ο πληθυσμός αποτελείται από 10 άτομα με τιμές fitness $\vec{f} = (1,2,3,4,5,6,7,8,9,10)$. Η πιθανότητα επιλογής για το καλύτερο άτομο είναι $p_{best} = \frac{10}{10*5,5} \approx 0.1818 = 18,18\%$ (με μέγεθος πληθυσμού $N = 10$ και μέση τιμή καταλληλότητας $M = \frac{1+2+\cdots+10}{10} = 5,5$) και για το χειρότερο άτομο $p_{worst} = \frac{1}{10*5,5} \approx 0.018 = 1,8\%$. Αν η συνάρτηση καταλληλότητας μετατοπιστεί κατά 100, δηλαδή προστεθεί η τιμή 100 σε κάθε τιμή της συνάρτησης fitness, τότε το καλύτερο άτομο είναι $p_{best}' = \frac{110}{10*105,5} \approx 0.1042 = 10,42\%$ και για το χειρότερο $p_{worst} = \frac{101}{10*105,5} \approx 0.0957 = 9,57\%$. Στην περίπτωση αυτή, η πιθανότητες επιλογής του χειρότερου και του καλύτερου ατόμου είναι σχεδόν οι ίδιες. Αυτό συμβαίνει γιατί η επιλογή ανάλογα με τις τιμές της συνάρτησης καταλληλότητας δεν είναι αμετάβλητη στη μετατόπιση. Για να αντιμετωπιστεί αυτό έχουν προταθεί διάφορες εναλλακτικές μέθοδοι κλιμάκωσης όπως γραμμική στατική και δυναμική κλιμάκωση, λογαριθμική κλιμάκωση, σίγμα αποκοπή. Μια εναλλακτική μέθοδος για τη βελτίωση ανάλογης επιλογής είναι η "υπέρ-επιλογή" ενός συγκεκριμένου ποσοστού από τα καλύτερα άτομα π.χ. το 80% των ατόμων θα επιλεγεί από το 20% των καλύτερων ατόμων.

5.1.2 Επιλογή με πρωτάθλημα

Ο μηχανισμός επιλογής πρωταθλήματος είναι δημοφιλής γιατί είναι απλός, γρήγορος και έχει καλά κατανοητές στατιστικές ιδιότητες. Στην περίπτωση αυτής της επιλογής,

επιλέγεται τυχαία μια ομάδα από t άτομα από τον πληθυσμό. Αυτά μπορούν να επιλεγούν από τον πληθυσμό, με ή χωρίς αντικατάσταση. Πρόκειται για ανεξάρτητες επιλογές, οπότε κάποιο άτομο μπορεί να επιλεγεί παραπάνω από μία φορές. Η ομάδα αυτή των t ατόμων, λαμβάνει μέρος σε ένα πρωτάθλημα, όπου ο νικητής αποφασίζεται με βάση την τιμή καταλληλότητας του. Το καλύτερο άτομο, που έχει τη μεγαλύτερη τιμή καταλληλότητας, επιλέγεται συνήθως ντετερμινιστικά. Παρόλα αυτά, περιστασιακά μπορεί να συμβεί και στοχαστική επιλογή. Και στις δύο περιπτώσεις, μόνο ο νικητής προχωρά στην επόμενη γενιά. Η διαδικασία αυτή επαναλαμβάνεται N φορές, μέχρι να σχηματιστεί ο νέος πληθυσμός. Το πλήθος των ατόμων που ανταγωνίζονται στο πρωτάθλημα αποτελούν το μέγεθος του πρωταθλήματος (tournament size). Όσο μεγαλύτερο είναι το μέγεθος του πρωταθλήματος τόσο κατευθύνεται αυτή η μέθοδος ώστε να επιλέξει τα πιο κατάλληλα άτομα. Αντίθετα, αν $n=1$ τότε η μέθοδος επιλέγει άτομα εντελώς τυχαία. Συνήθως, στο πρωτάθλημα λαμβάνουν μέρος δύο άτομα (δυαδικό πρωτάθλημα), αλλά εύκολα η περίπτωση αυτή γενικεύεται και με ομάδες αποτελούμενες από μεγαλύτερο μέγεθος (περισσότερο εκλεκτική επιλογή).

Στη συνέχεια περιγράφεται ο αλγόριθμος του πρωταθλήματος, όπου τα άτομα μπορούν να επιλεγούν παραπάνω από μία φορά, ενώ το κριτήριο για το νικητή είναι η τιμή καταλληλότητας του.

Είσοδος: Ο πληθυσμός $\vec{J}=(J_1,\ldots,J_N)$, οι τιμές της συνάρτησης καταλληλότητας για τα άτομα του πληθυσμού $\vec{f}=(f_1,\ldots,f_N)$ και το πλήθος των ατόμων που συμμετέχουν στο πρωτάθλημα είναι $t\in 1,2,\ldots,N$.

Έξοδος: Τα άτομα που έχουν επιλεγεί $\vec{J}'=(J_1',\ldots,J_L')$, όπου L είναι το πλήθος των ατόμων που έχουν επιλεγεί (Το L μπορεί να είναι μικρότερο, μεγαλύτερο ή ίσο του N, ανάλογα με το πλήθος των ατόμων που θέλουμε να επιλέξουμε).

$tournament(t,\vec{J},\vec{f})$:
    for $i\leftarrow 1$ **to** L **do**
        $J_i' \leftarrow$ *το καλύτερο άτομο από τα t τυχαία επιλεγμένα άτομα από το* $\vec{J}$
    end for
    return $\vec{J}'$

Η επιλογή στην περίπτωση του πρωταθλήματος μπορεί να υλοποιηθεί πολύ αποτελεσματικά, αφού δεν απαιτείται κάποια ταξινόμηση για τα άτομα του πληθυσμού. Η χρονική πολυπλοκότητα για τον υπολογισμό στην περίπτωση αυτή είναι $O(N)$. Η επιλογή με πρωτάθλημα χρησιμοποιείται πολύ συχνά γιατί δεν απαιτεί σύγκριση της

τιμής καταλληλότητας μεταξύ όλων των ατόμων του πληθυσμού, ενώ παρέχει τη δυνατότητα για παραλληλοποίηση του αλγορίθμου.

5.1.3 Επιλογή με αποκοπή

Στην επιλογή με αποκοπή όπου το κατώτατο όριο είναι T, μόνο το ποσοστό T των καλύτερων ατόμων και όλα έχουν την ίδια πιθανότητα αποκοπής. Η μέθοδος αυτή είναι ισοδύναμη με την $(\mu, \lambda) - επιλογή$, όπου έστω πως το μέγεθος του πληθυσμού είναι $\lambda = \kappa\mu$, με κ και μ θετικούς ακεραίους. Τα μ καλύτερα άτομα στον πληθυσμό επιλέγονται. Κάθε άτομο σε αυτή την ομάδα χρησιμοποιείται ώστε να παραχθούν κ νέα άτομα στην επόμενη γενιά. Στη συνέχεια περιγράφεται ο αλγόριθμος.

Είσοδος: Ο πληθυσμός $\vec{J} = (J_1, \ldots, J_N)$, οι τιμές της συνάρτησης καταλληλότητας για τα άτομα του πληθυσμού $\vec{f} = (f_1, \ldots, f_N)$ και το κατώτατο όριο αποκοπής $T \in 0,1$.

Έξοδος: Τα άτομα που έχουν επιλεγεί $\vec{J}' = (J_1', \ldots, J_L')$, όπου L είναι το πλήθος των ατόμων που έχουν επιλεγεί (Το L μπορεί να είναι μικρότερο, μεγαλύτερο ή ίσο του N, ανάλογα με το πλήθος των ατόμων που θέλουμε να επιλέξουμε).

$truncation(T, \vec{J}, \vec{f})$:

$\vec{J}^* \leftarrow$ *ταξινόμηση του πληθυσμού* $\vec{J}$ *με βάση τις τιμές καταλληλότητας* $\vec{f}$*, με το άτομο με τη μικρότερη τιμή καταλληλότητας στην πρώτη θέση*

for $i \leftarrow 1$ **to** L **do**

$r \leftarrow random\ \left[\ 1 - \mathrm{T}\ \ \mathrm{N}, \ldots, \mathrm{N}\right]$

$l \leftarrow$ *υπολογισμός θέσης* r *στο* $\vec{J}^*$

$J_i' \leftarrow J_r^*$

end for

return $\vec{J}'$

Δεδομένου πως για τη μέθοδο αυτή απαιτείται ταξινόμηση του πληθυσμού, η επιλογή αποκοπής έχει χρονική πολυπλοκότητα $O(N \log N)$.

5.1.4 Επιλογή με διαβάθμιση

Όπως αναφέρθηκε και παραπάνω, Ένα από τα προβλήματα της ανάλογης με την τιμή καταλληλότητας επιλογή είναι πως βασίζεται απευθείας στην καταλληλότητας. Οι τιμές καταλληλότητας, που υπολογίζονται, σπάνια αποτελούν ακριβής μετρήσεις για το πόσο καλό είναι στην πραγματικότητα ένα άτομο. Μια άλλη προσέγγιση, η οποία καταπιάνεται με αυτό το πρόβλημα, είναι η ιεράρχηση των ατόμων με βάση τη καταλληλότητας τους και χρήση αυτής της ιεράρχησης για να υπολογιστεί η πιθανότητα επιλογής.

5.1.4.1 Γραμμική επιλογή με διαβάθμιση

Η γραμμική επιλογή με διαβάθμιση παρουσιάστηκε, πρώτα, από τον Baker με σκοπό να αποβάλει τα μειονεκτήματα της ανάλογης επιλογής. Στη γραμμική επιλογή με ιεράρχηση, τα άτομα ταξινομούνται με βάση την τιμή καταλληλότητας τους. Το καλύτερο άτομο παίρνει και τη μεγαλύτερη τάξη (τιμή) N, ενώ το χειρότερο άτομο παίρνει την τιμή 1. Η πιθανότητα επιλογής κάθε ατόμου ορίζεται γραμμικά από την κατάταξη του ατόμου με βάση τον παρακάτω τύπο:

$$p_i = \frac{1}{N} * \left(\eta^- + \eta^+ - \eta^- * \frac{i-1}{N-1} \right) \mu\varepsilon\, i \in 1,\ldots,N \quad [3]$$

Η πιθανότητα επιλογής για το χειρότερο άτομο είναι $\frac{\eta^-}{N}$ και αυτή για το καλύτερο $\frac{\eta^+}{N}$.

Δεδομένου πως το μέγεθος του πληθυσμού παραμένει σταθερό, θα πρέπει να ισχύουν οι συνθήκες $\eta^+ = 2 - \eta^-$ και $\eta^- \geq 0$. Σε κάθε άτομο του πληθυσμού αντιστοιχεί διαφορετική θέση κατάταξης, έτσι ακόμα και αν έχουν δύο άτομα ίδια τιμή καταλληλότητας θα έχουν διαφορετική πιθανότητα επιλογής. Στη συνέχεια, ακολουθεί ο αλγόριθμος για την γραμμική επιλογή με διαβάθμιση. Για τη μέθοδο αυτή απαιτείται ταξινόμηση του πληθυσμού, οπότε και η πολυπλοκότητα του αλγορίθμου επηρεάζεται από την ταξινόμηση και είναι $O(N \log N)$.

Είσοδος: Ο πληθυσμός $\vec{J} = (J_1,\ldots,J_N)$, οι τιμές της συνάρτησης καταλληλότητας για τα άτομα του πληθυσμού $\vec{f} = (f_1,\ldots,f_N)$ και ο βαθμός αναπαραγωγής του χειρότερου ατόμου είναι $\eta^- \in 0,1$.

Έξοδος: Τα άτομα που έχουν επιλεγεί $\vec{J}' = (J_1',\ldots,J_L')$, όπου L είναι το πλήθος των ατόμων που έχουν επιλεγεί (Το L μπορεί να είναι μικρότερο, μεγαλύτερο ή ίσο του N, ανάλογα με το πλήθος των ατόμων που θέλουμε να επιλέξουμε).

$linear(\eta^-, \vec{J}, \vec{f})$:

$\vec{J}^* \leftarrow$ *ταξινόμηση του πληθυσμού* $\vec{J}$ *με βάση τις τιμές καταλληλότητας* $\vec{f}$*, με το άτομο με τη μικρότερη τιμή καταλληλότητας στην πρώτη θέση*

$sum_0 \leftarrow 0$

[3] Η ισοδύναμη σχέση με την οποία υπολογίζεται η πιθανότητα επιλογής για το άτομο στη σειρά i δίνεται από τη σχέση $p_i = \frac{1}{\|P\|}(2 - c + (2c-2)\frac{i-1}{\|P\|-1})$, όπου $\|P\|$ είναι το μέγεθος του πληθυσμού P, και $1 \leq c \leq 2$ είναι μια σταθερά επιλογής, όπου μεγαλύτερες τιμές του c ωθούν το σύστημα στο να επιλέγει μόνο τα καλύτερα άτομα. Το καλύτερο άτομο στον πληθυσμό επιλέγεται με πιθανότητα $\frac{c}{\|P\|}$ και το χειρότερο άτομο με πιθανότητα $\frac{2-c}{\|P\|}$.

for $i \leftarrow 1$ **to** N **do**

$sum_i \leftarrow sum_{i-1} + p_i$

end for

for $i \leftarrow 1$ **to** L **do**

$r \leftarrow random\ 0,1$

$επιλόγη\ l, έτσι\ ώστε\ sum_{l-1} \le r < sum_l$

$J_i' \leftarrow J_l^*$

end for

return $\vec{J}'$

5.1.4.2 Εκθετική επιλογή με διαβάθμιση

Η εκθετική επιλογή με διαβάθμιση διαφέρει από τη γραμμική επιλογή με διαβάθμιση στο γεγονός, πως οι πιθανότητες για ταξινομημένα άτομα έχουν εκθετικά βάρη. Η βάση του εκθέτη είναι η παράμετρος $0 < c < 1$ της μεθόδου. Όσο πιο κοντά στη μονάδα είναι η τιμή του c τόσο μικρότερη είναι η εκθετικότητα της μεθόδου επιλογής. Και σε αυτή της περίπτωση στο καλύτερο άτομο δίνεται η τιμή N, ενώ στο χειρότερο άτομο η τιμή 1. Έτσι, οι πιθανότητες επιλογής των ατόμων δίνεται από τη σχέση:

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}\ με\ i \in 1,\ldots,N$$

Το άθροισμα $\sum_{j=1}^{N} c^{N-j}$ κανονικοποιεί τις πιθανότητες ώστε να ισχύει $\sum_{i=1}^{N} p_i = 1$.

Δεδομένου πως $\sum_{j=1}^{N} c^{N-j} = \frac{c^N - 1}{c - 1}$, η παραπάνω σχέση μπορεί να γραφεί και ως:

$$p_i = \frac{c-1}{c^N - 1} * c^{N-i}\ με\ i \in 1,\ldots,N$$

Ο αλγόριθμος είναι ίδιος με αυτόν στην περίπτωση της γραμμικής, με μόνη διαφορά στον τρόπο υπολογισμού των πιθανοτήτων.

5.2 Σχήματα Επαναεισαγωγής (Reinsertion)

Μετά την ολοκλήρωση της παραγωγής νέων ατόμων, πρέπει να γίνει επιλογή των ατόμων που θα εισαχθούν στον πληθυσμό της επόμενης γενιάς και των ατόμων που θα αντικατασταθούν από απογόνους. Στα άτομα της νέας γενιάς ανήκουν τα άτομα που έχουν επιλεγεί και στα οποία έχουν εφαρμοστεί οι γενετικοί τελεστές. Αν οι απόγονοι που δημιουργούνται είναι λιγότεροι ή περισσότεροι από το μέγεθος του αρχικού πληθυσμού, τότε για να διατηρηθεί ίδιο το μέγεθος του πληθυσμού είτε διατηρούνται άτομα από την προηγούμενη γενιά είτε επιλέγονται μόνο κάποιοι από τους απογόνους για να εισαχθούν στην επόμενη γενιά, αντίστοιχα. Πολιτικές επαναεισαγωγής είναι οι:

Καθαρή Επανεισαγωγή (Pure Reinsertion): Πρόκειται για το πιο απλό σχήμα επαναεισαγωγής. Παράγονται τόσοι απόγονοι όσοι οι γονείς στον πληθυσμό και όλοι οι γονείς αντικαθίστανται από τους απογόνους. Κάθε άτομο επιβιώνει μόνο για μία γενιά και είναι πιθανό πολύ καλά άτομα να αντικατασταθούν χωρίς να προλάβουν να δώσουν καλύτερους απογόνους, με αποτέλεσμα να χαθεί πληροφορία.

Ομοιόμορφη Επανεισαγωγή (Uniform Reinsertion): Όταν επιλέγεται η μέθοδος αυτή παράγονται λιγότεροι απόγονοι από ότι γονείς, ενώ οι γονείς που αντικαθίστανται επιλέγονται ομοιόμορφα. Το πλεονέκτημα της μεθόδου αυτής είναι πως εισάγει ένα ποσοστό τυχαιότητας στην επιλογή, ενώ το βασικό της μειονέκτημα (σε μικρότερο βαθμό από την καθαρή επαναεισαγωγή) είναι πως καλά άτομα μπορεί να αντικατασταθούν από λιγότερο λειτουργικά άτομα.

Ελιτιστική Επανεισαγωγή (Elitist Reinsertion): Στη μέθοδο αυτή παράγονται λιγότεροι απόγονοι από γονείς και σε κάθε γενιά ένας αριθμός από τα λιγότερο κατάλληλα άτομα αντικαθίσταται από τον ίδιο αριθμό καταλληλότερων απογόνων. Έτσι, οι καλύτερες λύσεις δε χάνονται.

Με βάση την καταλληλότητα Επαναεισαγωγή (Fitness-based Reinsertion): Στη μέθοδο αυτή παράγονται περισσότερα άτομα από αυτά που χρειάζονται για την επόμενη γενιά και εισάγονται μόνο τα καλύτερα. Ένα πλεονέκτημα της πολιτικής αυτής είναι πως τα καλύτερα άτομα εισάγονται στην επόμενη γενιά, ενώ ένα μειονέκτημα πως απαιτείται περισσότερος χρόνος για τον υπολογισμό της καταλληλότητας σε κάθε γενιά (αφού παράγονται περισσότερα άτομα).

Μέρος Β΄

Δίκτυα Ομότιμων Κόμβων

ΚΕΦΑΛΑΙΟ 6
ΔΙΚΤΥΑ ΟΜΟΤΙΜΩΝ ΚΟΜΒΩΝ

6.1 Εισαγωγή

Τα δίκτυα ομότιμων κόμβων εισάγουν μια διαφορετική προσέγγιση στον τρόπο επικοινωνίας εφαρμογών και συστημάτων στο δίκτυο, σε σχέση με το κλασικό σχήμα εξυπηρετητή-καταναλωτή (server-client). Η αρχιτεκτονική αυτή χαρακτηρίζεται από την απευθείας επικοινωνία ανάμεσα στους ομότιμους κόμβους (peers), χωρίς να είναι απαραίτητη η ύπαρξη ενός κεντρικού εξυπηρέτη (central server). Η κυρίαρχη χρήση των δικτύων ομότιμων κόμβων είναι στο διαμοιρασμό αρχείων, όπου παρέχεται στο χρήστη η δυνατότητα να εισάγει, να αποκτήσει και να αναζητήσει περιεχόμενο.

Ο όρος peer-to-peer (P2P) αναφέρεται σε ένα σύνολο από συστήματα και εφαρμογές που χρησιμοποιούν κατανεμημένους πόρους για την πραγματοποίηση λειτουργιών χωρίς τη χρήση κεντρικού ελέγχου (centralized control) ή ιεραρχικής οργάνωσης (hierarchical organization) και όπου το λειτουργικό που εκτελείται σε κάθε κόμβο έχει ισοδύναμη λειτουργικότητα.

Το μοντέλο που χρησιμοποιούν τα P2P δίκτυα βασίζεται στη δημιουργία ενός εικονικού δικτύου (overlay network), το οποίο εκμεταλλεύεται την υπάρχουσα υποδομή και με χρήση των δικών του αλγορίθμων επιτυγχάνει επικοινωνία ανάμεσα στις εφαρμογές που συμμετέχουν σε αυτό. Το δίκτυο αυτό δεν έχει σχέση με το φυσικό δίκτυο που συνδέει τους κόμβους μεταξύ τους.

Τα P2P δίκτυα επιδεικνύουν τρία βασικά χαρακτηριστικά: αυτό-οργάνωση (self-organization), συμμετρική επικοινωνία (symmetric communication) και κατανεμημένο έλεγχο (distributed control). Η αυτό-οργάνωση των P2P δικτύων δίνει τη δυνατότητα προσαρμογής στις αλλαγές στη μορφή του δικτύου (αφίξεις, αναχωρήσεις, αποτυχίες κόμβων), η συμμετρική επικοινωνία αναφέρεται στο γεγονός πως κάθε κόμβος έχει την ικανότητα να διαδραματίζει ταυτόχρονα τόσο το ρόλο του εξυπηρέτη (παρέχει υπηρεσίες προς άλλους κόμβους) όσο και το ρόλο του καταναλωτή (ζητά υπηρεσίες από άλλους κόμβους), ενώ σε δεν απαιτείται η ύπαρξη κεντρικού σημείου ελέγχου.

6.1.1 Διαχωρισμός Αρχιτεκτονικών

Οι P2P αρχιτεκτονικές μπορούν να διαχωριστούν ανάλογα με την εξάρτηση από ένα ή περισσότερους εξυπηρετητές (βαθμός συγκέντρωσης - degree of centralization) σε

πλήρως αποκεντρωμένα (purely decentralized), με μερικό κεντρικό έλεγχο (partially centralized) και υβριδικά αποκεντρωμένα (hybrid decentralized).

Στις πλήρως αποκεντρωμένες P2P αρχιτεκτονικές (π.χ. αρχικό Gnutella, Freenet) όλοι οι κόμβοι ενεργούν τόσο ως εξυπηρετητές όσο και ως καταναλωτές, χωρίς να υπάρχει κεντρικός συντονισμός για τις ενέργειες τους. Τέτοιοι κόμβοι χαρακτηρίζονται ως servents (SERVers + cliENTS).

Στα συστήματα με μερικό κεντρικό έλεγχο (π.χ. Kazza, Morpheus, πρόσφατο Gnutella), η λογική δε διαφέρει σε σχέση με τα πλήρως αποκεντρωμένα συστήματα, αλλά κάποιοι κόμβοι έχουν πιο σημαντικό ρόλο σε σχέση με τους υπόλοιπους κόμβους. Οι κόμβοι αυτοί ονομάζονται υπέρ-κόμβοι (Supernodes) και δεικτοδοτούν αρχεία που διαμοιράζονται ανάμεσα σε τοπικούς κόμβους.

Στα υβριδικά αποκεντρωμένα συστήματα (π.χ. Napster), υπάρχει ένας κεντρικός εξυπηρετητής ο οποίος διευκολύνει την αλληλεπίδραση μεταξύ των κόμβων, διατηρώντας καταλόγους για τα διαμοιραζόμενα αρχεία και αναλαμβάνοντας τις αναζητήσεις για κάποιο αρχείο. Τα συστήματα αυτά, δεν είναι ακριβώς p2p συστήματα, αφού μόνο η ανταλλαγή αρχείων γίνεται μεταξύ κόμβων, αλλά περισσότερο συστήματα καταναλωτή-εξυπηρέτη.

Τα P2P συστήματα μπορούν να διαχωριστούν και ως προς την τοπολογία του δικτύου. Τα συστήματα διαφοροποιούνται ως προς το βαθμό όπου σχηματίζουν κάποια δομή ή δημιουργούνται ad-hoc. Η έννοια της δομής αναφέρεται στον τρόπο με τον οποίο είναι μοιρασμένο το περιεχόμενο του δικτύου σε σχέση με την τοπολογία του δικτύου. Δηλαδή, υπάρχει κάποιος τρόπος με τον οποίο γνωρίζουμε σε ποιους κόμβους μπορεί να υπάρχει κάποιο συγκεκριμένο περιεχόμενο ή πρέπει να ψάξουμε τυχαία ολόκληρο το δίκτυο για να το εντοπίσουμε. Τα συστήματα τα χωρίζουμε, με βάση την τοπολογία, σε τρεις κατηγορίες: δομημένα (structured), χαλαρά δομημένα (loosely structured) και αδόμητα (unstructured).

Στα αδόμητα δίκτυα (π.χ. Gnutella), τα δεδομένα είναι κατανεμημένα ανεξάρτητα από το εικονικό δίκτυο. Η αναζήτηση ισοδυναμεί με τυχαία αναζήτηση αφού δε διατηρείται κάποια πληροφορία για ποιοι κόμβοι έχουν ποια δεδομένα. Τα δίκτυα αυτά μπορούν να υποστηρίξουν προσωρινούς κόμβους στο δίκτυο, αλλά παρουσιάζουν προβλήματα επεκτασιμότητας (scalability).

Τα δομημένα δίκτυα (π.χ. Chord, CAN, PAST, Tapestry) δημιουργήθηκαν για να επιλύσουν τα προβλήματα επεκτασιμότητας που παρουσιάζονται στα αδόμητα δίκτυα.

Τα συστήματα αυτά παρέχουν συγκεκριμένη συσχέτιση ανάμεσα στα δεδομένα και την τοποθεσία τους με τη χρήση κατανεμημένων πινάκων δρομολόγησης για τις επερωτήσεις. Το μειονέκτημα αυτών των δικτύων είναι η δυσκολία που εισάγεται στην περίπτωση μεγάλου ρυθμού αφίξεων και αποχωρήσεων κόμβων.

Τα χαλαρά δίκτυα (π.χ. Freenet) είναι ανάμεσα στα δύο. Υπάρχουν πληροφορίες που αφορούν την αναζήτηση δεδομένων, αλλά δεν είναι πλήρεις με αποτέλεσμα κάποιες αναζητήσεις να αποτυγχάνουν.

Στη συνέχεια, θα γίνει αναλυτική αναφορά σε δύο συγγενικά πρωτόκολλα, στο Chord και στο S-Chord, στα οποία θα ενσωματωθεί ο γενετικός μηχανισμός. Πρόκειται για δύο πλήρως αποκεντρωμένες αρχιτεκτονικές οι οποίες χρησιμοποιούν δομημένα δίκτυα.

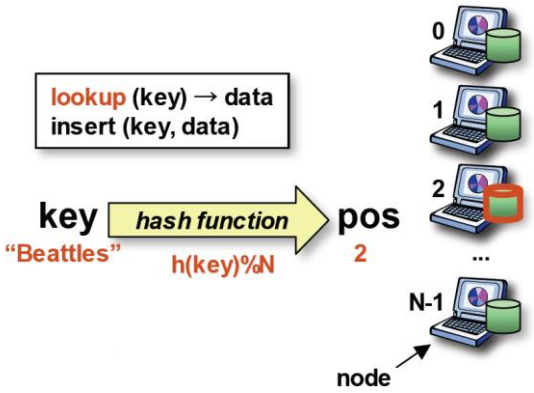
6.1.2 Κατανεμημένοι Πίνακες Κατακερματισμού

Πριν εξετάσουμε λεπτομέρειες στα δύο αυτά πρωτόκολλα, θα αναφερθούμε σε ένα κοινό χαρακτηριστικό και των δύο πρωτοκόλλων, τους κατανεμημένους πίνακες κατακερματισμού.

Η βασική πρόκληση στο σχεδιασμό ενός p2p συστήματος είναι ο αποτελεσματικός εντοπισμός δεδομένων που υπάρχουν στο δίκτυο. Το Chord και το S-Chord χρησιμοποιούν κατανεμημένους πίνακες κατακερματισμού (Distributed Hash Tables - DHT) για τον εντοπισμό δεδομένων στο δίκτυο. Η βασική ιδέα στο μηχανισμό αυτό είναι πως σε συγκεκριμένους κόμβους ανατίθεται ο ρόλος να διατηρούν συγκεκριμένο περιεχόμενο. Έτσι, όταν ένας κόμβος αναζητεί το περιεχόμενο αυτό γνωρίζει σε ποιο κόμβο πρέπει να απευθυνθεί. Ένας τέτοιος μηχανισμός πρέπει να μπορεί να κατανέμει αρμοδιότητες στους κόμβους του δικτύου, να υποστηρίζει τη σύνδεση νέων κόμβων (bootstrap), να είναι αποκεντρωμένος (να μην υπάρχει στενωπός ή σημείο αποτυχίας), καθώς και να χειρίζεται αφίξεις και αναχωρήσεις κόμβων ανακατανέμοντας κατάλληλα τα κλειδιά που επηρεάζονται. Σε έναν κατανεμημένο πίνακα κατακερματισμού, οι κόμβοι επιτελούν τον ρόλο των κουβάδων κατακερματισμού (hash buckets) και κατά την αναζήτηση τα κλειδιά κατακερματίζονται για τον εντοπισμό του κατάλληλου κόμβου.

Στα Chord και S-Chord για το διαμοιρασμό του φόρτου μεταξύ των κόμβων χρησιμοποιείται συνεπής κατακερματισμός (consistent hashing), ιδέα που προέρχεται από το web caching. Ο συνεπής κατακερματισμός ορίζει ένα σταθερό χώρο κατακερματισμού, όπου ανήκουν όλες οι τιμές κατακερματισμού ανεξάρτητα από το πλήθος των κόμβων του δικτύου (κουβάδες κατακερματισμού). Κάθε κλειδί τοποθετείται στον κόμβο που έχει προσδιοριστή πιο κοντά στον δικό του προσδιοριστή στο χώρο

κατακερματισμού, ενώ κάθε κόμβος χρειάζεται να γνωρίζει λίγους γειτονικούς κόμβους, μέσω των οποίων δρομολογούνται τα μηνύματα του στο εικονικό δίκτυο.



Εικόνα 6.1: Κατανεμημένος πίνακας κατακερματισμού

6.2 Πρωτόκολλο Chord

Το Chord ανήκει στη νέα γενιά πρωτοκόλλων, τα οποία υλοποιούν κατανεμημένη αναζήτηση (lookup) μέσω πρωτόκολλων κατανεμημένων πινάκων κατακερματισμού (distributed hash tables). Το Chord παρέχει ένα γρήγορο κατανεμημένο υπολογισμό μιας συνάρτησης κατακερματισμού, η οποία δεδομένου ενός κλειδιού (key), αντιστοιχεί το κλειδί αυτό σε κάποιο κόμβο. Στόχος αυτής της αναζήτησης, είναι ο αποτελεσματικός εντοπισμός ενός κόμβου (node) που έχει αποθηκευμένο ένα συγκεκριμένο πακέτο δεδομένων (data item). Η ανάθεση κλειδιών στους κόμβους πραγματοποιείται με συνεπή κατακερματισμό (consistent hashing), ώστε με μεγάλη πιθανότητα ο φόρτος να είναι ισορροπημένο (όλοι οι κόμβοι έχουν, κατά προσέγγιση, το ίδιο πλήθος κλειδιών) και σε περίπτωση άφιξης (join) ή αποχώρησης (leave) ενός νιοστού κόμβου από το δίκτυο μόνο ένα ποσοστό $O(1/N)$ των κλειδιών να πρέπει να μετακινηθεί ώστε ο φόρτος να παραμείνει ισορροπημένος. Κάθε κόμβος του Chord έχει στη διάθεση του μικρή πληροφορία δρομολόγησης και όποια επιπλέον πληροφορία χρειάζεται για τον υπολογισμό της συνάρτησης κατακερματισμού την αποκτά μέσω επικοινωνίας με άλλους κόμβους. Σε ένα δίκτυο με N κόμβους, κάθε κόμβος διατηρεί πληροφορία για μόνο $O(\log N)$ άλλους κόμβους και μια αναζήτηση απαιτεί ανταλλαγή $O(\log N)$ μηνυμάτων.

Ο χώρος αναζήτησης στο Chord, οργανώνεται σε έναν εικονικό δακτύλιο (Chord ring) όπου κάθε κόμβος και κάθε κλειδί το οποίο συμμετέχει στο Chord, χαρακτηρίζεται από έναν προσδιοριστή (identifier) m-bit ο οποίος προκύπτει από κατακερματισμό με έναν αλγόριθμο όπως ο SHA-1[4]. Οι προσδιοριστές ταξινομούνται σε ένα κύκλο

[4] Οι SHA (Secure Hash Algorithm) συναρτήσεις κατακερματισμού είναι τέσσερις κρυπτογραφικές συναρτήσεις κατακερματισμού (SHA-1, SHA-256, SHA-384, SHA-512), οι οποίοι χρησιμοποιούνται για τη

προσδιοριστών modulo 2^m. Ο προσδιοριστής ενός κόμβου προκύπτει από τον κατακερματισμό του ζευγαριού IP διεύθυνση και θύρα, ενώ αντίστοιχα παράγεται και ο προσδιοριστής για κάθε κλειδί πακέτου δεδομένων. Το μέγεθος m του προσδιοριστή πρέπει να έχει τέτοιο μέγεθος ώστε η πιθανότητα δύο κόμβοι ή δύο κλειδιά να αντιστοιχούν στον ίδιο προσδιοριστή να είναι αμελητέα.

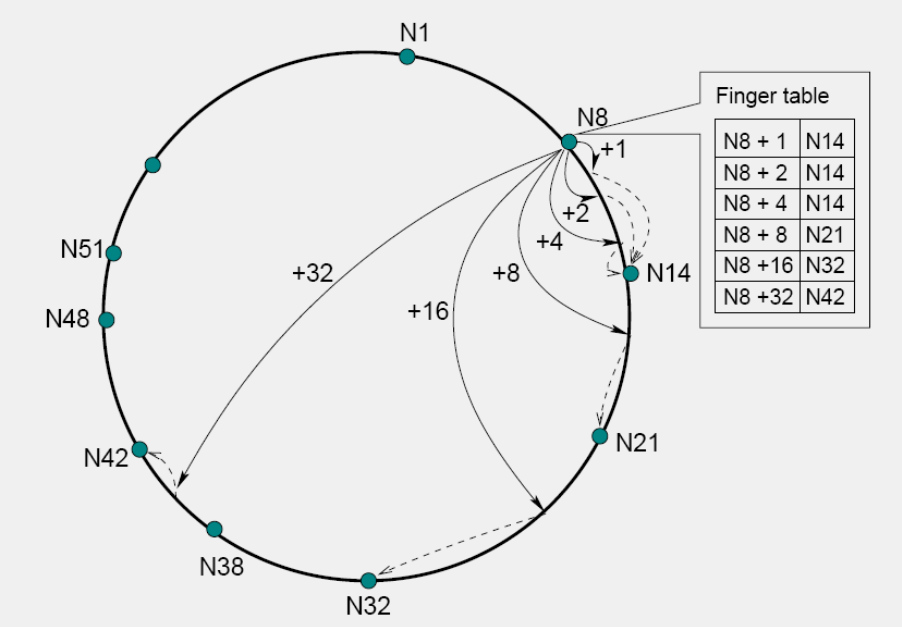
Κάθε κόμβος έχει έναν πρόγονο (predecessor) και έναν απόγονο (successor), οι οποίοι αντιστοιχούν σε αναφορές στον προηγούμενο και στον επόμενο κόμβο στον κυκλικό χώρο αναζήτησης. Για τον καταμερισμό των πακέτων δεδομένων στους κόμβους, στο Chord, αρχικά οι κόμβοι διατάσσονται με βάση τον προσδιοριστή τους στο δακτύλιο. Στη συνέχεια, το κλειδί ενός πακέτου δεδομένων ανατίθεται στον πρώτο κόμβο που έχει προσδιοριστή ίσο ή μεγαλύτερο από αυτόν του κλειδιού στον κυκλικό χώρο αναζήτησης.

Για την αναζήτηση ενός κλειδιού κάθε κόμβος αρκεί να γνωρίζει τον επόμενο κόμβο (successor) στον κύκλο κόμβων. Οι επερωτήσεις (queries) για κάποιον προσδιοριστή (identifier) διαβιβάζονται γύρω στον κύκλο μέσω του δείκτη κάθε κόμβου για τον επόμενο κόμβο, μέχρι να βρεθεί ένας κόμβος του οποίου ο προσδιοριστής να ξεπερνά ή είναι ίσος με τον προσδιοριστή που αναζητούμε. Η γνώση του επόμενου κόμβου αρκεί για να επιλυθούν οι επερωτήσεις, αλλά κάτι τέτοιο δεν είναι πολύ αποδοτικό, αφού μπορεί να χρειαστεί να διασχίσουμε όλους τους κόμβους του δικτύου.

Για να επιταχυνθεί αυτή η διαδικασία, το Chord, διατηρεί επιπλέον πληροφορίες δρομολόγησης. Κάθε κόμβος, n, έχει στη διάθεση του έναν πίνακα δρομολόγησης/ δεικτών (routing/finger table), ο οποίος εκτός από τη διεύθυνση του προηγούμενου και του επόμενου κόμβου έχει και μια λίστα με δείκτες προς επόμενους κόμβους. Αν θεωρήσουμε ότι m είναι το πλήθος bits στους προσδιοριστές κλειδιών/κόμβων (keys/nodes) και πως ο χώρος αναζήτησης έχει μέγεθος N^m, τότε κάθε κόμβος στο Chord αποθηκεύει το πολύ m δείκτες προς γειτονικούς κόμβους στον πίνακα δεικτών. Η i-οστή εγγραφή του κόμβου n περιέχει τον προσδιοριστή του πρώτου κόμβου, s, ο οποίος ξεπερνά τον κόμβο n το λιγότερο κατά 2^{i-1} στον κύκλο των προσδιοριστών, δηλαδή $s = successor(n + 2^{i-1})$, όπου $1 \le i \le m$ (σε αριθμητική modulo 2^m). Ο πρώτος δείκτης στον πίνακα δεικτών του κόμβου n, είναι ο αμέσως επόμενος κόμβος (successor) του στον κύκλο των προσδιοριστών. Στην παρακάτω Εικόνα απεικονίζεται ένα κυκλικό δίκτυο Chord. Για τον κόμβο με προσδιοριστή 8, απεικονίζεται και ο

γέννηση και πιστοποίηση ψηφιακών υπογραφών (digital signatures) και κωδικών γνησιότητας, καθώς και για την παραγωγή τυχαίων αριθμών.

πίνακας δεικτών. Ο πρώτος δείκτης για τον κόμβο 8 αναφέρεται στον κόμβο 14, ο οποίος είναι ο πρώτος κόμβος που ακολουθεί τον κόμβο $8+2^0 \mod 2^6 = 9$. Αντίστοιχα, ο τελευταίος δείκτης του κόμβου 8 είναι ο κόμβος 42, αφού ο κόμβος 42 είναι ο πρώτος που ακολουθεί τον κόμβο $8+2^5 \mod 2^6 = 40$.



Εικόνα 6.2: Δακτύλιος Chord

Εκτός από τον πίνακα δεικτών, πολλές φορές για να είναι πιο εύρωστο (robust) το δίκτυο κάθε κόμβος διατηρεί και μια λίστα τους πρώτους r επόμενους κόμβους (successor list). Σε περίπτωση αποτυχίας του επόμενου κόμβου, αντικαθίσταται ο κόμβος που αποτυγχάνει με τον επόμενο του από τη λίστα επόμενων.

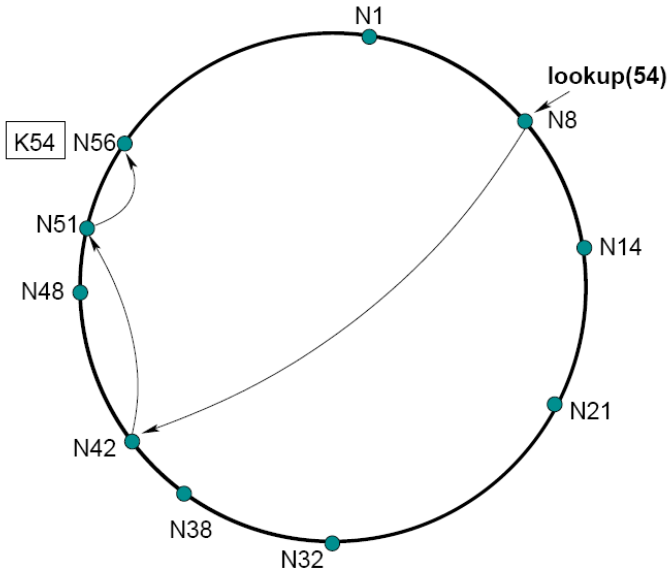
Στο Chord, κάθε κόμβος αποθηκεύει πληροφορίες μόνο για ένα μικρό αριθμό από κόμβους και γνωρίζει περισσότερα για κόμβους που είναι κοντά του στον κύκλο των προσδιοριστών από τους κόμβους που είναι μακριά του. Οι πληροφορίες που περιέχονται στον πίνακα δεικτών ενός κόμβου, δεν επαρκούν πάντα για να εντοπιστούν όλα τα κλειδιά. Στην περίπτωση αυτή, ο κόμβος n μεταβιβάζει την επερώτηση σε κάποιο κόμβο που είναι πιο κοντά στο κλειδί που αναζητούμε. Η διαδικασία αυτή επαναλαμβάνεται και σε κάθε βήμα μειώνεται στο μισό η απόσταση προς τον προσδιοριστή που αναζητούμε. Έτσι, με μεγάλη πιθανότητα, το πλήθος των κόμβων που πρέπει να διασχιστεί για να βρεθεί ένας επόμενος (successor) σε ένα δίκτυο N κόμβων είναι $O(\log N)$. Επιπλέον, σε ένα δίκτυο N κόμβων με K κλειδιά, με μεγάλη πιθανότητα, κάθε κόμβος είναι υπεύθυνος για το πολύ $1+\varepsilon\ K/N$ κλειδιά (όπου

$\varepsilon = O(\log N)$), ενώ σε περίπτωση άφιξης ή αποχώρησης κόμβου $O(K/N)$ κλειδιά πρέπει να μετακινηθούν (μόνο προς ή από τον κόμβο που έρχεται ή φεύγει αντίστοιχα).

Στη συνέχεια γίνεται αναφορά στις βασικές πράξεις που υποστηρίζει το Chord, δίνοντας έμφαση στην αναζήτηση κλειδιού.

***Εισαγωγή (insert)*:** Η λειτουργία αυτή χρησιμοποιείται για την αποθήκευση ενός ζευγαριού κλειδί-δεδομένα (key-data). Υπάρχει η δυνατότητα το ζευγάρι αυτό να αποθηκευτεί σε περισσότερους από έναν κόμβους ώστε να υπάρχει μεγαλύτερη διαθεσιμότητα.

***Αναζήτηση (lookup)*:** Η λειτουργία αυτή χρησιμοποιείται για τον εντοπισμό του ζευγαριού κλειδί-δεδομένα (key-data) που αποθηκεύτηκε με την εισαγωγή και επιστρέφει τα δεδομένα που αντιστοιχούν στο κλειδί. Στη διαδικασία της αναζήτησης χρησιμοποιείται ο πίνακας δεικτών που αναφέρθηκε παραπάνω. Αρχικά, αν ο τρέχων κόμβος έχει το κλειδί τότε απλά το επιστρέφει. Διαφορετικά, βρίσκει στον πίνακα δεικτών που διατηρεί τον κόμβο που είναι πιο κοντά και έχει τιμή μικρότερη από το κλειδί που αναζητά και προωθεί σε αυτόν την επερώτηση. Η επερώτηση προωθείται αναδρομικά σε κόμβους που είναι σε κάθε βήμα πιο κοντά στο κλειδί που αναζητούμε μέχρι να βρεθεί ο κατάλληλος κόμβος. Το πλήθος των μηνυμάτων που ανταλλάσσονται είναι $O(\log N)$[5]. Στην συνέχεια ακολουθεί ένα παράδειγμα αναζήτηση.



Εικόνα 6.3: Δακτύλιος αναζήτησης

[5] Η τιμή αυτή ενδέχεται να αυξηθεί, εφόσον ο πίνακας δεικτών ή η λίστα επόμενων είναι ανακριβείς (π.χ. λόγω αλλαγών στη δομή του δικτύου από αποχωρήσεις ή αφίξεις νέων κόμβων)

Στο παραπάνω, Chord δίκτυο, ξεκινάει αναζήτηση από τον κόμβο 8, για τον κόμβο που έχει αποθηκευμένο το κλειδί με τιμή 54. Ο κόμβος 8, δεν έχει αποθηκευμένο το κλειδί 54, γι' αυτό συνεχίζει την αναζήτηση στον πίνακα δεικτών του. Ο μεγαλύτερος δείκτης που προηγείται του 54 είναι ο 42. Άρα, ο κόμβος 8 μεταβιβάζει την επερώτηση στον κόμβο 42. Ο κόμβος 42 χρησιμοποιώντας τον πίνακα δεικτών του μεταβιβάζει την επερώτηση στον κόμβο 51. Ο κόμβος 51 εντοπίζει πως ο επόμενος κόμβος του, ξεπερνά το κλειδί 54 και επιστρέφει τον κόμβο 56 ως τον κόμβο που έχει το κλειδί.

***Ενημέρωση (update)*:** Ενημέρωση του ζευγαριού κλειδί-δεδομένα (key-data), αλλά μόνο από το δημιουργό του συγκεκριμένου κλειδιού.

***Δημιουργία (create)*:** Καλείται για τη δημιουργία ενός νέου δικτύου Chord.

Εγγραφή (join): Χρησιμοποιείται όταν ένας καινούργιος κόμβος εισάγεται στο σύστημα. Αρχικά, ο καινούργιος κόμβος συνδέεται με έναν υπάρχοντα κόμβο του δικτύου, ο οποίος τον ενημερώνει για το ποιος κόμβος είναι ο επόμενός του. Στη συνέχεια, με τη βοήθεια της μεθόδου stabilize, που εκτελούν περιοδικά οι κόμβοι, θα ενημερωθούν και οι υπόλοιποι ενδιαφερόμενοι κόμβοι για την ύπαρξη του νέου κόμβου και με τη μέθοδο fix_fingers αρχικοποιείται ο πίνακας δεικτών.

Επιλογή Πίνακα Δεικτών (fix_fingers): Η μέθοδος αυτή καλείται περιοδικά από κάθε κόμβο ώστε να ελεγχθεί πως οι εγγραφές του πίνακα δεικτών είναι ορθές. Με τη μέθοδο αυτή αρχικοποιείται ο πίνακας δεικτών για κάθε νέο κόμβο, ενώ παρέχει τη δυνατότητα για ενσωμάτωση νέων κόμβων στον πίνακα δεικτών.

***Αποχώρηση (leave)*:** Η διαδικασία αυτή ενεργοποιείται όταν ένας κόμβος αποχωρεί από το δίκτυο. Στην περίπτωση αυτή, μεταφέρει τα κλειδιά του στον επόμενο κόμβο και ενημερώνει τον προηγούμενο για τον νέο επόμενο. Σε περίπτωση αποτυχίας ενός κόμβου, η αλλαγή που προέκυψε θα διαδοθεί στο δίκτυο μέσα από τις περιοδικές κλήσεις της σταθεροποίησης από τους κόμβους.

***Σταθεροποίηση (stabilize)*:** Με τη μέθοδο αυτή εκτελείται περιοδικά και ενημερώνει τους κόμβους για νέους κόμβους που εισάγονται στο δίκτυο. Όταν ο κόμβος n καλεί τη μέθοδο σταθεροποίησης, ζητά από τον επόμενό του, κόμβο s, να τον ενημερώσει για τον προηγούμενο κόμβο p του s και ελέγχει κατά πόσο ο p θα πρέπει να είναι ο επόμενος του (κάτι τέτοιο μπορεί να συμβαίνει εφόσον ο p έχει εισαχθεί πρόσφατα στο δίκτυο). Ακόμα, ενημερώνει τον κόμβο s για την ύπαρξη n, δίνοντας τη δυνατότητα στον s να αλλάξει τον προηγούμενο του σε n, αν ο n είναι ο πιο κοντινός προηγούμενος. Η

μέθοδος αυτή βοηθά στην ορθότητα των πινάκων δρομολόγησης, και συνεπώς, και στα αποτελέσματα στις αναζητήσεις.

6.3 Πρωτόκολλο S-Chord

Το Chord, στο οποίο αναφερθήκαμε στην προηγούμενη παράγραφο, είναι ένα αντιπροσωπευτικό πρωτόκολλο που επιλύει το πρόβλημα του εντοπισμού δεδομένων, αποδοτικά, σε ένα δίκτυο ομότιμων κόμβων. Η δρομολόγηση, όμως, είναι οργανωμένη ασύμμετρα με αποτέλεσμα την αδυναμία έγκαιρης ενημερώσεων για αλλαγές στη δρομολόγηση, υποστήριξης συμμετρικών εφαρμογών και αποτελεσματικής εκμετάλλευσης της εγγύτητας στο δίκτυο. Μια λύση στους περιορισμούς αυτούς προσφέρει μια επέκταση του Chord, το S-Chord (Symmetric Chord), στο οποίο οι πίνακες δρομολόγησης οργανώνονται συμμετρικά και ο κυκλικός χώρος αναζήτησης μπορεί να προσπελαστεί και προς τις δύο κατευθύνσεις (πολιτική δρομολόγησης). Στη συνέχεια θα παρουσιαστούν τα βασικά χαρακτηριστικά του S-Chord και πως αυτό διαφοροποιείται από τον κλασικό αλγόριθμο του Chord, δίνοντας έμφαση στις διαφορές στους πίνακες δεικτών.

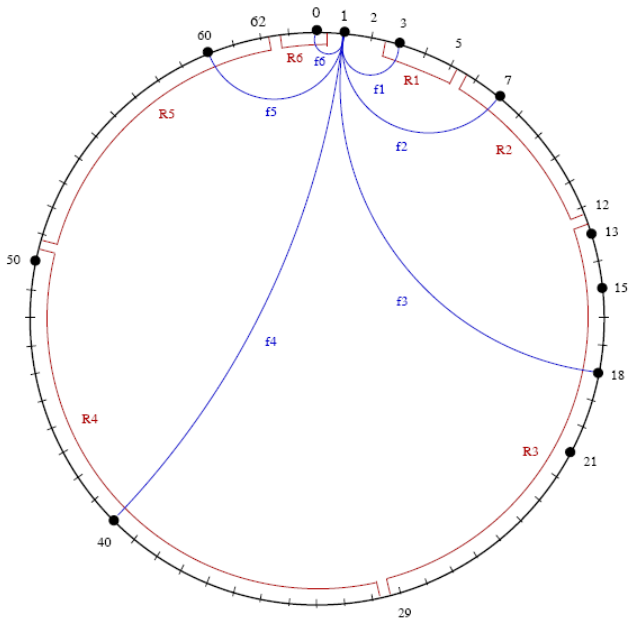
Στο S-Chord, η συμμετρία είναι βασικός άξονας. Υπάρχει συμμετρία στις εγγραφές δρομολόγησης (routing entry), στο κόστος δρομολόγησης (routing cost) και στον πίνακα δεικτών (finger table). Η συμμετρία στις εγγραφές δρομολόγησης αναφέρεται στο χαρακτηριστικό πως για δύο διαφορετικούς κόμβους, n και p, αν ο p έχει ένα δείκτη στο n, τότε και ο n έχει ένα δείκτη προς τον p. Επιπλέον, σε αντίθεση με το Chord, όπου το εικονικό κυκλικό δίκτυο μπορεί να προσπελαστεί μόνο προς τη μία κατεύθυνση, στο S-Chord η συμμετρία στις εγγραφές επιτρέπει τη διάσχιση του δικτύου και προς τις δύο κατευθύνσεις. Η συμμετρία στις εγγραφές σε συνδυασμό με τον αλγόριθμο αναζήτησης επιτρέπει τη συμμετρία στο κόστος δρομολόγησης (είναι πολύ πιθανό το μήκος του μονοπατιού αναζήτησης ανάμεσα σε δύο κόμβους να είναι ίσο). Τέλος, οι πίνακες δρομολόγησης ενός κόμβου n είναι οργανωμένοι, συμμετρικά, με βάση την απόσταση του n από το $n \oplus \frac{N}{2}$, παρέχοντας γρήγορη πρόσβαση σε ολόκληρο το χώρο αναζήτησης.

Οι πίνακες δρομολόγησης στο S-Chord, έχουν το ίδιο μέγεθος με αυτούς του Chord ($\lceil \log_2 N \rceil$, όπου N το μέγεθος του χώρου αναζήτησης), αλλά διαφέρουν στον τρόπο που είναι οργανωμένοι. Ο πίνακας δρομολόγησης κάθε κόμβου έχει μέγεθος $2*m$, όπου $m = \lceil \log_4 N \rceil$. Ο δείκτης i για τον κόμβο n, ορίζεται ως εξής:

$$n.f\ i = \begin{Bmatrix} successor^{+}\ \ n \oplus 4^{i-1} & i \in 1, m \\ successor^{-}\ \ n \ominus 4^{2m-i} & i \in m+1, 2m \end{Bmatrix}$$

Ο i-οστός δείκτης που ανήκει στο διάστημα $1, m$, αναφέρεται στον προσδιοριστή του πρώτου κόμβου που ξεπερνά τον n το λιγότερο κατά 4^{i-1}, με φορά όπως αυτή των δεικτών του ρολογιού, ενώ ο i-οστός δείκτης που ανήκει στο διάστημα $m+1, 2m$, αναφέρεται στον προσδιοριστή του πρώτου κόμβου που ξεπερνά τον n το λιγότερο κατά 4^{2m-i}, με φορά αντίθετη των δεικτών του ρολογιού.

Στο παρακάτω σχήμα, παρουσιάζεται ένα δίκτυο που αποτελείται από 11 κόμβους, επιλεγμένους από ένα χώρο αναζήτησης μεγέθους $N = 64$. Ο πρώτος δείκτης για τον κόμβο 1, δείχνει στον κόμβο 3, που είναι ο πρώτος κόμβος που ξεπερνά τον κόμβο 1 κατά $1 \oplus 4^0 = 2$. Αντίστοιχα, ο κόμβος 40 είναι ο πρώτος κόμβος που ξεπερνά το $1 \ominus 4^2 = 49$ με φορά αντίθετη των δεικτών του ρολογιού.



Εικόνα 6.4: Δακτύλιος S-Chord

Για κάθε δείκτη του πίνακα δεικτών υπάρχει ένα διάστημα ευθύνης (responsibility interval), το οποίο ορίζει το εύρος κλειδιών που αναμένεται να βρεθούν με ένα ελάχιστο αριθμό βημάτων μέσω αυτού του δείκτη, ξεκινώντας από τον κόμβο n. Τα διαστήματα ευθύνης χρησιμοποιούνται κατά τη δρομολόγηση, όπου επερωτήσεις για κάποιο κλειδί k κατευθύνονται στο διάστημα ευθύνης που περιέχει το συγκεκριμένο κλειδί. Στο Chord ένας δείκτης βρίσκεται στην αρχή του διαστήματος του χώρου αναζήτησης για το οποίο είναι υπεύθυνος, ενώ στο S-Chord βρίσκεται μέσα στο διάστημα. Η περιοχή ευθύνης

του i-οστού δείκτη ενός κόμβου ξεκινά από τη μισή απόσταση ανάμεσα σε αυτόν και στον (i-1)-οστό δείκτη και τελειώνει στη μισή απόσταση ανάμεσα σε αυτό και στον (i+1)-οστό δείκτη. Η περιοχή ευθύνης για τον i-οστό δείκτη του κόμβου n ορίζεται από την παρακάτω σχέση:

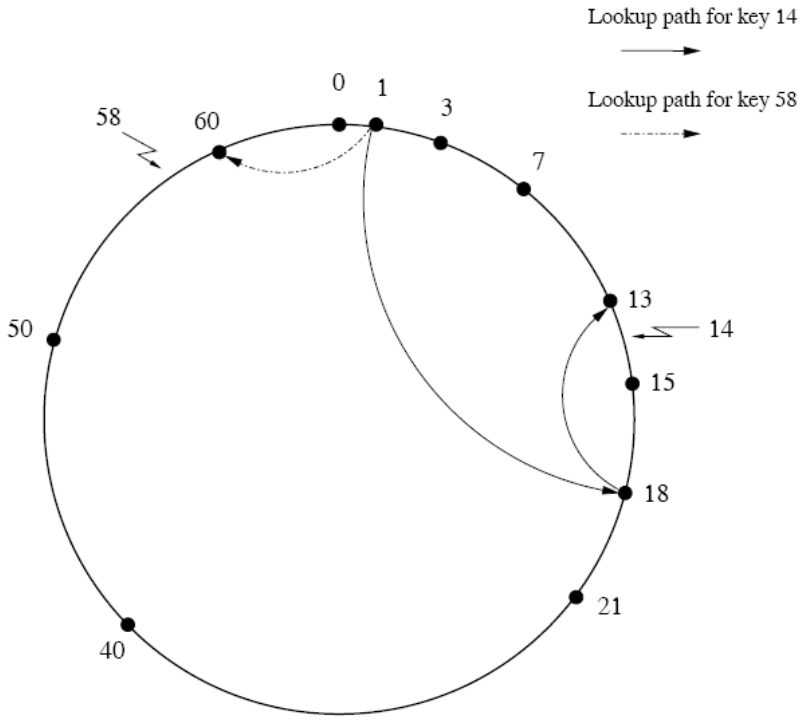
$$R_n\ i \coloneqq \left[n.f\ \ i-1 \oplus \left\lfloor \frac{n.f\ \ i \ominus n.f[i-1]}{2} \right\rfloor \rightarrow n.f\ \ i \oplus \left\lfloor \frac{n.f\ \ i+1 \ominus n.f[i]}{2} \right\rfloor \right], i \in [1, 2m]$$

όπου $n.f\ \ 0 = n.f\ \ 2m+1 = n$ για κάθε κόμβο $0 < n < N$, όπου για $n = 0$ έχουμε $n.f\ \ 0 = 0$ και $n.f\ \ 2m+1 = N$. Για παράδειγμα, οι περιοχές ευθύνης για τους δείκτες 1, 2 και 4 για τον κόμβο 1 είναι $R_1\ 1 = 1 \rightarrow 5$, $R_1\ 2 = 5 \rightarrow 12$ και $R_1\ 4 = 29 \rightarrow 50$, αντίστοιχα.

Στο σημείο αυτό θα αναφερθούμε στο πως υλοποιείται ο αλγόριθμος αναζήτησης στο S-Chord. Τα κλειδιά, όπως και στο Chord, αποθηκεύονται στον πρώτο κόμβο με προσδιοριστή (id) μεγαλύτερο ή ίσο με τον προσδιοριστή του κλειδιού. Η αναζήτηση περιορίζεται στην εύρεση του επόμενου κόμβου του κλειδιού. Αρχικά, ελέγχεται αν το κλειδί ανήκει στο διάστημα ανάμεσα στον κόμβο n και τον επόμενο του s (successor) ή στον προηγούμενο του p (predecessor) και στον κόμβο n. Οπότε επιστρέφεται ο επόμενος s στην μία περίπτωση και ο ίδιος ο n στην άλλη. Σε διαφορετική περίπτωση, πραγματοποιείται αναζήτηση στον πίνακα δεικτών του κόμβου και επιλέγεται ο πιο κοντινός δείκτης τον οποίο γνωρίζει ο n και σε αυτόν προωθείται η αναζήτηση. Ο πιο κοντινός κόμβος σε κάποιο κλειδί k τον οποίο γνωρίζει ο n, προσδιορίζεται από το δείκτη του πίνακα δεικτών στου οποίου την περιοχή ευθύνης ανήκει το k. Στην περίπτωση του S-Chord μια αναζήτηση μπορεί να δρομολογηθεί και προς τις δύο κατευθύνσεις στον κυκλικό χώρο αναζήτησης, ανάλογα με το που βρίσκεται το κλειδί που αναζητούμε.

Στη συνέχεια ακολουθούν δύο παραδείγματα αναζήτησης κλειδιών. Στην πρώτη περίπτωση θέλουμε να εντοπίσουμε το κλειδί 14 ξεκινώντας από τον κόμβο 1. Για τον κόμβο 1, το κλειδί 14 ανήκει στη δικαιοδοσία του δείκτη 3 στον πίνακα δρομολόγησης, γι' αυτό και η επερώτηση προωθείται στον κόμβο 18. Από τον κόμβο 18, η επερώτηση προωθείται ακολουθώντας ανάποδα τη φορά των δεικτών του ρολογιού στον κόμβο 13 και ο κόμβος 13 θα εντοπίσει πως ο επόμενος του, κόμβος 15, είναι υπεύθυνος για το κλειδί και η τιμή αυτού θα επιστραφεί στον κόμβο 1. Στην περίπτωση του κλειδιού 58, το κλειδί 58 περιλαμβάνεται στην περιοχή ευθύνης του δείκτη 5 του κόμβου 1, οπότε η αναζήτηση προωθείται στον κόμβο 60, ο οποίος έχει και το κλειδί 58.

Το μέγιστο πλήθος βημάτων που χρειάζονται για τον εντοπισμό ενός κλειδιού σε ένα δίκτυο S-Chord μεγέθους N είναι $\left\lceil \frac{3}{4}\log_2 N \right\rceil$, που είναι 25% λιγότερο σε σχέση με το κλασικό Chord.



Εικόνα 6.5: Παράδειγμα αναζήτησης S-Chord

Όσον αφορά τις αφίξεις και αναχωρήσεις κόμβων από το δίκτυο δεν υπάρχουν μεγάλες διαφορές ανάμεσα στο S-Chord και το Chord. Και σε αυτή την περίπτωση η άφιξη ενός κόμβου οδηγεί σε ανανέωση των δεικτών των πινάκων δεικτών των κόμβων όπου είναι επιθυμητό να δείχνουν στον καινούργιο κόμβο. Αντίστοιχα, η απομάκρυνση ενός κόμβου οδηγεί στην κατάργηση των δεικτών προς τον κόμβο αυτό.

Μέρος Γ΄

Εφαρμογή Γενετικού Προγραμματισμού στα Πρωτόκολλα Chord και S-Chord

ΚΕΦΑΛΑΙΟ 7
ΓΕΝΕΤΙΚΗ ΓΛΩΣΣΑ

7.1 Εισαγωγή

Η γενετική γλώσσας είναι μια LISP-like γλώσσα η οποία χρησιμοποιείται για την αναπαράσταση των γενετικών προγραμμάτων. Η γλώσσα αυτή υποστηρίζει αρκετές συναρτήσεις, ενώ μπορεί να υποστηρίζει και διάφορους τύπους δεδομένων. Ακόμα, τόσο οι πράξεις όσο και οι τύποι δεδομένων που υποστηρίζει είναι δυνατόν να εμπλουτιστούν με νέους. Η γλώσσα αυτή σχεδιάστηκε ώστε να μπορεί να εκφράσει πολύπλοκα γενετικά προγράμματα, με δομές ελέγχου, δομές μετάβασης και συναρτήσεις. Σκοπός της ψευδογλώσσας δεν είναι να πραγματοποιεί μόνο απλές πράξεις με αριθμητικούς τελεστές, αλλά να μπορεί να διατυπώσει κανονικά προγράμματα (π.χ. να μπορεί να πραγματοποιήσει εναλλακτικούς υπολογισμούς ανάλογα με ενδιάμεσα αποτελέσματα, να μπορεί να χειρίζεται διαφορετικούς τύπους μεταβλητών). Ο διερμηνευτής μπορεί να εκτελέσει αυτά τα προγράμματα, ενώ στα προγράμματα αυτά μπορούν να εφαρμοστούν γενετικοί τελεστές. Στη συνέχεια θα γίνει αναφορά στο συντακτικό, τη σημασιολογία, τους τύπους δεδομένων και τα χαρακτηριστικά της γλώσσας. Όπως θα αναφερθεί και στη συνέχεια, η γλώσσα έχει οριστεί ώστε να διαθέτει χαρακτηριστικά που της επιτρέπουν να εκφράσει αλγορίθμους δικτύων, παρέχοντας τόσο τους κατάλληλους τύπους δεδομένων όσο και τις κατάλληλες εντολές (π.χ. εντολή ανταλλαγής μηνυμάτων). Στόχος ήταν με τη βοήθεια αυτών των επιπλέον εντολών και τύπων δεδομένων να εκφραστεί, μια πιο απλοποιημένη έκδοση του πρωτοκόλλου ομότιμων κόμβων (peer-to-peer) Chord.

Κάθε συμβολική έκφραση αποτελείται από στοιχεία της γλώσσας τα οποία θα αναλυθούν στη συνέχεια, και με τη βοήθεια του Αναλυτή (Parser) και του Διερμηνευτή (Interpreter) είναι δυνατόν να αποτιμηθούν σταθερές τιμές και μεταβλητές. Κάθε έκφραση περικλείεται από παρενθέσεις και σε κάθε έκφραση το πρώτο στοιχείο θεωρείται πως αντιστοιχεί στη συνάρτηση στην οποία εφαρμόζονται τα υπόλοιπα στοιχεία που περικλείονται από τις παρενθέσεις. Τα στοιχεία αυτά μπορούν με τη σειρά τους να είναι εκφράσεις οι οποίες πρέπει να αποτιμηθούν, ώστε να εφαρμοστεί σε αυτές ο αρχικός τελεστής.

Μια τέτοια έκφραση είναι η (+ 1 2), στην οποία η συνάρτηση της πρόσθεσης, η οποία εμφανίζεται ως πρώτο στοιχείο στην παρένθεση, λαμβάνει ως ορίσματα δύο στοιχεία τα

1 και 2. Μετά την αποτίμηση της έκφρασης θα προκύψει η τιμή 3. Σε όλες τις εκφράσεις της ψευδογλώσσας χρησιμοποιείται ο πολικός συμβολισμός (Polish notation – prefix notation). Εάν κάποια από τα ορίσματα της συνάρτησης είναι και αυτά εκφράσεις τότε, πρώτα αποτιμώνται οι εκφράσεις αυτές (αναδρομικά, πρώτα κατά βάθος, ξεκινώντας από τον αριστερό τελεστέο). Μια τέτοια περίπτωση είναι η έκφραση (+ (- 8 4) 3), όπου η συνάρτηση της πρόθεσης έχει ως όρισμα την έκφραση (- 8 4) και τη σταθερά 3. Για την αποτίμηση όλης της έκφρασης αρχικά, θα αποτιμηθεί το όρισμα (- 8 4) από όπου θα προκύψει ως αποτέλεσμα 4 και στη συνέχεια θα ληφθεί υπόψη η σταθερά 3, οπότε θα προκύψει και το τελικό αποτέλεσμα 7.

Στη συνέχεια, θα πραγματοποιηθεί μια σύντομη περιγραφή της γλώσσας και των βασικών χαρακτηριστικών της. Στις μονάδες αυτές είναι δυνατόν να προστεθούν επιπλέον μονάδες ανάλογα με την εφαρμογή με μικρές σχετικά τροποποιήσεις στον κώδικα. Κατά την περιγραφή της γλώσσας θα δοθεί BNF (Backus–Naur form) συμβολισμός για κάποια από τα στοιχεία της γραμματικής. Ο συμβολισμός BNF (Backus–Naur form) που δίνεται, δε ξεκαθαρίζει πλήρως τη γραμματική. Υπάρχουν σημεία όπου η γραμματική της γλώσσας είναι διφορούμενη. Τα σημεία αυτά μπορούν να ξεπεραστούν αν λάβουμε υπόψη ποιοι τύποι δεδομένων είναι συμβατοί με ποιους τελεστές, καθώς και τα χαρακτηριστικά της γλώσσας στα οποία αναφερόμαστε στο κείμενο.

7.2 Περιγραφή της Γλώσσας

Σε πρώτο στάδιο θα αναφερθούμε στις λεκτικές μονάδες που υποστηρίζονται από τη γλώσσα.

Οι λέξεις κλειδιά της γλώσσας είναι οι ακόλουθες: *AND, OR, NOT, IF, TRUE, FALSE, GOTO, LABEL, RETURN, SENDMESSAGE, SERIAL, INT, BOOLEAN, ID, ADDRESS, KEY, DATA, DATAKEY, STRING, ADDRESSARRAY, KEYARRAY, KEYDATAARRAY*.

Τα ειδικά σύμβολα της γλώσσας είναι: +, - , *, / , %, ==, <, !=, :=, [,], ., (,), ενώ δεν υποστηρίζονται σύμβολα σχολίων. Τα σύμβολα [,], (και) χρησιμοποιούνται ως διαχωριστές.

Οι λεκτικές μονάδες (tokens) χωρίζονται σε δύο κατηγορίες τα ονόματα και τις αριθμητικές σταθερές. Τα ονόματα, αποτελούνται από γράμματα του λατινικού αλφαβήτου (τα κεφαλαία είναι διαφορετικά γράμματα από τα πεζά), εν δυνάμει ακολουθούμενα από μια σειρά γραμμάτων, ψηφίων ή χαρακτήρων υπογράμμισης (underscore), ενώ δεν πρέπει να συμπίπτουν με κάποια από τις παραπάνω λέξεις

κλειδιά. Οι αριθμητικές σταθερές μπορεί να είναι είτε ακέραιοι είτε δεκαδικοί με ή χωρίς πρόσημο.

$$letter \rightarrow [a-zA-Z]$$
$$digit \rightarrow [0-9]$$
$$identifier \rightarrow letter\ (letter \mid digit \mid '\_')*$$
$$sign \rightarrow + \mid - \mid \varepsilon$$
$$integer \rightarrow sign(0 \mid [1-9]digit*)$$
$$real \rightarrow (integer \mid integer.digit*)\ E\ sign\ digit*$$

Οι σταθερές συμβολοσειρές αποτελούνται από μια ακολουθία χαρακτήρων ή αριθμητικών ψηφίων ή χαρακτήρων διαφυγής (π.χ. \n, \t, \r, \0) και περικλείονται από διπλά εισαγωγικά (π.χ. “abcdef”, “Good Morning!\n”).

$$string \rightarrow "(letter \mid digit \mid escape-char)*"$$
$$escape-char \rightarrow "\ " < ASCII\ character >$$

Στα προγράμματα επιτρέπεται η ύπαρξη κενών, χαρακτήρων νέας γραμμής και χαρακτήρων στηλοθέτησης (tabs), αλλά δεν επιτρέπεται να διαχωρίζουν τις λέξεις κλειδιά και τις λεκτικές μονάδες. Ακολουθίες από τέτοιους χαρακτήρες αγνοούνται κατά την επεξεργασία.

7.2.1 Τύποι δεδομένων

Οι γλώσσα υποστηρίζει αρκετούς τύπους δεδομένων. Εκτός από τους βασικούς τύπους δεδομένων έχουν οριστεί και τύποι δεδομένων που μπορούν να χρησιμοποιηθούν σε δίκτυα ομότιμων κόμβων (p2p). Οι τύποι δεδομένων μπορούν να εμπλουτιστούν σχετικά απλά με τον καθορισμό του τύπου δεδομένων και των πράξεων που επιτρέπονται με τον τύπου αυτό. Οι τύποι δεδομένων που υποστηρίζονται παρουσιάζονται στη συνέχεια:

Πίνακας 7.1: Τύποι Δεδομένων

Σύμβολο	Παρατηρήσεις
INT:	Ακέραιοι αριθμοί μεγέθους 32-bit (-2,147,483,648 έως 2,147,483,647).
DOUBLE:	Πραγματικός αριθμός διπλής ακρίβειας 64-bit IEEE 754 κινητής υποδιαστολής.
BOOLEAN:	Λογικός τύπος δεδομένων, που δέχεται δύο δυνατές τιμές, αληθές (TRUE) και ψευδές (FALSE). Ο τύπος αυτός δεδομένων αναπαριστά πληροφορία ενός bit.
ID:	Προσδιοριστής (Identifier) m-bit κόμβου. Το μέγιστο μέγεθος του σε αυτή την υλοποίηση είναι 32-bit.
ADDRESS:	Διεύθυνση κόμβου, έχει μέγεθος 32-bit και πρόκειται για έναν ακέραιο που προσδιορίζεται από το ID με τη σχέση $100000 + 4 * ID$.

	Έχει βοηθητικό ρόλο και λειτουργεί ως η IP του κάθε κόμβου.
KEY:	Πρόκειται για το κλειδί (Key) το οποίο αντιστοιχεί σε κάποιο δεδομένο. Το κλειδί αυτό κατακερματίζεται για την αναζήτηση του κόμβου που έχει κάποιο δεδομένο. Το μέγεθος του είναι 32-bit.
DATA:	Δεδομένα (Value). Μπορεί να είναι κάποια διεύθυνση, αρχείο ή κάποιο τυχαίο δεδομένο. Αναπαρίσταται με τη βοήθεια μιας ακολουθίας χαρακτήρων.
DATAKEY:	Ζευγάρι Κλειδί/Δεδομένα (Key/Value).
STRING:	Συμβολοσειρά, αντιστοιχεί σε μια ακολουθία χαρακτήρων.
VOID:	Απουσία τύπου, χρησιμοποιείται μόνο ως επιστρεφόμενος τύπος (return type) για τις συναρτήσεις.
NULL:	Χρησιμοποιείται για να δηλώσει σφάλμα και χρησιμοποιείται μόνο ως επιστρεφόμενος τύπος (return type) για τις συναρτήσεις.

Εκτός από τους παραπάνω βασικού τύπους, υπάρχει υποστήριξη για τύπους πινάκων μιας διάστασης. Οι πίνακες που υποστηρίζονται είναι οι πίνακες διευθύνσεων, κλειδιών και ζευγαριών κλειδί/δεδομένα. Η προσπέλαση ενός στοιχείου του πίνακα γίνεται με τη βοήθεια [ ], όπου ο ακέραιος ανάμεσα στις αγκύλες προσδιορίζει τη θέση στον πίνακα (π.χ. a[i] αντιστοιχεί στο στοιχείο i του πίνακα a). Μπορούν να οριστούν και άλλοι τύποι δεδομένων πίνακα από τους βασικούς τύπους δεδομένων. Στη συνέχεια, παρουσιάζονται οι υποστηριζόμενοι τύποι πίνακα.

Πίνακας 7.2: Τύποι πίνακα

Σύμβολο	Παρατηρήσεις
ADDRESSARRAY	Πίνακας από διευθύνσεις κόμβων
KEYARRAY	Πίνακας από κλειδιά
KEYDATAARRAY	Πίνακας από ζευγάρια κλειδί/δεδομένο

$$data-type \rightarrow INT \mid DOUBLE \mid BOOLEAN \mid$$
$$ID \mid ADDRESS \mid KEY \mid DATA \mid$$
$$DATAKEY \mid STRING \mid VOID \mid NULL$$
$$data-array \rightarrow ADDRESSARRAY \mid KEYARRAY \mid KEYDATAARRAY$$

7.2.2 Μεταβλητές

Οι δηλώσεις μεταβλητών γίνεται με την αναφορά πρώτα του τύπου δεδομένων και στη συνέχεια του ονόματος της μεταβλητής. Η δήλωση μεταβλητής περικλείεται και σε αυτή την περίπτωση από παρενθέσεις. Στην περίπτωση δήλωσης πίνακα υπάρχει και ένα τρίτο όρισμα το οποίο αναφέρεται στο μέγεθος του πίνακα, το οποίο πρέπει να είναι θετική σταθερά. Όλες οι μεταβλητές πριν χρησιμοποιηθούν θα πρέπει πρώτα να έχουν δηλωθεί, ενώ δεν υποστηρίζεται η ταυτόχρονη δήλωση πολλών μεταβλητών με τον ίδιο τύπο. Οι μεταβλητές μαζί με τις σταθερές (ακέραιες τιμές, πραγματικές τιμές, συμβολοσειρές κ.τ.λ.) αποτελούν τα τερματικά σύμβολα της γλώσσας. Στη συνέχεια,

ακολουθούν δύο παραδείγματα δήλωσης μεταβλητών, μία απλού τύπου δεδομένων και μία δήλωσης πίνακα.

(BOOLEAN res): Δήλωση λογικής μεταβλητής res.

(KEYARRAY x 10): Δήλωση μεταβλητής x, τύπου πίνακα κλειδιών μεγέθους 10.

$$\text{var}-declaration \rightarrow (data-type\ \ identifier)\ |\ (data-array\ \ identifier\ \ integer)$$

7.2.3 Τελεστές

Στην παράγραφο αυτή θα γίνει αναφορά στους τελεστές που υποστηρίζονται από τη γλώσσα. Οι τελεστές που υποστηρίζονται είναι με ένα ή δύο ορίσματα και γράφονται πριν τα ορίσματα (prefix notation). Τα ορίσματα μπορεί να είναι και εκφράσεις, των οποίων η αποτίμηση δίνει συμβατούς τύπους δεδομένων. Ο τρόπος με τον οποίο γίνεται η αποτίμηση θα αναλυθεί στη συνέχεια.

Οι τελεστές λογικών πράξεων που υποστηρίζονται είναι: λογική σύζευξη που συμβολίζεται με το AND, λογική διάζευξη που συμβολίζεται με το OR και ο τελεστής λογικής άρνησης που συμβολίζεται με το NOT.

Οι αριθμητικοί τελεστές δέχονται δύο ορίσματα και οι πράξεις που υλοποιούν είναι πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), διαίρεση (/) και modulo (%). Οι τελεστές + και – μπορούν να χρησιμοποιηθούν και για τον προσδιορισμό πρόσημου σε αριθμητικές τιμές.

Οι σχέσεις σύγκρισης ανάμεσα στους τύπους δεδομένων υλοποιούνται με τη βοήθεια των τελεστών σύγκρισης μικρότερο (<), ίσο (==) και διάφορο (!=).

Η γλώσσα υποστηρίζει και τελεστή ανάθεσης για μεταβλητές και τιμές. Αν θεωρήσουμε τη μεταβλητή “x” του γενετικού προγράμματος τότε ο τελεστής ανάθεσης τιμής θα έχει την μορφή (:= x 3), όπου “:=” είναι ο τελεστής ανάθεσης και “3” είναι η τιμή των δεδομένων εισόδου (αντί για δεδομένα εισόδου θα μπορούσε να υπήρχε μεταβλητή).

$$\begin{aligned}
cond &\rightarrow TRUE\ |\ FALSE\ |\ (cond)\ |\ (NOT\ cond)\ |\\
&\quad (AND\ |\ OR\ \ cond\ \ cond)\ |\\
&\quad (<\ |\ ==\ |\ !=\ \ expr\ \ expr)\\
expr &\rightarrow integer\ |\ real\ |\ string\ |\ identifier\ |\\
&\quad identifier[expr]\ |\\
&\quad (\text{"+"}\ |\ \text{"-"}\ |\ \text{"*"}\ |\ \text{"/"}\ |\ \text{"\%"}\ |\ \text{"-"}\ \ expr\ \ expr)\ |\\
&\quad function-call\\
assign-stmt &\rightarrow (\text{":="}\ \ (identifier\ \ |\ \ identifier[expr])\\
&\qquad (read-stmt\ |\ expr\ |\ send-stmt\ |\ function-call))
\end{aligned}$$

7.2.4 Εντολές

Στην παράγραφο αυτή θα αναφερθούμε στις εντολές που υποστηρίζει η γλώσσα.

Εντολές ανάγνωσης και εγγραφής δεδομένων στη μνήμη: Το σύστημα διαθέτει δύο δομικές μονάδες που αντιστοιχούν σε μνήμες. Η πρώτη αντιστοιχεί σε μνήμη στην οποία μπορούν να αποθηκευτούν ζευγάρια κλειδί/τιμή (έχει μεταβλητό μέγεθος και τη δυνατότητα να μεταβάλει, προσθέσει, αφαιρέσει στοιχείο σε κάποια θέση της μνήμης), ενώ η δεύτερη μπορεί να υποστηρίζει οποιοδήποτε τύπο δεδομένων (έχει σταθερό μέγεθος), αλλά χρησιμοποιείται στην περίπτωση μας για αποθήκευση δεδομένων τύπου ADDRESS. Οι εντολές αυτές είναι οι READ και WRITE. Η εντολή READ δέχεται δύο ορίσματα, το πρώτο προσδιορίζει από ποια μνήμη θα διαβάσει (ADDRESS για τη μνήμη με δεδομένα τύπου ADDRESS και KEYDATA για τη μνήμη με ζευγάρια κλειδί/τιμή), ενώ το δεύτερο προσδιορίζει τη θέση ανάγνωσης. Η εντολή WRITE δέχεται τρία ορίσματα, το πρώτο προσδιορίζει τη μνήμη εγγραφής, το δεύτερο τη θέση εγγραφή και το τρίτο τα δεδομένα εγγραφής. Αν η θέση εγγραφής είναι μεγαλύτερη από το μέγεθος της μνήμης, αντιστοιχεί σε νέα εισαγωγή, ενώ αν τα δεδομένα είναι null, αντιστοιχεί σε διαγραφή της συγκεκριμένης θέσης μνήμης. Τα δεδομένα που γράφονται στη μνήμη πρέπει να είναι συμβατά με τον τύπο δεδομένων της μνήμης. Στη συνέχεια ακολουθούν δύο παραδείγματα ανάγνωσης και εγγραφής στη μνήμη.

> **(READ KEYDATA 5):** ανάγνωση από τη μνήμη με ζευγάρια κλειδί/τιμή από τη θέση 5.
> **(WRITE ADDRESS 8 nodeAddress):** εγγραφή στη μνήμη με ADDRESS στη θέση 8 την τιμή που προσδιορίζεται από τη μεταβλητή nodeAddress (η μεταβλητή nodeAddress θα πρέπει να είναι τύπου ADDRESS).

Εντολή ελέγχου: Η εντολή ελέγχου IF δέχεται δύο ή τρία ορίσματα. Το πρώτο όρισμα αντιστοιχεί στη συνθήκη ελέγχου, από την οποία προκύπτει λογική τιμή (αληθής ή ψευδής). Αν η συνθήκη ελέγχου αποτιμηθεί ως αληθής, τότε αποτιμάται και το δεύτερο όρισμα του IF, διαφορετικά η εκτέλεση μεταβαίνει στο τρίτο όρισμα (εφόσον υπάρχει γιατί είναι προαιρετικό).

Εντολή μετάβασης ελέγχου: Ο συνδυασμός εντολών GOTO και LABEL αποτελεί τη δομή μετάβασης ελέγχου. Το GOTO χρησιμοποιείται μαζί με το LABEL που προσδιορίζει σε πιο σημείο του προγράμματος πραγματοποιείται μετάβαση. Τόσο το GOTO όσο και το LABEL έχουν ως όρισμα μια ακέραια αριθμητική τιμή. Το όρισμα του GOTO προσδιορίζει σε ποια εντολή LABEL θα μεταφερθεί ο έλεγχος (στην εντολή που έχει ως όρισμα την ίδια ακέραια τιμή). Το ζευγάρι GOTO και LABEL χρησιμοποιείται σε συνδυασμό με μια δομή ελέγχου συνθήκης για την υλοποίηση επαναληπτικής

διαδικασίας. Όταν εμφανίζεται εντολή GOTO τότε για να είναι σωστό το πρόγραμμα θα πρέπει να εμφανίζεται και εντολή LABEL με την ίδια ακέραια αριθμητική τιμή με τον εντολή GOTO.

Η εντολή RETURN προσδιορίζει την ολοκλήρωση της εκτέλεσης μιας συνάρτησης (υπό-ρουτίνας). Με τη βοήθεια του RETURN είναι δυνατόν να επιστραφεί και κάποια τιμή, εφόσον επιστρέφει η συνάρτηση.

$$
\begin{aligned}
&read-stmt \rightarrow (READ\ \ (KEYDATA \mid ADDRESS)\ \ expr)\\
&write-stmt \rightarrow (WRITE\ \ (KEYDATA \mid ADDRESS)\ \ expr\ \ expr)\\
&return-stmt \rightarrow (RETURN\ \ \ expr \mid NULL)\\
&goto-stmt \rightarrow (GOTO\ \ \ integer)\\
&label-stmt \rightarrow (LABEL\ \ \ integer)\\
&if\ -stmt \rightarrow (IF\ cond\ stmt) \mid (IF\ cond\ stmt\ stmt)\\
&stmt \rightarrow assign-stmt \mid if\ -stmt \mid read-stmt \mid write-stmt \mid\\
&\qquad goto-stmt \mid label-stmt \mid return-stmt \mid\\
&\qquad function-call \mid stmt \mid serial \mid\\
&\qquad \text{var}-declaration \mid send-stmt \mid expr
\end{aligned}
$$

7.2.5 Βοηθητικές Εντολές

Εκτός από τις βασικές εντολές και τους τελεστές που αναφέρθηκαν παραπάνω υπάρχουν κάποια στοιχεία της γλώσσας που έχουν βοηθητικό ρόλο ή προσφέρουν επιπλέον χαρακτηριστικά.

Βοηθητικό ρόλο στη γλώσσα έχει ο τελεστής SERIAL. Ο συγκεκριμένος τελεστής έχει βοηθητικό ρόλο και η παρουσία του δεν είναι απαραίτητο να απεικονίζεται στη σχηματική αναπαράσταση του δέντρου. Ένας κόμβος τύπου SERIAL μπορεί να έχει 1 έως v παιδιά. Κάθε παιδί αντιστοιχεί σε εντολές οι οποίες εκτελούνται διαδοχικά, από τα αριστερά προς τα δεξιά. Ουσιαστικά αποτελεί ένα τελεστή για την ομαδοποίηση εντολών.

Τόσο οι τελεστές όσο και οι εντολές στις οποίες αναφερθήκαμε μπορούν να εμπλουτιστούν με επιπλέον ανάλογα με την εφαρμογή που χρησιμοποιεί τη γλώσσα. Για παράδειγμα, σε μια ρομποτική εφαρμογή, βασικές συναρτήσεις θα μπορούσαν να ήταν τα στρίψε αριστερά, στρίψε δεξιά κτλ. Στην περίπτωση μας, η γλώσσα έχει επαυξηθεί ώστε να μπορεί να υποστηρίζει προγράμματα που χρησιμοποιούν επικοινωνίας μέσω δικτύου. Για το σκοπό αυτό χρησιμοποιείται ένας συνδυασμός εντολών οι TIMEOUT και SENDMESSAGE. Η εντολή TIMEOUT δέχεται δύο ορίσματα. Το πρώτο είναι μια εντολή τύπου SENDMESSAGE που προσδιορίζει τον τύπο του μηνύματος που αποστέλλεται και το δεύτερο όρισμα είναι ένας ακέραιος ο οποίος

προσδιορίζει τη διάρκεια για την οποία η αποστολή ενός μηνύματος θεωρείται έγκυρη. Στην εντολή SENDMESSAGE πρώτο όρισμα είναι ένας προσδιοριστής που καθορίζει τον τύπου μηνυμάτων, ενώ στη συνέχεια ακολουθούν ορίσματα που αφορούν τον κόμβο στον οποίο αποστέλλεται το μήνυμα και τυχόν δεδομένα που αποστέλλονται. Στον παρακάτω πίνακα παρατίθενται οι τύποι μηνυμάτων, καθώς και τα ορίσματα του για κάθε ένα από αυτούς.

Πίνακας 7.3: Τύποι Μηνυμάτων

Προσδιοριστής	Ορίσματα	Περιγραφή
pingID	ADDRESS add	Μήνυμα τύπου ping στον κόμβο με διεύθυνση add. Σε περίπτωση που ο κόμβος δεν απαντήσει, θεωρούμε πως έχει απομακρυνθεί από το δίκτυο.
successorListID	ADDRESS add	Μήνυμα προς τον κόμβο με διεύθυνση add, με αίτημα για τη λίστα επόμενων κόμβων του.
getSuccessorID	ADDRESS add	Μήνυμα προς τον κόμβο με διεύθυνση add, με αίτημα τη διεύθυνση του επόμενου του κόμβου.
findPredecessorID	ADDRESS add, ADDRESS node	Μήνυμα προς τον κόμβο με διεύθυνση add, με αίτημα τη διεύθυνση του προηγούμενου κόμβου του κόμβου node.
notifyID	ADDRESS add, ADDRESS node	Μήνυμα προς τον κόμβο με διεύθυνση add, με ειδοποίηση πως ο κόμβος node μπορεί να είναι ο προηγούμενος του.
getKeyListID	ADDRESS nodeAddress, ADDRESS succ	Μήνυμα προς τον κόμβο με διεύθυνση succ, με αίτημα τη λίστα κλειδιών/τιμών για τον κόμβο με διεύθυνση nodeAddress.
findNodeID	ADDRESS succ, ADDRESS key	Μήνυμα προς τον κόμβο με διεύθυνση succ, με αίτημα τη διεύθυνση του κόμβου που έχει το κλειδί key.
getDataID	ADDRESS s, ADDRESS key	Μήνυμα προς τον κόμβο με διεύθυνση s, με αίτημα τα δεδομένα που αντιστοιχούν στο κλειδί key.
insertKeyID	ADDRESS s, DATAKEY dataKey	Μήνυμα προς τον κόμβο με διεύθυνση s, με αίτημα να αποθηκεύσει το ζευγάρι κλειδί/τιμή dataKey.
sendKeyListID	KEYDATAARRAY keydataList, ADDRESS succ	Μήνυμα προς τον κόμβο με διεύθυνση s, με τη λίστα κλειδιών/τιμές του κόμβου.

Σε περίπτωση όπου η λειτουργικότητα που αφορά την επικοινωνία μεταξύ κόμβων δεν απαιτείται για την εφαρμογή, τότε μπορεί χωρίς πρόβλημα να αγνοηθεί στα γενετικά προγράμματα.

$$program -> (serial)$$

$$serial \rightarrow (serial \mid stmt(serial)^*)$$
$$send-stmt \rightarrow (TIMEOUT \; (SENDMESSAGE \;\; identifier \;\; expr^*) \; integer)$$

7.2.6 Συμβατότητα Πράξεων

Όπως αναφέρθηκε παραπάνω η γλώσσας υποστηρίζει αρκετούς τύπους δεδομένων και αρκετούς τελεστές. Όμως δεν είναι επιτρεπτές όλες οι πράξεις για όλους τους τύπους δεδομένων, ώστε να διατηρείται η συνέπεια στα προγράμματα. Για παράδειγμα δεν επιτρέπεται να προστεθεί ακέραιος και συμβολοσειρά. Στον παρακάτω πίνακα προσδιορίζονται οι πράξεις ανάμεσα στους τύπος δεδομένων, καθώς και οι συγκρίσεις οι οποίες υποστηρίζονται.

Πίνακας 7.4: Τελεστές και Τύποι Δεδομένων

Τελεστής	Τελεστέος Α	Τελεστέος Β
-, +	INT	-
-, +	DOUBLE	-
NOT	BOOLEAN	-
AND, OR	BOOLEAN	BOOLEAN
-, +, *, /, %, ==, <, !=	INT	INT
-, +, *, /, ==, <, !=	DOUBLE	DOUBLE
-, +, *, /, ==, <, !=	INT	DOUBLE
-, +, *, /, ==, <, !=	DOUBLE	INT
-, +, ==, <, !=	ID	ID
-, +, ==, <, !=	KEY	KEY
-,+ , ==, <, !=	ADDRESS	ADDRESS
+, ==, <, !=	STRING	STRING
==, !=	BOOLEAN	BOOLEAN
==, <, !=	ID	INT
==, <, !=	INT	ID
==, <, !=	ID	KEY
==, <, !=	KEY	ID
==, <, !=	ADDRESS	INT
==, <, !=	INT	ADDRESS

Ο παραπάνω πίνακας πρέπει να ανανεώνεται στην περίπτωση προσθήκης νέων τελεστών ή τύπων δεδομένων.

7.2.7 Συναρτήσεις

Στη γλώσσα υπάρχει υποστήριξη και για συναρτήσεις. Οι συναρτήσεις αποτελούν μια ακολουθία από μηδέν ή περισσότερες εντολές. Στη περίπτωση μας πρέπει να δηλώνονται ανεξάρτητα από το υπόλοιπο πρόγραμμα. Κάθε συνάρτηση χαρακτηρίζεται από το όνομα της, μπορεί να διαθέτει μηδέν ή περισσότερα ορίσματα και επιστρεφόμενο τύπο (ο τύπος VOID χρησιμοποιείται για να δηλώσει πως μια

συνάρτηση δεν επιστρέφει αποτέλεσμα). Κάθε όρισμα περιγράφεται από το όνομα του και τον τύπο δεδομένων του.

$$function-call \rightarrow (identifier\ args)$$
$$args \rightarrow args-list \mid \varepsilon$$
$$args-list = args-list\ \ expr \mid expr$$

7.2.8 Επιπλέον Δυνατότητες

Η γλώσσα υποστηρίζει κάποιες επιπλέον δυνατότητες προσαρμοσμένες για τους τύπους δεδομένων που υποστηρίζει. Οι δυνατότητες αυτές αφορούν κάποιους από τους τύπους δεδομένων για τους αλγορίθμους δικτύων. Αρχικά, θα αναφερθούμε στον τελεστή "." ο οποίος μπορεί να χρησιμοποιηθεί από τους τύπους δεδομένων ADDRESS και DATAKEY για την προσπέλαση του ενός μέλους του ζευγαριού (με ".ID" λαμβάνεται η τιμή του μέλους ID για τον τύπο δεδομένων ADDRESS, ενώ με .KEY και .DATA η τιμή του μέλους KEY και DATA, αντίστοιχα, στον τύπο δεδομένων DATAKEY). Ακόμα, για τους τύπους δεδομένων πίνακα υποστηρίζεται ο τελεστής ".SIZE" με τον οποίο επιστρέφεται το μέγεθος του πίνακα. Το χαρακτηριστικό αυτό προστέθηκε για να διευκολυνθεί η προσπέλαση των στοιχείων των πινάκων σε κάποια επαναληπτική διαδικασία.

7.2.9 Παράδειγμα

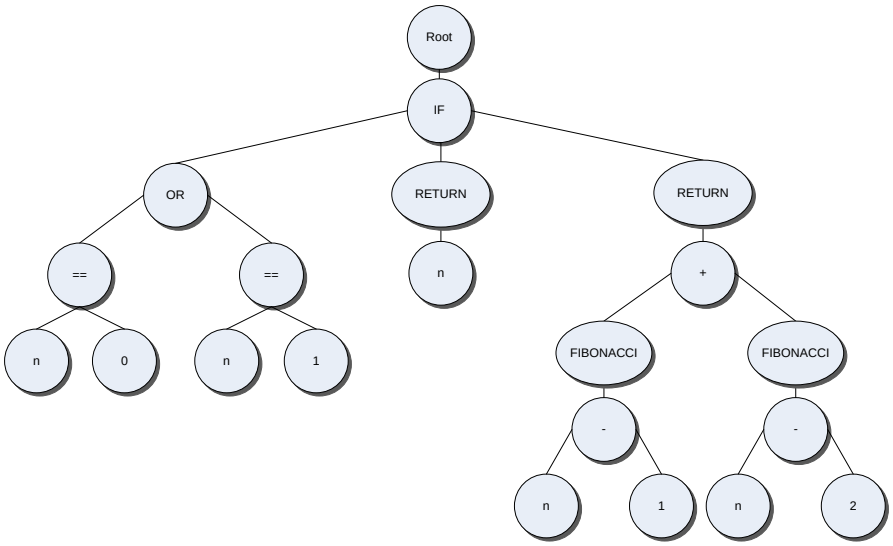
Στη συνέχεια, παρουσιάζεται μια συνάρτηση η οποία υπολογίζει το νιοστό αριθμό της ακολουθίας Fibonacci χρησιμοποιώντας το συντακτικό της γλώσσας που αναφέρθηκε παραπάνω. Η συνάρτηση δέχεται ως είσοδο έναν ακέραιο n και υπολογίζει το νιοστό αριθμό της ακολουθίας Fibonacci χρησιμοποιώντας αναδρομή.

```
((
(IF (OR (== n 0) (== n 1))
      (RETURN n)
      (RETURN (+ (FIBONACCI (- n 1)) (FIBONACCI (- n 2))))
)
))
```

Η γλώσσα αναπαρίσταται γραφικά με τη βοήθεια δεντρικής δομής, όπου κάθε κόμβος του δέντρου αντιστοιχεί σε κάποιο από τα στοιχεία του προγράμματος. Στην Εικόνα 7.1 παρουσιάζεται αυτή η δεντρική δομή.

Στη γραφική αναπαράσταση της συνάρτησης παρατηρούμε την ύπαρξη 11 εσωτερικών κόμβων οι οποίοι έχουν αναγνωριστικά συναρτήσεων (π.χ. IF, RETURN, OR, +, -). Παρατηρούμε και την ύπαρξη των εσωτερικών κόμβων με αναγνωριστικό FIBONACCI, οι οποίοι αντιστοιχούν στην αναδρομική κλίση της συνάρτησης. Οι εννιά εξωτερικοί κόμβοι (φύλλα), έχουν αναγνωριστικά τερματικών συμβόλων (π.χ. μεταβλητή n,

σταθερές 0, 1, 2). Αμέσως μετά τον κόμβο ρίζα του δέντρου (ROOT), βρίσκεται ο κόμβος με αναγνωριστικό IF ο οποίος αντιστοιχεί στην πρώτη συνάρτηση μετά τις πιο αριστερές παρενθέσεις της συνάρτησης. Τα παιδιά του, από αριστερά προς τα δεξιά, είναι συνθήκη ελέγχου, το then κομμάτι και τέλος το else κομμάτι. Στο παράδειγμα αυτό γίνεται εμφανής, ο ρόλος των παρενθέσεων στη γλώσσα, όπου τόσο οι εντολές όσο και οι τελεστές περικλείονται από παρενθέσεις. Η χρήση των παρενθέσεων βοηθά στο διαχωρισμό των βασικών συστατικών της γλώσσας.



Εικόνα 7.1: Δεντρική δομή προγράμματος fibonacci

Η γλώσσα που δημιουργήθηκε για να εκφράσει τα γενετικά προγράμματα, προσφέρει τη δυνατότητα τόσο τα προγράμματα όσο και τα δεδομένα να έχουν την ίδια μορφή. Κάτι τέτοιο είναι αρκετά βολικό, αφού επιτρέπει ένα πρόγραμμα του γενετικού πληθυσμού, αρχικά, να το χειριστούμε ως δεδομένα και στη συνέχεια να το εκτελέσουμε ως πρόγραμμα (π.χ. ανάθεση σε μεταβλητή τιμής που θα προκύψει από την εκτέλεση συνάρτησης). Ακόμα, οι εκφράσεις μέσω των οποίων σχηματίζονται τόσο τα προγράμματα όσο και τα δεδομένα, έχουν μορφή αντίστοιχη με το συντακτικό δέντρο που παράγουν πολλοί μεταγλωττιστές για την αναπαράσταση προγραμμάτων υπολογιστών. Κάτι τέτοιο μας επιτρέπει να έχουμε πρόσβαση στο συντακτικό δέντρο ενός προγράμματος ώστε να μπορούμε να τροποποιούμε γενετικά κομμάτια του προγράμματος, ενώ παρέχει τη δυνατότητα για αναπαράσταση του προγράμματος με κατανοητό τρόπο. Επιπλέον, η γλώσσα αυτή, διευκολύνει τη δημιουργία προγραμματιστικών δομών των οποίων το μέγεθος και σχήμα μεταβάλλεται δυναμικά, χωρίς να είναι προκαθορισμένο εκ των προτέρων.

7.2.10 Κλειστότητα

Σε προηγούμενο κεφάλαιο έγινε αναφορά στην ιδιότητα της κλειστότητας σε σχέση με τα γενετικά προγράμματα. Στην παράγραφο αυτή, πριν ολοκληρωθεί η παρουσίαση της γενετικής γλώσσας, θα αναφερθούμε στην ιδιότητα αυτή σε σχέση με τη γλώσσα που δημιουργήθηκε. Η ιδιότητα της κλειστότητας είναι γενικά επιθυμητή, αλλά δεν αποτελεί πανάκεια. Αν η συνθήκη της κλειστότητας δεν υφίσταται, τότε απαιτείται διαφορετικός χειρισμός ως προς τις περιπτώσεις όπου προγράμματα δεν οδηγούν σε σωστά αποτελέσματα (π.χ. κάποια ποινή σε τέτοια προγράμματα).

Στην περίπτωση της δικιά μας ψευδογλώσσας χρησιμοποιείται η προσέγγιση όπου δεν εφαρμόζεται η κλειστότητα. Οι τύποι δεδομένων τις ψευδογλώσσας είναι πολλοί και πολλές πράξεις ανάμεσα σε αυτούς δεν έχουν πάντα νόημα (π.χ. πρόσθεση ενός INT με ένα STRING). Γι’ αυτό κρίθηκε σκόπιμο να μην υπάρχει κλειστότητα για όλους τους τελεστές, αλλά σε περίπτωση αντικανονικών πράξεων να θεωρείται πως το γενετικό πρόγραμμα δεν είναι κατάλληλο. Η περίπτωση αυτή αντιστοιχεί στην ανάπτυξη συντακτικών δομών με περιορισμούς. Τα άτομα που αποτελούν τον πληθυσμό έχουν περιορισμένη συντακτική δομή, η οποία καθορίζεται από κανόνες συντακτικής κατασκευής, ανάλογα με το πρόβλημα που επιλύεται. Τέτοιοι περιορισμοί είναι:

- Ο αρχικός πληθυσμός προγραμμάτων, πρέπει να δημιουργηθεί ώστε τα αρχικά άτομα που τον αποτελούν να υπόκεινται στους συντακτικούς κανόνες (π.χ. περιορισμός επιλογής συναρτήσεων σε συγκεκριμένους εσωτερικούς κόμβους).
- Όταν πραγματοποιείται πράξη η οποία μεταβάλει ένα άτομο του πληθυσμού (π.χ. crossover, mutation), πρέπει η συντακτική δομή να εξακολουθεί να ισχύει. Για παράδειγμα, η διασταύρωση (crossover) τροποποιείται ώστε να παράγει απογόνους με τη απαιτούμενη συντακτική δομή. Τέτοιοι περιορισμοί στη δομή είναι να αποκλειστεί η ρίζα του δέντρου ως σημείο επιλογής για διασταύρωση και στους δύο γονείς ή για πιο πολύπλοκες περιπτώσεις να επιλέγεται αρχικά ένα σημείο διασταύρωσης στον πρώτο γονέα, αλλά να περιορίζεται η επιλογή στον δεύτερο γονέα ώστε να είναι ίδιου τύπου με τον πρώτο.
- Η μετρική ικανότητας (fitness) πρέπει να λαμβάνει υπόψη τη συντακτική δομή, επικροτώντας τα άτομα που έχουν τη σωστή και τιμωρώντας αυτά που παρουσιάζουν σφάλματα.

7.3 Διερμηνευτής

Το υποκεφάλαιο αυτό είναι πιο τεχνικό και παρουσιάζει στοιχεία που αφορούν τη λειτουργία του διερμηνευτή της γενετικής γλώσσας. Αρχικά, θα παρουσιαστούν οι βασικές μονάδες όπως ο πίνακας συμβόλων, ο λεκτικός και ο συντακτικός αναλυτής, ενώ στη συνέχεια θα δοθεί και ένα παράδειγμα χρήσης τους.

7.3.1 Σύμβολο

Για την περιγραφή των μεταβλητών που χρησιμοποιούνται στη γλώσσα χρησιμοποιείται η κλάση Symbol. Κάθε σύμβολο (symbol) περιγράφεται από το όνομα της μεταβλητής και τον τύπο δεδομένων του συμβόλου. Δύο σύμβολα θεωρούνται ίδια αν έχουν το ίδιο όνομα. Αντίστοιχα, για την περιγραφή των συναρτήσεων συμβόλων που έχουν δημιουργηθεί για τις ανάγκες της γλώσσας χρησιμοποιείται η κλάση FunctionSymbol. Κάθε συναρτησιακό σύμβολο περιγράφεται από το όνομα του, τον τύπο δεδομένων που επιστρέφει (αν επιστρέφει) και τα ορίσματα που δέχεται ως είσοδο.

7.3.2 Πίνακας συμβόλων

Ο πίνακας συμβόλων είναι μια δομή που χρησιμοποιείται για την αποθήκευση συμβόλων μαζί με την πληροφορία που αφορά την τιμή τους. Οι πληροφορίες αυτές συλλέγονται κατά τη φάση της ανάλυσης του προγράμματος. Πιο συγκεκριμένα, κατά τη λεκτική ανάλυση προσδιορίζεται το όνομα ενός προσδιοριστή, ενώ κατά τη σημασιολογική ανάλυση προσδιορίζεται ο τύπος του προσδιοριστή και εισάγεται στον πίνακα συμβόλων. Για κάθε προσδιοριστή αποθηκεύεται στον πίνακα συμβόλων ένα σύμβολο και ένα αντικείμενο το οποίο αντιστοιχεί στο σύμβολο αυτό (π.χ. ένα Object Integer για μια μεταβλητή τύπου INT). Σε κάθε βήμα εκτέλεσης, ουσιαστικά, περιγράφει την κατάσταση του προγράμματος, αφού περιέχει πληροφορία για τις μεταβλητές του προγράμματος. Ανάλογα την εντολή που εκτελείται ανανεώνονται οι τιμές των μεταβλητών που βρίσκονται στον πίνακα συμβόλων, εφόσον αυτό χρειάζεται.

Ο πίνακας συμβόλων υλοποιείται από την κλάση SymbolTable. Ουσιαστικά, πρόκειται για ένα Hashtable όπου γίνεται αντιστοιχία συμβόλων στο Object που αντιστοιχεί στο αντικείμενο αυτό. Ο πίνακας συμβόλων υποστηρίζει λειτουργίες όπως εισαγωγή (insertSymbol), ενημέρωση/τροποποίηση τιμής/τύπου συμβόλου (updateSymbolValue, updateSymbolType), αναζήτηση (lookupSymbol), επισκόπηση (getSymbolValue, getSymbolType) χαρακτηριστικών των συμβόλων. Ειδική μέριμνα έχει ληφθεί για σύμβολα τύπου πίνακα ώστε να λαμβάνεται υπόψη και το μέγεθος του πίνακα. Στη συνέχεια ακολουθεί μια εκτύπωση του πίνακα συμβόλων, όπου περιέχει δύο

μεταβλητές τύπου Double (τις y και res) και μια μεταβλητή τύπου Integer (την x), μαζί με τις τιμές των μεταβλητών.

```
Number of Symbols: 3
[DOUBLE y = 21.0]
[INT x = 15]
[DOUBLE res = 0.7142857142857143]
```

7.3.3 Λεκτικός αναλυτής

Η διαδικασία της λεκτικής ανάλυσης πραγματοποιείται με τη βοήθεια της κλάσης Scanner. Ο σαρωτής είναι υπεύθυνο για τη σάρωση και το διαχωρισμό των λέξεων (token) του προγράμματος τα οποία θα οδηγήσουν στη δημιουργία του συντακτικού δέντρου. Ο σαρωτής ξεπερνά τους λευκούς χαρακτήρες και ξεχωρίζει πότε το token είναι αριθμός (ακέραιος ή πραγματικός), πότε είναι συμβολοσειρά (κείμενο που περικλείεται από “”) και πότε είναι αναγνωριστικό (συμβολικές τιμές που αντιστοιχούν σε ονόματα μεταβλητών ή τυχόν δεσμευμένες λέξεις του συντακτικού της γλώσσας) χρησιμοποιώντας τις κατάλληλες κανονικές εκφράσεις (regular expressions). Ακόμα, επιστρέφει πότε ανοίγει ή κλείνει παρένθεση, αλλά και πότε έχουμε αριστερή ή δεξιά αγκύλη (προσπέλαση στοιχείου πίνακα). Σε κάθε βήμα προσπαθεί να ταιριάξει το μεγαλύτερο ορισμένο token, ενώ επιστρέψει εύρεση άγνωστου συμβόλου σε περίπτωση που δεν μπόρεσε να αναγνωρίσει το token που διαβάζει με βάση τους κανόνες.

7.3.4 Συντακτικός αναλυτής

Ο συντακτικός αναλυτής υλοποιείται από την κλάση Parser και αναλαμβάνει την κατασκευή του δέντρου που συμβολίζει τη δομή του προγράμματος. Ο συντακτικός αναλυτής είναι υπεύθυνος για τον έλεγχο της ορθής σύνταξης των εντολών, την επαλήθευση της γραμματικής της γλώσσας και τη δημιουργία του συντακτικού δέντρου που αντιστοιχεί στο πρόγραμμα. Το συντακτικό δέντρο είναι αυτό που θα εκτελεστεί στη φάση της σημασιολογικής ανάλυσης. Αν ο συντακτικός αναλυτής εντοπίσει σφάλμα στο πρόγραμμα σταματά τη διαδικασία δημιουργίας συντακτικού δέντρου. Η δημιουργία του συντακτικού δέντρου προσφέρει ένα μηχανισμό ανεξαρτητοποίησης της εκτέλεσης του προγράμματος από τη συντακτική ανάλυση. Οι εσωτερικοί κόμβοι του συντακτικού δέντρου αντιστοιχούν σε τελεστές ή σε δεσμευμένες λέξεις της γλώσσας και οι κόμβοι παιδιά τους στους τελεστέους ή τα ορίσματα πάνω στα οποία εφαρμόζονται οι τελεστές. Η δομή του συντακτικού δέντρου είναι τέτοια ώστε να διευκολύνεται η εκτέλεση του προγράμματος.

Η δημιουργία του συντακτικού δέντρου γίνεται με τη βοήθεια του σαρωτή. Πιο συγκεκριμένα, ο σαρωτής προμηθεύει το συντακτικό αναλυτή με ένα-ένα τα token του προγράμματος. Ο συντακτικός αναλυτής προσπερνά τις παρενθέσεις και τις αγκύλες με βάση τη γραμματική της γλώσσας και αναγνωρίζει τα token για τα οποία ανάλογα με τον τύπο τους θα δημιουργήσει τους κατάλληλους κόμβους για το συντακτικό δέντρο. Για παράδειγμα, αν το token που προήλθε από το σαρωτή είναι το “+” τότε ο συντακτικός αναλυτής δημιουργεί ένα κόμβο για τον τελεστή “+” και στη συνέχεια αναγνωρίζει πως πρόκειται για τελεστή που δέχεται δύο ορίσματα. Άρα, ο κόμβος αυτός θα έχει δύο παιδιά τα οποία θα προκύψουν από συντακτική ανάλυση των token που ακολουθούν από το “+”.

Κατά τη δημιουργία του συντακτικού δέντρου, ο συντακτικός αναλυτής αναλαμβάνει να συλλέξει πληροφορία που αφορά τα σημεία μετάβασης εκτέλεσης του προγράμματος. Οι μεταβάσεις εκτέλεσης στην ψευδογλώσσας υλοποιούνται με ζευγάρια εντολών GOTO-LABEL. Όταν κατά την εκτέλεση συναντούμε ένα κόμβο GOTO τότε ο έλεγχος ροής του προγράμματος μεταφέρεται στο αντίστοιχο LABEL. Για να μην απαιτείται αναζήτηση στο δέντρο για τον κατάλληλο κόμβο LABEL, ο συντακτικός αναλυτής καταγράφει σε ένα Hashtable τους κόμβους LABEL και τους διαχωρίζει ανάλογα με το χαρακτηριστικό αριθμό του καθενός. Το Hashtable αυτό από κόμβους και τιμές που αντιστοιχούν στα LABEL, εισάγεται στον διερμηνευτή κατά την εκτέλεση του προγράμματος.

7.3.5 Δομή συντακτικού δέντρου

Η δομή στην οποία αποθηκεύεται το συντακτικό δέντρο υλοποιείται με τη βοήθεια των κλάσεων Tree και TreeNode. Η κλάση Tree είναι αυτή που περιέχει τη ρίζα το δέντρου, ενώ οι κόμβοι του δέντρου υλοποιούνται με τη κλάση TreeNode. Κάθε κόμβος του δέντρου διατηρεί πληροφορία που αφορά αυτόν αλλά και γειτονικούς κόμβους. Στον παρακάτω πίνακα συνοψίζονται οι πληροφορίες αυτές.

Πίνακας 7.5: Πληροφορίες κόμβου δέντρου

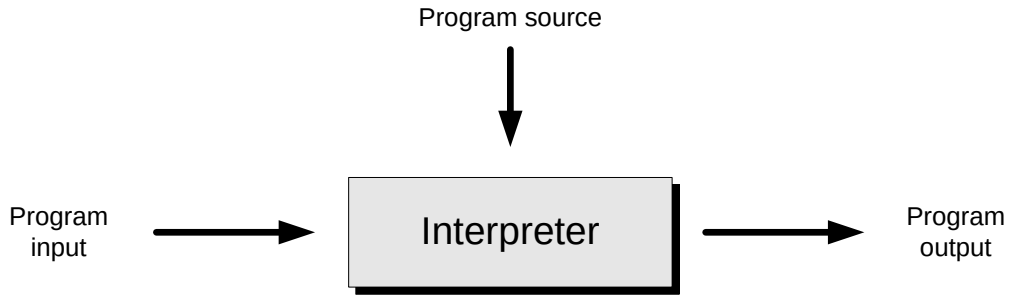
Μεταβλητές	Παρατηρήσεις
typeNode	Αριθμητική τιμή που έχει προκύψει από το σαρωτή και προσδιορίζει τον τύπο/είδος του token (π.χ. ARITHMETIC, IDENTIFIER, STRING).
typeName	Συμβολοσειρά με το όνομα του token όπως προέκυψε από το σαρωτή (π.χ. “IF”, “+”, “x”).
numChildren	Το πλήθος των παιδιών του κόμβου.
nextSibling	Αναφέρεται στον κόμβο αδερφό (έχουν τον ίδιο πατέρα και είναι γειτονικοί). Χρησιμοποιείται κατά την προσπέλαση του δέντρου για μετάβαση σε διπλανό υποδέντρο.

parent	Πατέρας του κόμβου.
children	Λίστα με τα παιδία του κόμβου.
depth	Βάθος του κόμβου στο δέντρο.
arraypos	Λογική μεταβλητή που είναι αληθής, αν ο κόμβος αποτελεί το δείκτη προσπέλασης ενός πίνακα. Έχει βοηθητικό ρόλο κατά την εκτύπωση του δέντρου.

Η υλοποίηση του δέντρου είναι τέτοια ώστε να είναι εύκολη η εφαρμογή των γενετικών τελεστών. Πιο συγκεκριμένα, υποστηρίζεται η προσθήκη στο δέντρο ενός νέου υποδέντρου (insertNewChild), η αντικατάσταση ενός υπάρχοντος υποδέντρου (replaceChild), η εύρεση όλων τον κόμβων ενός συγκεκριμένου τύπου (getNodes), η οποία βοηθά στην επιλογή κόμβων για διασταύρωση οι οποίοι θα οδηγήσουν σε έγκυρα προγράμματα και ενημέρωση του βάθους τον κόμβων μετά από αλλαγές στη δομή του δέντρου (setNodesDepth). Επιπλέον, υπάρχει η δυνατότητα για εκτύπωση όλων των κόμβων του συντακτικού δέντρου καθώς και εκτύπωση του προγράμματος από το οποίο προήλθε το συντακτικό δέντρο.

7.3.6 Λειτουργία Διερμηνευτή (Interpreter)

Στην παράγραφο αυτή θα παρουσιαστεί η βασική μονάδα εκτέλεσης της γενετικής γλώσσας που είναι ο Διερμηνευτής. Ο Διερμηνευτής μπορεί να θεωρηθεί ως μια εικονική μηχανή (virtual machine) δέχεται ως είσοδο ένα πρόγραμμα και την είσοδο για το πρόγραμμα και παράγει την έξοδο για το πρόγραμμα αυτό, μέσω μιας από κάτω προς τα πάνω (bottom-up) αποτίμησης. Για να επιτύχει αυτό διασχίζει το συντακτικό δέντρο και εκτελεί σε υψηλό επίπεδο το πρόγραμμα με βάση τους κόμβους που συναντά στη διαδρομή του. Κατά την εκτέλεση του προγράμματος υπολογίζει το αποτέλεσμα, ενώ μεταβάλλει και την κατάσταση κάθε συντακτικού δέντρου, η οποία εκφράζεται μέσω των τιμών των μεταβλητών στον πίνακα συμβόλων.



Εικόνα 7.2: Σχηματική Αναπαράσταση Λειτουργίας Διερμηνευτή

Για την εκτέλεση, ενός προγράμματος ο Διερμηνευτής πρέπει να έχει στη διάθεση του το συντακτικό δέντρο του προγράμματος, το Hashtable που αφορά τους κόμβους με

χαρακτηριστικό LABEL των σημείων μετάβασης εκτέλεσης και το σύνολο από όποιες επιπλέον συναρτήσεις έχουν οριστεί και μπορεί να υπάρχουν στον κώδικα του προγράμματος (οι συναρτήσεις αυτές είναι επιπλέον συναρτήσεις στα ήδη υπάρχοντα δομικά στοιχεία της γλώσσας). Κάθε πρόγραμμα που εκτελείται μπορεί να δέχεται ως είσοδο τιμές από κάποιες μεταβλητές και να επιστρέφει κάποια τιμή. Έτσι, κάθε συντακτικό δέντρο που εκτελείται μπορεί να αποτελεί είτε ένα ανεξάρτητο πρόγραμμα είτε μια συνάρτηση για κάποιο πιο σύνθετο πρόγραμμα. Ο Διερμηνευτής έχει πρόσβαση στις μνήμες που διαθέτει κάθε πρόγραμμα (KeyDataMemory keyMem, Memory mem), αλλά και σε έναν καθολικό πίνακα συμβόλων όπου υπάρχουν μεταβλητές με καθολική εμβέλεια ανάμεσα στις κλίσεις συναρτήσεων (symbolTable globalVariables).

Ο Διερμηνευτής ξεκινά την εκτέλεση από τη ρίζα του συντακτικού δέντρου. Διατρέχει τους κόμβους ακολουθώντας μια πρώτα κατά βάθος διαδρομή και ενεργεί ανάλογα με τον τύπο του κόμβου. Όταν συναντάει δηλώσεις μεταβλητών τις εισάγει στον πίνακα συμβόλων, ενώ όταν βρει κόμβο GOTO τότε συμβουλεύεται το Hashtable με τους κόμβους LABEL και συνεχίζει την εκτέλεση από τον κατάλληλο κόμβο τύπου LABEL. Η εκτέλεση ολοκληρώνεται αν έχουν εκτελεστεί όλες οι εντολές σε ένα μονοπάτι του δέντρου ή αν συναντήσουμε κόμβο τύπου RETURN, ο οποίος και σηματοδοτεί την ολοκλήρωση εκτέλεσης του συντακτικού δέντρου.

Σημαντικό ρόλο στην εκτέλεση ενός προγράμματος επιτελεί η μέθοδος getExpressionValue, η οποία εξετάζει έναν κόμβο ή ένα υποδέντρο και επιστρέφει το αποτέλεσμα που προκύπτει από αυτό, καθώς και τον τύπο δεδομένων. Αν πρόκειται για υποδέντρο μεταβιβάζει τον υπολογισμό των παιδιών στην κατάλληλη μέθοδο (π.χ. αποστολή μηνύματος, πρόσβαση στη μνήμη, τελεστής αριθμητικής πράξης). Αν πρόκειται για κόμβο ελέγχει αν ο κόμβος είναι λογική σταθερά (TRUE ή FALSE), αν είναι μεταβλητή (είτε από το καθολικό πίνακα συμβόλων είτε από το σύνολο συμβόλων του υποδέντρου) και υπολογίζει ειδικές περιπτώσεις όπου γίνεται χρήση της “.” (π.χ. .SIZE, .KEY, .ID). Επίσης, προσδιορίζει τον τύπο που αντιστοιχεί σε κάποια αριθμητική τιμή (π.χ. αν η τιμή του κόμβου είναι 15 αναγνωρίζει με τη βοήθεια του προγράμματος πως “αναμένεται” να είναι ακέραιος ο τύπος δεδομένων). Δεδομένου πως ο Διερμηνευτής επιτρέπει κλήσεις συναρτήσεων είναι σημαντικό να μην αλλοιώνεται ο πίνακας συμβόλων ανάμεσα στις κλήσεις αυτές. Αυτό επιτυγχάνεται με τη βοήθεια της κλάσης ExecutionState, η οποία αποθηκεύει τον πίνακα συμβόλων και άλλες μεταβλητές που περιγράφουν την κατάσταση ενός προγράμματος. Εφόσον, σε κάποιο πρόγραμμα εμφανιστεί κλήση συνάρτησης, τότε αποθηκεύεται η τρέχουσα κατάσταση

του προγράμματος (addNewExecutionState) και ο έλεγχος εκτέλεσης περνάει στη συνάρτηση αυτή. Με την ολοκλήρωση της κλήσης της συνάρτησης ο έλεγχος επανέρχεται στο σημείο όπου ξεκίνησε η κλήση στο αρχικό πρόγραμμα και συνεχίζεται η εκτέλεση από το σημείο αυτό που είχε διακοπεί χωρίς αλλαγή στην κατάσταση.

Στη συνέχεια ακολουθεί ένα παράδειγμα χρήσης του Διερμηνευτή. Για το παράδειγμα αυτό θα χρησιμοποιηθεί το παρακάτω πρόγραμμα:

```
((
(INT i)(:= i 2)
(LABEL 1)
(IF
     (< i (/ n 2))
     (IF
           (== (% n i) 0)
           (RETURN FALSE)((:= i (+ i 1))(GOTO 1))
     )
)
(RETURN TRUE)
))
```

Το πρόγραμμα αυτό δέχεται ως είσοδο ένα ακέραιο αριθμό n και εξετάζει αν ο αριθμός αυτός είναι πρώτος (ένας αριθμός είναι πρώτος αν διαιρείται μόνο από τον εαυτό του και τη μονάδα). Αν είναι πρώτος τότε επιστρέφει TRUE διαφορετικά επιστρέφει FALSE. Στον αλγόριθμο που ακολουθείται αρκεί να εξεταστούν οι αριθμοί στο $\left[2, n/2\right.$, αν διαιρούν το n.

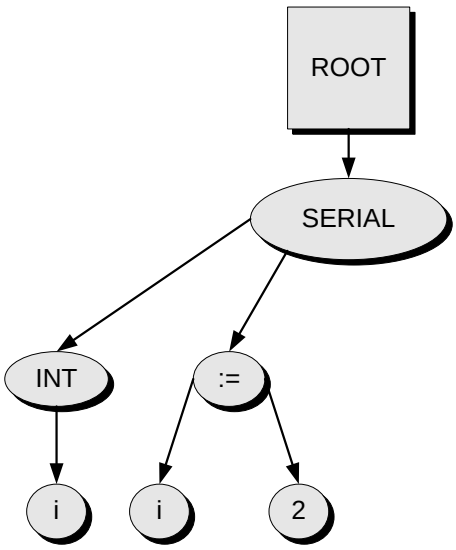
Η διαδικασία επεξεργασίας του προγράμματος ξεκινά με τη χρήση του σαρωτή (Scanner). Ο σαρωτής αναγνωρίζει ένα-ένα τα token του προγράμματος. Πιο συγκεκριμένα, στο παραπάνω πρόγραμμα αρχικά θα έχουμε τρεις φορές αριστερή παρένθεση (*LEFT_PAR*), στη συνέχεια δύο αναγνωριστικά τα INT και i (*IDENTIFIER*). Η διαδικασία αυτή ολοκληρώνεται με τρεις δεξιές παρενθέσεις (*RIGHT_PAR*), ενώ όταν έχει “καταναλώσει” και το τελευταίο στοιχείο της εισόδου ο σαρωτής επιστρέφει EOF (End-Of-File), σηματοδοτώντας το τέλος επεξεργασίας.

Τα στοιχεία που επιστρέφει ο σαρωτής τα χρησιμοποιεί ο συντακτικός αναλυτής (Parser) για τη δημιουργία του συντακτικού δέντρου. Πιο συγκεκριμένα, ο συντακτικός αναλυτής αναγνωρίζει ανάλογα με το πλήθος των παρενθέσεων που ανοίγουν και κλείνουν, πότε πρόκειται για κόμβο τύπου SERIAL και πότε πρέπει να δημιουργήσει νέο παιδί για κάποιο κόμβο. Ο συντακτικός αναλυτής δημιουργεί το συντακτικό δέντρο, δυναμικά, με βάση τις τιμές που δέχεται από το σαρωτή σε κάθε βήμα. Κάθε τμήμα του προγράμματος που περικλείεται από παρενθέσεις, αποτελεί, πρακτικά, ένα υποδέντρο του συντακτικού δέντρου. Άρα, το άνοιγμα μιας αριστερής παρένθεσης σηματοδοτεί τη

δημιουργία ενός κόμβου παιδί. Κατά τη δημιουργία του συντακτικού δέντρου, ο συντακτικός αναλυτής ελέγχει αν όσες παρενθέσεις ανοίγουν τόσες κλείνουν και κατά πόσο το πρόγραμμα είναι ορθό συντακτικά. Για παράδειγμα, όταν συναντήσει κόμβο τύπου LABEL, αναμένει να συνοδεύεται από ένα όρισμα (έκφραση ή αριθμό) και στη συνέχεια να κλείνει μια παρένθεση ($LABEL < \exp r \mid Integer >$). Αν δε συμβαίνει αυτό τότε υπάρχει κάποιο λάθος στο συντακτικό. Ο συντακτικός αναλυτής σε καμία περίπτωση δεν είναι υπεύθυνος να ελέγξει κατά πόσο το πρόγραμμα είναι σημασιολογικά ορθό ή δεν οδηγεί σε ατέρμονες επαναλήψεις. Η σημασιολογική πλευρά του προγράμματος είναι κάτι που αναλαμβάνει ο Διερμηνευτής.

Για το συγκεκριμένο πρόγραμμα, αρχικά, δημιουργεί έναν κόμβο τύπου SERIAL (κόμβος που έχει από $1 \ldots n$ παιδιά, αναγνωρίζεται σε περιπτώσεις όπου έχουμε διαδοχικά ανοίγματα αριστερών παρενθέσεων και ομαδοποιεί διαδοχικά υποδέντρα που πρέπει να εκτελεστούν ακολουθιακά) μετά τη ρίζα. Στη συνέχεια, για τους προσδιοριστές, INT και i, δημιουργεί ένα υποδέντρο όπου ο κόμβος με τιμή INT έχει ως παιδί τον κόμβο με τιμή i και είναι το πρώτο παιδί του κόμβου SERIAL.

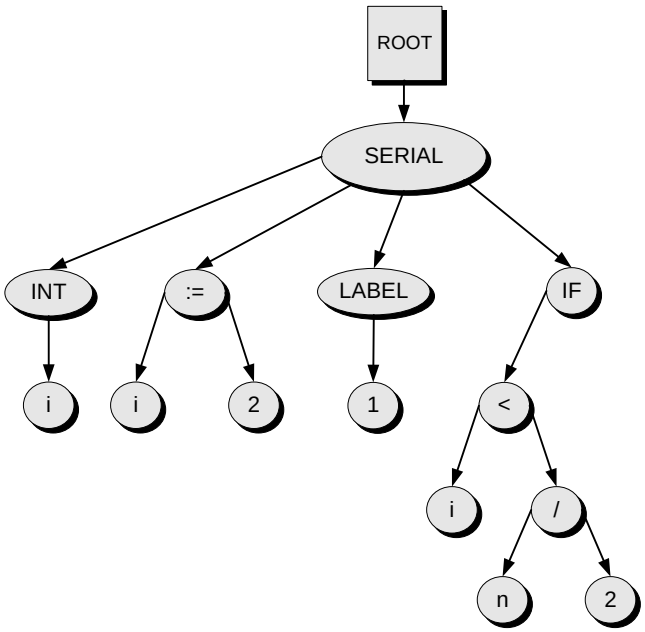
Για το συγκεκριμένο πρόγραμμα, αρχικά, θα δημιουργηθούν πέντε υποδέντρα για τον κόμβο SERIAL όπου θα έχουν για ρίζα INT, ":=", LABEL, IF και RETURN αντίστοιχα. Ακόμα, ο συντακτικός αναλυτής θα δημιουργήσει και το Hashtable που καταγράφει τους κόμβους τύπου LABEL. Στη συνέχεια, παρουσιάζεται σχηματικά η δημιουργία του συντακτικού δέντρου.



Εικόνα 7.3: Δέντρο από τη σάρωση του (((INT i)(:= i 2)

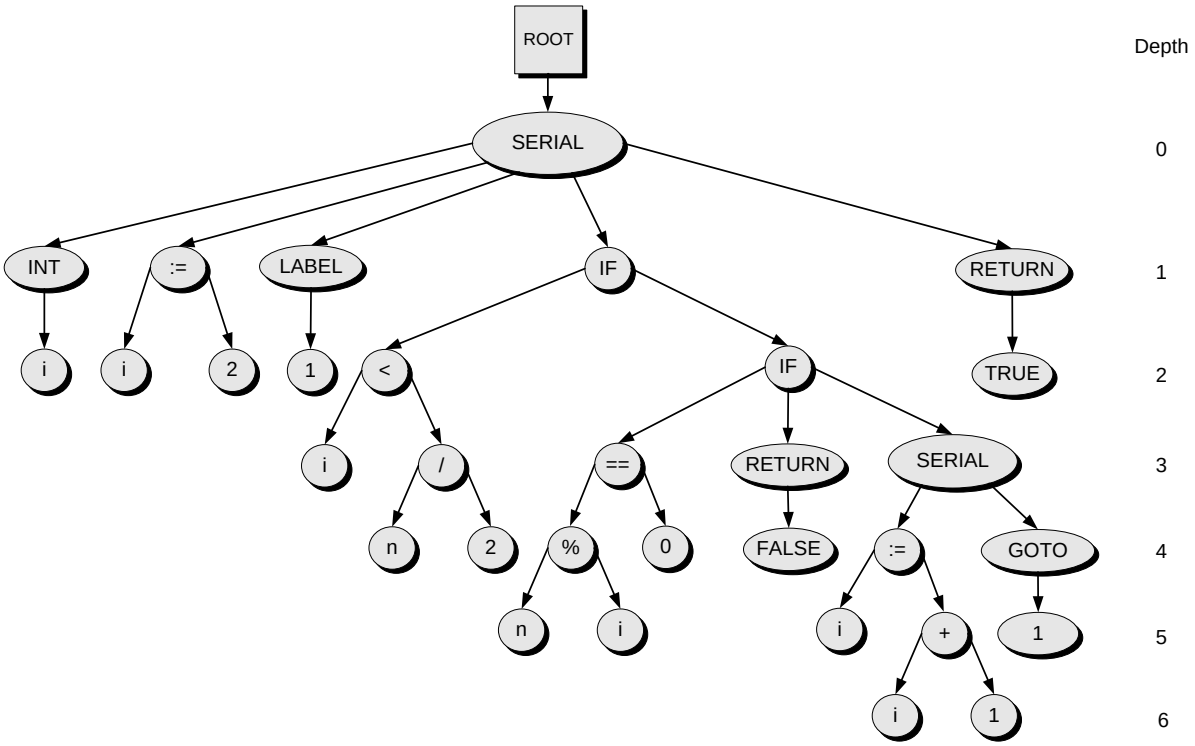
Έστω πως ο σαρωτής έχει καταναλώσει το (((INT i)(:= i 2) τότε το δέντρο που θα έχει δημιουργηθεί από το συντακτικό αναλυτή φαίνεται παραπάνω. Αν τώρα έχει

καταναλώσει μέχρι το (((INT i)(:= i 2)(LABEL 1)(IF (< i (/ n 2)) το δέντρο έχει την παρακάτω μορφή:



Εικόνα 7.4: Δέντρο από τη σάρωση του (((INT i)(:= i 2)(LABEL 1)(IF (< i (/ n 2))

Η τελική μορφή του συντακτικού δέντρου η παρακάτω:



Εικόνα 7.5: Δέντρο μετά την ολοκλήρωση της σάρωσης

Με την ολοκλήρωση του σχηματισμού του συντακτικού δέντρου, ξεκινά η χρήση του Διερμηνευτής. Ο Διερμηνευτής δέχεται ως είσοδο το συντακτικό δέντρο του

προγράμματος και μια ακέραια τιμή n ως όρισμα. Με την ολοκλήρωση της αποτίμησης, θα επιστρέψει αληθές ή ψευδές ανάλογα με το αν ο αριθμός n είναι ή όχι πρώτος. Η αποτίμηση του προγράμματος ξεκινά από τα αριστερά προς τα δεξιά και εξετάζεται κάθε υποδέντρο με bottom-up προσέγγιση. Το πρώτο υποδέντρο αφορά τη δήλωση της μεταβλητής i, ενώ το δεύτερο υποδέντρο την ανάθεση της τιμής δύο στη μεταβλητή i. Με την ολοκλήρωση της αποτίμησης αυτών των δύο υποδέντρων θα έχει εισαχθεί στον πίνακα συμβόλων του προγράμματος η μεταβλητή i και θα έχει τιμή δύο. Το τρίτο υποδέντρο με τύπο LABEL δεν εκτελείται σε αυτό το βήμα, αλλά είναι το σημείο στο οποίο θα μεταφερθεί ο έλεγχος του προγράμματος όταν εκτελεστεί κόμβος τύπου GOTO. Στη συνέχεια, αποτιμάται το υποδέντρο με ρίζα τον κόμβο IF. Το αριστερό παιδί του κόμβου αυτού αποτελεί τη συνθήκη ελέγχου, όπου ελέγχεται αν $i < n/2$. Αν είναι ψευδές, τότε το πρόγραμμα τερματίζει και επιστρέφει TRUE (κόμβος RETURN που σηματοδοτεί τον τερματισμό του προγράμματος). Διαφορετικά, αποτιμάται το δεύτερο IF, όπου εξετάζεται αν $i \bmod n$ είναι μηδέν. Αν είναι, τότε ο αριθμός n δεν είναι πρώτος οπότε στη συνέχεια το πρόγραμμα επιστρέφει FALSE, διαφορετικά, αυξάνεται η τιμή της μεταβλητής i και στη συνέχεια ο κόμβος GOTO μεταφέρει τον έλεγχο εκτέλεσης του προγράμματος στον κόμβο LABEL, ώστε να συνεχιστεί από εκεί η επαναληπτική διαδικασία.

ΚΕΦΑΛΑΙΟ 8
ΠΕΡΙΓΡΑΦΗ ΕΦΑΡΜΟΓΗΣ ΓΕΝΕΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Στα προηγούμενα κεφάλαια παρουσιάστηκαν στοιχεία από το πεδίο του γενετικού προγραμματισμού και τα πρωτόκολλα Chord και S-Chord για δίκτυα ομότιμων κόμβων. Στο κεφάλαιο αυτό θα παρουσιαστεί μια εφαρμογή των γενετικών αλγορίθμων σε δίκτυα ομότιμων κόμβων (P2P).

8.1 Ορισμός Προβλήματος

Ένα από τα βασικά θέματα στα δίκτυα ομότιμων κόμβων είναι η αναζήτηση και ανάκτηση δεδομένων από το δίκτυο. Δηλαδή, ο αποτελεσματικός εντοπισμός του κόμβου που έχει αποθηκεύσει κάποιο συγκεκριμένο δεδομένο. Στην εργασία αυτή προτείνουμε έναν ευρετικό μηχανισμό, παραγόμενο με τις αρχές του γενετικού προγραμματισμού, ο οποίος προστίθεται και λειτουργεί επικουρικά στους μηχανισμούς αναζήτησης κλειδιών, των συγγενικών P2P πρωτόκολλων Chord και S-Chord.

Η βασική ιδέα του ευρετικού μηχανισμού είναι πως δοθέντος ενός κλειδιού επιστρέφει έναν κόμβο όσο γίνεται πιο κοντά στον κόμβο που έχει αποθηκευμένο το δεδομένο που αναζητούμε. Ο προτεινόμενος μηχανισμός χρησιμοποιεί την οργάνωση που υποθέτει η αρχιτεκτονική του πρωτοκόλλου Chord (και S-Chord), δεν εισάγει χαρακτηριστικά που μεταβάλλουν τη φύση του πρωτοκόλλου και είναι σχεδιασμένος ώστε να αξιοποιεί την πληροφορία δρομολόγησης (πίνακες δρομολόγησης/δεικτών) του πρωτοκόλλου (κατά την εφαρμογή του χρησιμοποιεί τους ίδιους πίνακες δρομολόγησης). Για το μηχανισμό αυτό έχει υλοποιηθεί μια πειραματική εφαρμογή που ενσωματώνει το μηχανισμό, στο μηχανισμό αναζήτησης κλειδιών του Chord (και S-Chord).

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, τόσο το Chord όσο και το S-Chord οργανώνονται σε έναν εικονικό δακτύλιο (Chord ring). Κάθε δεδομένο κατακερματίζεται σε ένα κλειδί και κάθε κόμβος στο Chord περιγράφεται από ένα m-bit προσδιοριστή, ο οποίος προκύπτει από κατακερματισμό ενός χαρακτηριστικού του κόμβου (π.χ. URL). Τόσο οι προσδιοριστές των κόμβων όσο και κλειδιά των δεδομένων καταλαμβάνουν τον ίδιο εικονικό χώρο τιμών στο πρόβλημα. Επιπλέον, ο φόρτος διαμοιράζεται μεταξύ των κόμβων με χρήση συνεπούς κατακερματισμού, με τα κλειδιά να τοποθετούνται στον ίδιο χώρο τιμών με τους προσδιοριστές των κόμβων. Κάθε κλειδί τοποθετείται στον κόμβο που έχει προσδιοριστή μεγαλύτερο και πιο κοντά στον δικό του προσδιοριστή στο χώρο κατακερματισμού. Έτσι κάθε κόμβος χειρίζεται ένα τμήμα του χώρου κατακερματισμού και είναι υπεύθυνος για ένα εύρος από κλειδιά. Τη σχέση εγγύτητας που παρουσιάζεται

ανάμεσα στους κόμβους και στα κλειδιά στο εικονικό δακτύλιο, επιδιώκουν να εκμεταλλευτούν τα γενετικά προγράμματα.

Στη συνέχεια, θα παρουσιαστούν τα τερματικά και συναρτησιακά σύμβολα από τα οποία παράγονται τα γενετικά προγράμματα, θα οριστεί η μετρική καταλληλότητας για το πρόβλημα, οι παράμετροι που επηρεάζουν την εκτέλεση, ο τρόπος λειτουργίας του γενετικού μηχανισμού και πως τροποποιείται η αναζήτησης στα προαναφερθέντα P2P πρωτόκολλα με την εφαρμογή του μηχανισμού. Σε επόμενο κεφάλαιο θα μελετηθεί η συμπεριφορά του μηχανισμού χρησιμοποιώντας διαφορετικούς αλγορίθμους επεξεργασίας των γενετικών προγραμμάτων και πως επηρεάζεται το κόστος αναζήτησης κλειδιών.

8.2 Επιλογή Τερματικών / Συναρτησιακών Συμβόλων

Ένα πολύ σημαντικό βήμα πριν τη χρήση των γενετικών προγραμμάτων είναι η επιλογή των κατάλληλων συναρτήσεων και μεταβλητών που επιτρέπουν την έκφραση του προβλήματος. Στην περίπτωση μας, το γενετικό πρόγραμμα επεξεργάζεται την τιμή ενός κλειδιού. Το κλειδί αυτό εκφράζεται με τη βοήθεια της ανεξάρτητης μεταβλητής X. Εκτός από τη μεταβλητή X, ως τερματικά σύμβολα χρησιμοποιούνται και σταθερές από πραγματικές τιμές μεταξύ -3 και 3. Έτσι, το σύνολο των τερματικών συμβόλων διαμορφώνεται ως εξής:

$$T = X, Y \in -3,3 : Y \in \Re$$

Για το πρόβλημα που μελετάμε, δε χρησιμοποιούνται όλες η δυνατότητες της γενετικής γλώσσας, γιατί κάτι τέτοιο θα προσέφερε μεγάλη ευελιξία στο γενετικό πρόγραμμα, μεγαλώνοντας σημαντικά το χώρο αναζήτησης. Έχουν επιλεγεί να χρησιμοποιηθούν μόνο απλές μαθηματικές συναρτήσεις όπως πρόσθεση, αφαίρεση, πολλαπλασιασμός και προστατευμένη διαίρεση (επιστρέφει 0 όταν ο διαιρέτης είναι μηδέν). Το σύνολο των συναρτησιακών συμβόλων που χρησιμοποιείται είναι το παρακάτω:

$$F = +, -, *, /$$

8.3 Μετρική Καταλληλότητας

Για το υπολογισμό της συνάρτησης καταλληλότητας, στο πρόβλημα που μελετάμε, απαιτείται η ύπαρξη πληροφορίας από προηγούμενες αναζητήσεις κλειδιών στο δίκτυο. Δηλαδή, χρησιμοποιείται ένα δείγμα εκπαίδευσης (training set), που αποτελείται ένα υποσύνολο των κλειδιών που υπάρχουν στο δίκτυο και τροφοδοτεί το γενετικό μοντέλο πρόβλεψης με δεδομένα από προηγούμενες επιτυχημένες αναζητήσεις (τους κόμβους που είναι υπεύθυνοι για τα κλειδιά). Οι τιμές αυτές αποτελούν τη βάση για τη γενίκευση

των αποτελεσμάτων σε όλο το πεδίο ορισμού. Η ανάγκη για δείγμα εκπαίδευσης, προϋποθέτει την ύπαρξη ενός διαστήματος συλλογής δεδομένων πριν την ενεργοποίηση του γενετικού μηχανισμού.

Γενικά, οι αλγόριθμοι εκπαίδευσης βελτιώνουν την ακρίβεια τους όσο τροφοδοτούνται με περισσότερα δεδομένα. Αν τα δεδομένα για το δείγμα εκπαίδευσης αρκούν για τη στατιστική αναπαράσταση του χώρου του προβλήματος τότε η διαδικασία εκπαίδευσης τερματίζει. Στο πλαίσιο αυτής της εργασίας δεν μελετάται κάποιος αλγόριθμος για τη συλλογή του δείγματος εκπαίδευσης ούτε και ποιο είναι το πλήθος των δεδομένων που πρέπει να έχει στη διάθεση του ο γενετικός αλγόριθμος ώστε να αποδίδει βέλτιστα (αν και στο κεφάλαιο των αποτελεσμάτων των προσομοιώσεων θα παρουσιαστεί ένα παράδειγμα υπολογισμού καταλληλότητας με διαφορετικά μεγέθη δειγμάτων εκπαίδευσης). Γενικά, ο αλγόριθμος τροφοδοτείται με ένα ομοιόμορφα κατανεμημένο υποσύνολο των κλειδιών που υπάρχουν στο δίκτυο (π.χ. 10% ή 20%), τη στιγμή ενεργοποίησης του γενετικού μηχανισμού. Το μέγεθος του δείγματος εκπαίδευσης αποτελεί μια από τις παραμέτρους του συστήματος.

Το γενετικό πρόγραμμα δέχεται ως είσοδο ένα κλειδί και επιστρέφει τον κόμβο που έχει το κλειδί αυτό. Στο Chord (και S-Chord), ο χώρος των προσδιοριστών είναι ένα υποσύνολο των φυσικών αριθμών N, και η απόσταση ανάμεσα σε δύο κόμβους x και y ορίζεται στο χώρο των κόμβων/κλειδιών του δακτυλίου και δίνεται από μια σχέση της μορφής $d(x,y) = |y - x| \bmod N$. Χρησιμοποιώντας μια τέτοια σχέση για την εγγύτητα δύο κόμβων, υπολογίζουμε τη καταλληλότητα για μια τιμή από το δείγμα εκπαίδευσης, συγκρίνοντας το σφάλμα της απόστασης του κόμβου που επέστρεψε το γενετικό πρόγραμμα σε σχέση με τον κόμβο που έχει το κλειδί.

Όπως είναι λογικό, όσο μικρότερο είναι το σφάλμα τόσο καλύτερο θεωρείται το γενετικό πρόγραμμα. Η καταλληλότητα (raw - $r(t)$) του ατόμου υπολογίζεται με ολόκληρο το δείγμα εκπαίδευσης και είναι το άθροισμα των αποστάσεων, για όλα τα δείγματα εκπαίδευσης και δίνεται από τη σχέση $r(t) = \sum_{j=1}^{M} d(S(j), C(j))$, όπου $d(x,y)$ είναι η απόσταση ανάμεσα σε δύο κόμβους x και y όπως ορίστηκε παραπάνω, $S(j)$ είναι ο προσδιοριστής κόμβου που επιστρέφει το γενετικό πρόγραμμα για ένα κλειδί j, $C(i,j)$ ο προσδιοριστής κόμβου στο δίκτυο για το ίδιο κλειδί και M είναι το μέγεθος του δείγματος εκπαίδευσης. Ο ορισμός της συνάρτηση καταλληλότητας, με βάση το άθροισμα των αποστάσεων, εισάγει ακρίβεια στον υπολογισμό και αρκετά επίπεδα

διακριτότητας, ώστε να εντοπίζονται πιο εύκολα τα προγράμματα που είναι πιο κοντά στο στόχο. Επιπλέον, ο υπολογισμός της είναι σχετικά απλός, δεδομένου πως τόσο οι τιμές στο δείγμα εκπαίδευσης όσο και η έξοδο από το γενετικό πρόγραμμα είναι αριθμητικές τιμές.

8.4 Παράμετροι Εκτέλεσης

Γενικά, υπάρχουν πολλές παράμετροι που πρέπει να ληφθούν υπόψη κατά την εκτέλεση του γενετικού μηχανισμού. Τέτοιες είναι το μέγεθος του πληθυσμού, το πλήθος των γενεών εξέλιξης, ο αλγόριθμος αρχικοποίησης των γενετικών προγραμμάτων, το μέγιστο βάθος των αρχικών προγραμμάτων, το μέγιστο μέγεθος των προγραμμάτων κ.ά. Στην παράγραφο αυτή θα αναφερθούμε στις σημαντικότερες από αυτές, αλλά και ποιες τιμές έχουμε επιλέξει στην προσομοίωση για τις παραμέτρους αυτές.

Το μέγεθος του πληθυσμού (Population Size) εκφράζει το πλήθος των γενετικών προγραμμάτων που υπάρχουν σε κάθε γενιά. Μεγάλο πλήθος προγραμμάτων καθυστερεί τον υπολογισμό της επόμενης γενιάς, ενώ μικρή τιμή μπορεί να οδηγήσει σε έλλειψη ποικιλίας στον πληθυσμό και πρόωρη σύγκλιση. Στις προσομοιώσεις, για τον υπολογισμό του γενετικού μηχανισμού, θεωρούμε το πλήθος των γενετικών προγραμμάτων ανάλογα με το πλήθος των κόμβων που συμμετέχουν στο δίκτυο. Μια τέτοια θεώρηση μας επιτρέπει να καταβάλουμε υπολογιστική προσπάθεια ανάλογη με το μέγεθος του δικτύου για το οποίο προορίζεται να χρησιμοποιηθεί ο μηχανισμός. Το πλήθος των γενετικών προγραμμάτων που χρησιμοποιούνται δε ξεπερνά τα 500.

Το μέγιστο πλήθος των γενεών (Maximum Generations) καθορίζει τον αριθμό των γενεών που θα τρέξει ο αλγόριθμος. Πέρα από αυτή την τιμή θεωρούμε πως έχει επέλθει σύγκλιση και σταματάει η εκτέλεση του αλγορίθμου. Συνήθως, μεγάλη βελτίωση παρατηρείται στις πρώτες γενιές, ενώ για επιπλέον βελτίωση χρειάζονται περισσότερες γενιές. Οι τιμές που έχουμε επιλέξει για τις προσομοιώσεις κυμαίνονται από 50 έως 200 γενιές. Η προσομοίωση είναι δυνατόν να ολοκληρωθεί πριν συμπληρωθεί ο αριθμός γενεών που είχαμε ορίσει αρχικά. Αυτό συμβαίνει σε δύο περιπτώσεις, είτε αν βρεθεί η λύση είτε αν όλα τα άτομα συγκλίνουν στην ίδια τιμή. Σε κάθε περίπτωση, στην παρουσίαση των αποτελεσμάτων αναφέρετε κάθε φορά το πλήθος των γενεών που έτρεξε η προσομοίωση.

Σε προηγούμενο κεφάλαιο αναφερθήκαμε στους αλγορίθμους αρχικοποίησης των γενετικών προγραμμάτων. Στις προσομοιώσεις έχει επιλεγεί η μέθοδος ramped half-and-half, όπου ο πληθυσμός χωρίζεται ομοιόμορφα σε άτομα που έχουν x-1, x-2, x-3, x-4,… βάθος και για κάθε ομάδα με διαφορετικό βάθος, τα μισά δέντρα αρχικοποιούνται

με την πλήρη μέθοδο και τα μισά με τη μέθοδο εξέλιξης. Ο αλγόριθμος αρχικοποίησης γενετικών προγραμμάτων συνδυάζεται και με την παράμετρο που προσδιορίζει το μέγιστο βάθος των νέων προγραμμάτων (τιμή 5 στην προσομοίωση). Αυτές οι παράμετροι επηρεάζουν την ποικιλία που θα έχει ο αρχικός πληθυσμός και τη διαφοροποίηση στην τιμή καταλληλότητας για τα αρχικά τυχαία δημιουργημένα άτομα. Ένα παράδειγμα αρχικού προγράμματος είναι το `((RETURN (+ (* X (+ -2.875 X )) -2.747)))`.

Το μέγεθος των ατόμων του πληθυσμού τείνει να μεγαλώνει όσο περνάνε οι γενιές με την εφαρμογή των γενετικών τελεστών. Για την προσομοίωση ορίζεται μέγιστη τιμή στο βάθος που μπορεί να φτάσει ένα πρόγραμμα (τιμή 250 για τις προσομοιώσεις). Ο λόγος που εισάγεται αυτή η παράμετρος είναι ώστε να αποφευχθούν τα πολύ μεγάλα δέντρα που καταναλώνουν σημαντικούς υπολογιστικούς πόρους.

Σημαντική παράμετρο στις προσομοιώσεις αποτελεί το ποσοστό των ατόμων του πληθυσμού στα οποία εφαρμόζεται κάθε γενετικός τελεστής για να παραχθούν τα παιδιά της επόμενης γενιάς. Στην περίπτωση μας έχουμε επιλέξει 90% για τον τελεστή διασταύρωσης και από 5% για τους τελεστές αναπαραγωγής και μετάλλαξης. Για την επιλογή των ατόμων που θα διασταυρωθούν (crossover) για να παραχθεί η επόμενη γενιά, χρησιμοποιούνται οι αλγόριθμοι επιλογής: επιλογή με πρωτάθλημα (tournament selection), επιλογή με αποκοπή (truncation selection), γραμμική και εκθετική επιλογή με διαβάθμιση (linear και exponential selection ranking) και επιλογή ανάλογα με την τιμή της συνάρτησης καταλληλότητας (fitness proportional selection). Για την επιλογή των ατόμων που θα εισαχθούν στον πληθυσμό της επόμενης γενιάς χρησιμοποιούνται, στην ανάπτυξη του γενετικού μηχανισμού, οι αλγόριθμοι: Καθαρή Επαναεισαγωγή (Pure Reinsertion) με απογόνους όσους οι γονείς, Ομοιόμορφη Επαναεισαγωγή (Uniform Reinsertion) με απογόνους το 75% των γονέων, Ελιτιστική Επαναεισαγωγή (Elitist Reinsertion) με απογόνους το 75% των γονέων και με βάση την καταλληλότητα Επαναεισαγωγή (Fitness-based Reinsertion) με απογόνους διπλάσιους από τους γονείς. Περισσότερα στοιχεία για τους γενετικούς τελεστές, τους αλγορίθμους επιλογής και του αλγορίθμους επαναεισαγωγής παρουσιάστηκαν σε προηγούμενο κεφάλαιο.

Στον παρακάτω πίνακα συνοψίζονται οι παράμετροι μαζί με ενδεικτικές τιμές που επιλέγουμε στις προσομοιώσεις.

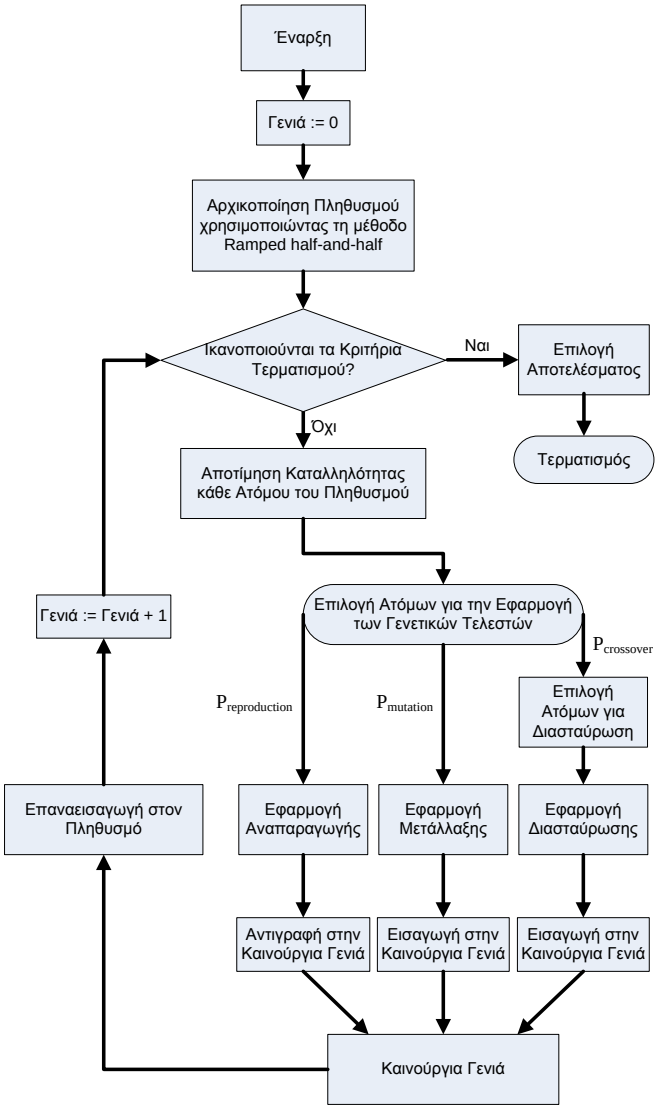
Πίνακας 8.1: Παράμετροι Προσομοίωσης

Παράμετροι Προσομοίωσης	
Μέγεθος Πληθυσμού	Όσο το πλήθος των κόμβων του δικτύου που προσομοιώνεται
Μέγιστο Πλήθος Γενεών	Από 50 έως 100 γενιές
Τερματικά Σύμβολα	$T = X, Y \in -3,3 : Y \in \Re$
Συναρτησιακά Σύμβολα	$F = +, -, *, /$
Συνάρτηση Καταλληλότητας	Ακατέργαστη Καταλληλότητα με βάση την απόσταση του μηχανισμού από τον κανονικό κόμβο, στο χώρο του εικονικού δακτυλίου
Μέγεθος Δείγματος Εκπαίδευσης	10% των κλειδιών του δικτύου
Αλγόριθμος Αρχικοποίησης	Ramped half-and-half
Μέγιστο Αρχικό Βάθος	5
Μέγιστο Βάθος Προγραμμάτων	250
Εφαρμογή Τελεστών	Διασταύρωση: 90% Μετάλλαξη: 5% Αναπαραγωγή: 5%
Αλγόριθμοι Επιλογής	επιλογή με πρωτάθλημα, επιλογή με αποκοπή, γραμμική επιλογή με διαβάθμιση, επιλογή ανάλογα με την τιμή της συνάρτησης καταλληλότητας
Αλγόριθμοι Επαναεισαγωγής	Καθαρή, Ομοιόμορφη, Ελιτιστική και με βάση την καταλληλότητα Επαναεισαγωγή

8.5 Προσομοίωση Γενετικού Μηχανισμού

Στην παράγραφο αυτή θα πραγματοποιηθεί μια συνολική περιγραφή της διαδικασίας που ακολουθείται για τη δημιουργία του γενετικού μηχανισμού.

Αρχικά, δημιουργείται ένας τυχαίος πληθυσμός από άτομα με τη μέθοδο ramped half-and-half. Κάθε πρόγραμμα έχει δεντρική δομή. Ως ρίζα του επιλέγεται ένα συναρτησιακό σύμβολο και στη συνέχεια τα ορίσματα του λαμβάνουν τιμές με βάση τα δομικά στοιχεία της γλώσσας που έχουμε επιλέξει μέχρι το επιλεγμένο μέγιστο βάθος για τα αρχικά προγράμματα. Τα φύλλα του δέντρου λαμβάνουν τιμές από το σύνολο των τερματικών και των συναρτησιακών συμβόλων. Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε νέο πρόγραμμα του πληθυσμού. Με την ολοκλήρωση της διαδικασίας για κάθε ένα από τα προγράμματα αυτά αποτιμάται η συνάρτησης καταλληλότητας.



Εικόνα 8.1: Διάγραμμα Ροής Γενετικού Μηχανισμού

Στο επόμενο βήμα, πραγματοποιείται ένας διαχωρισμός των ατόμων στα οποία θα εφαρμοστούν οι γενετικοί τελεστές (στο 90% των ατόμων θα εφαρμοστεί διασταύρωση[6], στο 5% μετάλλαξη και στο 5% αναπαραγωγή). Από το σύνολο των ατόμων που έχουν διαχωριστεί για διασταύρωση, επιλέγονται με τη βοήθεια των αλγορίθμων επιλογής τα άτομα που τελικά θα συμμετέχουν στη διαδικασία διασταύρωσης (ενδιάμεσος πληθυσμός). Με την εφαρμογή των γενετικών τελεστών στα επιλεγμένα άτομα του αρχικού πληθυσμού, σχηματίζεται μια νέα γενιά προγραμμάτων που ευελπιστούμε να έχει καταλληλότερα άτομα από την προηγούμενη. Μετά την ολοκλήρωση της παραγωγής νέων ατόμων, πραγματοποιείται επιλογή των ατόμων που θα εισαχθούν στον πληθυσμό της επόμενης γενιάς και των ατόμων που θα αντικατασταθούν από

[6] Τα σημεία διασταύρωσης επιλέγονται ανάλογα με το βάθος κάθε σημείου, δίνοντας μεγαλύτερη πιθανότητα επιλογής στα σημεία που είναι πιο κοντά στα φύλλα.

απογόνους. Η επιλογή αυτή πραγματοποιείται με κάποιο από τους αλγόριθμους επαναεισαγωγής που αναφέρθηκαν παραπάνω και οδηγεί στη δημιουργία του πληθυσμού της επόμενης γενιάς. Για το νέο πληθυσμό μπορεί να υπολογιστεί τώρα η συνάρτηση καταλληλότητας.

Η διαδικασία που παρουσιάστηκε παραπάνω, επαναλαμβάνεται είτε μέχρι να εξαντληθούν οι γενιές προσομοίωσης είτε μέχρι να ισχύει κάποιο από τα κριτήρια πρόωρου τερματισμού. Με την ολοκλήρωση της διαδικασίας, το άτομο με την καλύτερη τιμή καταλληλότητας επιλέγεται για να χρησιμοποιηθεί στο γενετικό μηχανισμό. Το άτομο αυτό γίνεται γνωστό σε όλους τους κόμβους που πρόκειται να χρησιμοποιήσουν το μηχανισμό.

8.6 Δίκτυο και Γενετικός Μηχανισμός

Τα δίκτυα ομότιμων κόμβων μεταβάλλονται δυναμικά κατά τη διάρκεια της ζωής τους, παρουσιάζοντας αλλαγές στη μορφή τους (αφίξεις και αναχωρήσεις κόμβων, προσθήκη κλειδιών, επανέκδοση κλειδιών κ.ά.). Οι μεταβολές αυτές είναι λογικό να επηρεάζουν την ικανότητα της γενετικής συνάρτησης, δεδομένου πως έχει υπολογιστεί για κάποιο στιγμιότυπο του δικτύου (συγκεκριμένοι κόμβοι, συγκεκριμένη κατανομή κλειδιών). Όμως, ο υπολογισμός της δεν πρέπει να πραγματοποιείται πολύ συχνά, αφού κάτι τέτοιο απαιτεί τόσο επικοινωνία μεταξύ των κόμβων όσο και υπολογιστικό κόστος. Πριν αποφασίσουμε πόσο συχνά πρέπει να υπολογίζεται ξανά η γενετική συνάρτηση πρέπει να λάβουμε υπόψη πως τα κλειδιά κατανέμονται στο δακτύλιο με συνεπή κατακερματισμό. Έτσι, για ένα δίκτυο με N κόμβους και K κλειδιά (με μεγάλη πιθανότητα) όταν ένας κόμβος συνδέεται ή αποχωρεί από το δίκτυο, $O(K/N)$ κλειδιά μετατοπίζονται προς ή από τον κόμβο που συνδέεται ή αποχωρεί, αντίστοιχα. Επιπλέον, κατά τις αναχωρήσεις και αφίξεις κόμβων τα κλειδιά μετατοπίζονται προς γειτονικούς κόμβους στο δακτύλιο. Λαμβάνοντας υπόψη τις δύο παραπάνω παρατηρήσεις και δεδομένου πως η γενετική συνάρτηση μας κατευθύνει προς μια περιοχή αναζήτησης γίνεται κατανοητό πως εφόσον τα κλειδιά παραμένουν στην ίδια περιοχή του δακτυλίου η γενετική συνάρτηση μπορεί να συνεχίσει να υποστηρίζει τις αναζητήσεις.

Πόσο συχνά πρέπει να συμβεί ο υπολογισμός της συνάρτησης, εξαρτάται από το πόσο ριζικές είναι οι αλλαγές στο δίκτυο και πόσο επηρεάζεται η ικανότητα της γενετικής συνάρτησης. Όπως θα αναφερθεί στην περιγραφή των αλγορίθμων αναζήτησης, αν η γενετική συνάρτηση καθοδηγεί εσφαλμένα την αναζήτηση για κάποιο κλειδί τότε η αναζήτηση παύει να χρησιμοποιεί το γενετικό μηχανισμό και χρησιμοποιεί τον κλασικό

αλγόριθμο αναζήτησης. Έτσι, ακόμα και αν μεταβληθεί αισθητά το δίκτυο, οι αναζητήσεις δε θα επηρεαστούν σημαντικά. Ο υπολογισμός της συνάρτησης, μπορεί να πραγματοποιείται είτε σε κάποιο συγκεκριμένο χρονικό διάστημα είτε αν εντοπίζεται πως σε μεγάλο ποσοστό των αναζητήσεων καθοδηγεί λανθασμένα.

Ο υπολογισμός της γενετικής συνάρτησης απαιτεί τη χρήση ενός δείγματος εκπαίδευσης. Τα δεδομένα του δείγματος εκπαίδευσης συλλέγονται από τους κόμβους κατά τη λειτουργία του δικτύου. Στην εκκίνηση του υπολογισμού του αλγορίθμου επιλέγεται ένας κόμβος από το δίκτυο για να πραγματοποιήσει τον υπολογισμό της συνάρτησης. Ο κόμβος αυτός επιλέγεται με κάποιο αλγόριθμο εκλογής (election algorithm π.χ. ring-based, bully). Στη συνέχεια, συλλέγει τα δεδομένα από τους κόμβους που παρέχουν στοιχεία για το δείγμα εκπαίδευσης, υπολογίζει την καινούργια γενετική συνάρτηση και την προωθεί στους υπόλοιπους κόμβους. Εναλλακτικά, ο υπολογισμός της συνάρτησης θα μπορούσε να πραγματοποιηθεί με κάποιον παράλληλο αλγόριθμο, δεδομένου πως η έννοια του πληθυσμού στους γενετικούς αλγορίθμους επιτρέπει τη σχετικά απλή παραλληλοποίηση του υπολογισμού.

Όπως φαίνεται από τα παραπάνω, ο γενετικός μηχανισμός δε χρειάζεται να επεμβαίνει στη βασική δομή του Chord (και S-Chord). Τροποποιεί εν μέρει τον αλγόριθμο αναζήτησης και συνεργάζεται με το υπάρχον σύστημα χρησιμοποιώντας πληροφορίες του συστήματος για τη βελτιστοποίηση του γενετικού μηχανισμού.

8.7 Τροποποίηση Αλγορίθμων Αναζήτησης

Στη συνέχεια, θα παρουσιαστούν οι αλλαγές στους αλγορίθμους αναζήτησης με την εφαρμογή του γενετικού μηχανισμού. Οι περιπτώσεις του Chord και S-Chord μελετώνται χωριστά, αφού χρησιμοποιούν διαφορετικούς πίνακες δρομολόγησης. Για κάθε μια από τις περιπτώσεις δίνεται ένα παράδειγμα αναζήτησης κλειδιού του αλγορίθμου.

8.7.1 Αναζήτηση με πίνακες δρομολόγησης Chord

Στην παράγραφο αυτή, θα παρουσιαστεί πως τροποποιείται η αναζήτηση με την προσθήκη της πληροφορίας από το γενετικό πρόγραμμα. Κάθε κόμβος στην περίπτωση του κλασικού Chord έχει στη διάθεση του μια λίστα με δείκτες προς επόμενους κόμβους στο δακτύλιο, τους οποίους χρησιμοποιεί στη διαδικασία της αναζήτησης. Αν ο τρέχων κόμβος έχει το κλειδί το επιστρέφει, διαφορετικά βρίσκει στον πίνακα δεικτών τον κόμβο που είναι πιο κοντά και έχει τιμή μικρότερη από το κλειδί που

αναζητά και προωθεί σε αυτόν την επερώτηση, με τη διαδικασία να προωθείται μεταξύ των κόμβων μέχρι να βρεθεί ο κατάλληλος.

Στην τροποποίηση του Chord, όπου εφαρμόζουμε το γενετικό μηχανισμό, χρησιμοποιούμε πάλι τον ίδιο πίνακα δεικτών (δημιουργείται με τον ίδιο αλγόριθμο και έχει το ίδιο μέγεθος) αλλά μεταβάλλεται ο αλγόριθμος αναζήτησης. Όπως αναφέρθηκε παραπάνω, ο γενετικός μηχανισμός για κάθε κλειδί που αναζητούμε, επιστρέφει τον κόμβο που θεωρεί πως είναι υπεύθυνος για το κλειδί. Αυτό πραγματοποιείται με τη βοήθεια της γενετικής συνάρτησης. Η γενετική συνάρτηση υπολογίζει μια τιμή για κάποιο κλειδί και η τιμή αυτή κανονικοποιείται ώστε να ανήκει στο χώρο τιμών των κλειδιών/κόμβων. Το αποτέλεσμα που δίνει το γενετικό πρόγραμμα χρησιμοποιείται επικουρικά με την κλασική αναζήτηση. Κάτι τέτοιο επιλέγεται λόγω της ευρετικής φύσης του μηχανισμού, όπου υπάρχει πιθανότητα να μας καθοδηγήσει σε λάθος κόμβο. Η τροποποίηση στον κλασικό αλγόριθμο αναζήτησης μπορεί να εντοπίσει τις περιπτώσεις όπου ο αλγόριθμος μας καθοδηγεί λανθασμένα και να “επιστρέψει” στην κλασική αναζήτηση, αποφεύγοντας έτσι περιττά βήματα στο μονοπάτι αναζήτησης.

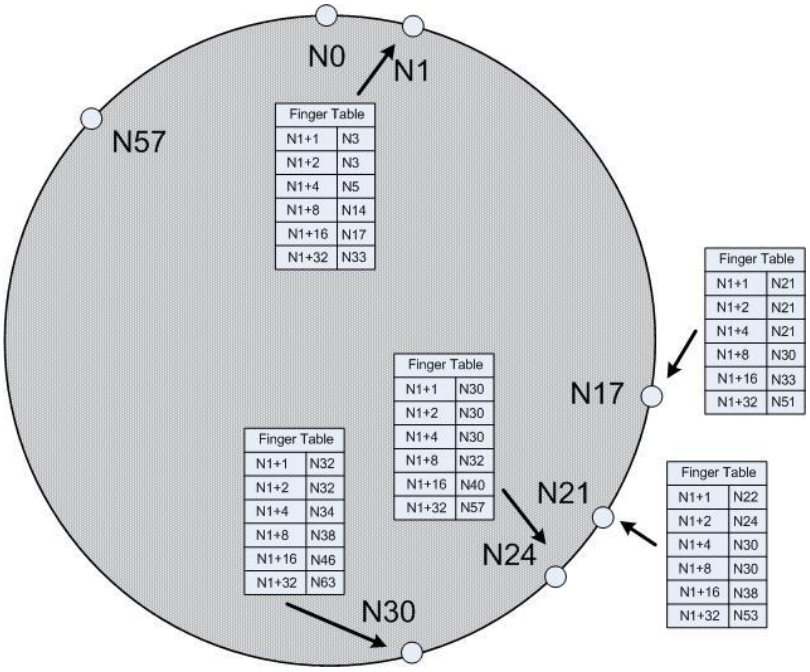
Αρχικά, αν ο τρέχων κόμβος έχει το κλειδί τότε απλά το επιστρέφει. Διαφορετικά, αναζητά στον πίνακα δεικτών τον κόμβο που είναι πιο κοντά στον κόμβο που έχει προτείνει το γενετικό πρόγραμμα. Όσο δεν εντοπίζεται το κλειδί, η επερώτηση προωθείται σε επόμενους κόμβους του δικτύου. Υπάρχει περίπτωση το γενετικό πρόγραμμα να μας καθοδηγήσει σε κόμβο όπου έχει τιμή μεγαλύτερη από το κλειδί, αλλά να μην έχει το κλειδί. Οπότε ο κόμβος που έχει το κλειδί βρίσκεται σε αντίθετη με τη φορά του ρολογιού κατεύθυνση από τον κόμβο που βρισκόμαστε στο δακτύλιο. Στην περίπτωση αυτή υπάρχουν δύο παραλλαγές του αλγορίθμου που μπορούμε αν ακολουθήσουμε. Στην πρώτη παραλλαγή επιστρέφουμε (rollback) στον προηγούμενο κόμβο (από τον οποίο βρεθήκαμε στον τρέχον) και συνεχίζουμε από εκεί χωρίς να χρησιμοποιήσουμε άλλο το γενετικό μηχανισμό αλλά χρησιμοποιώντας μόνο τους πίνακες δεικτών όπως στην κλασική περίπτωση του Chord. Στη δεύτερη παραλλαγή από τον τρέχον κόμβο που έχει ξεπεράσει το κλειδί που αναζητούμε, κινούμαστε ανάποδα από τη φορά του ρολογιού χρησιμοποιώντας το δείκτη προς τον προηγούμενο κόμβο[7] μέχρι να εντοπίσουμε τον κατάλληλο κόμβο. Και στις δύο περιπτώσεις, η αδυναμία του γενετικού μηχανισμού να μας κατευθύνει προς τον κατάλληλο κόμβο επιφέρει επιπλέον κόστος στην αναζήτηση. Στην πρώτη περίπτωση αναγκαζόμαστε να επισκεφτούμε τον ίδιο δύο φορές, ενώ στη δεύτερη περίπτωση

[7] Η παραλλαγή αυτή απαιτεί την ύπαρξη δείκτη από ένα κόμβο προς τον προηγούμενο κόμβο στην τοπολογία δακτυλίου.

ακολουθούμε μια σειριακή αναζήτηση με φορά αντίθετη των δεικτών του ρολογιού μέχρι να εντοπιστεί ο κόμβος που έχει το προς αναζήτηση κλειδί.

Στη συνέχεια παρουσιάζεται ένα παράδειγμα αναζήτησης σε ένα μικρό δίκτυο, όπου φαίνεται πως λειτουργούν οι δύο παραλλαγές του αλγορίθμου αμέσως μετά τον υπολογισμό της συνάρτησης καταλληλότητας. Το δίκτυο αποτελείται από 27 κόμβους και είναι αποθηκευμένα σε αυτό 54 κλειδιά. Ο μέγιστος αριθμός κόμβων στο δίκτυο είναι $2^6 = 64$ κόμβοι και οι πίνακες δρομολόγησης περιέχουν 6 εγγραφές. Η συνάρτηση καταλληλότητας έχει προκύψει από 27 γενετικά προγράμματα και μετά από προσομοίωση 50 γενεών. Η τιμή της συνάρτησης καταλληλότητας για το καλύτερο γενετικό πρόγραμμα είναι 19,0 για το δείγμα εκπαίδευσης (30% του συνόλου των κλειδιών) και 113,0 για ολόκληρο το σύνολο κλειδιών.

Η αναζήτηση ξεκινά από τον κόμβο με προσδιοριστή N1 και αναζητά το κλειδί με τιμή K30. Οι πίνακες δρομολόγησης είναι ίδιοι και στις δύο περιπτώσεις, αυτό που αλλάζει είναι ο αλγόριθμος αναζήτησης.



Εικόνα 8.2: Δίκτυο Chord

Στην παραπάνω εικόνα απεικονίζεται ένα μέρος του εικονικού δακτυλίου για το δίκτυο του παραδείγματος. Για τους κόμβους που απεικονίζονται, στο σχήμα υπάρχουν και οι αντίστοιχοι πίνακες δρομολόγησης. Χρησιμοποιώντας τον κλασικό αλγόριθμο του Chord, αρχικά, από τον κόμβο N1, προωθούμε την ερώτηση για το κλειδί K30 στον κόμβο N17. Στη συνέχεια, από τον κόμβο N21, η ερώτηση προωθείται στον κόμβο N24 (προηγούμενο του N30) και από τον κόμβο N24 στον κόμβο N30, ο οποίος απαντά πως

διαθέτει τα δεδομένα που αντιστοιχούν στο κλειδί K30. Το μονοπάτι για την εύρεση του κλειδιού είναι $N1, N17, N21, N24, N30$. Ενσωματώνοντας τη λειτουργία του γενετικού μηχανισμού στο παράδειγμα αυτό διαφοροποιείται ο αλγόριθμος αναζήτησης. Ο γενετικός μηχανισμός έχει υπολογίσει το κλειδί K30 το έχει ο κόμβος N31, ενώ υπεύθυνος για αυτό είναι ο κόμβος N30. Ο τροποποιημένος αλγόριθμος θα ακολουθήσει το αποτέλεσμα που έδωσε ο γενετικός μηχανισμός και θα κατευθύνει την αναζήτηση προς τον κόμβο N31. Αρχικά, θα προωθήσει την επερώτηση στον κόμβο N17. Ο κόμβος N17 με βάση τον πίνακα δρομολόγησης του θα προωθήσει την ερώτηση στον κόμβο N30, που έχει και το κλειδί. Το μονοπάτι που ακολουθήθηκε είναι το $N1, N17, N30$. Συγκρίνοντας το αποτέλεσμα από τους δύο αλγορίθμους παρατηρούμε πως η τροποποίηση του αλγορίθμου χρειάστηκε δύο βήματα λιγότερα. Όπως θα αναφέρουμε και στη συνέχεια, παρατηρήθηκε πως ο γενετικός μηχανισμός επιτυγχάνει πολλές φορές αποφεύγει να επισκεφτεί τον προηγούμενο του κόμβου που έχει το κλειδί (στο παράδειγμα αυτό τον N24).

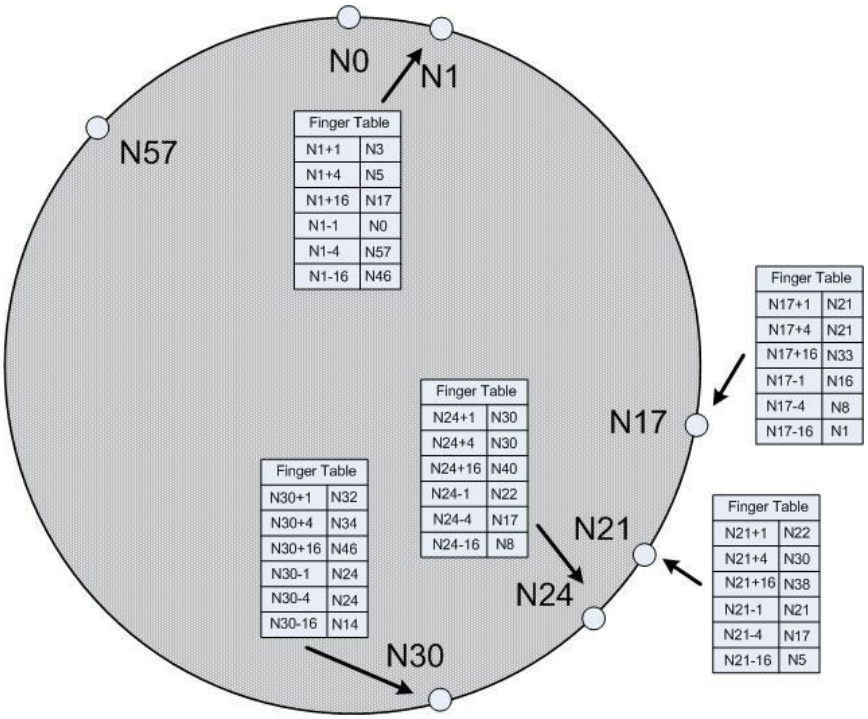
8.7.2 Αναζήτηση με πίνακες δρομολόγησης S-Chord

Στην παράγραφο αυτή, θα παρουσιαστεί πως εφαρμόζεται ο γενετικός μηχανισμός στην περίπτωση όπου έχουμε πίνακες δρομολόγησης αντίστοιχους με το S-Chord. Ο πίνακας δεικτών στην περίπτωση του S-Chord έχει δείκτες και προς προηγούμενους και προς επόμενους κόμβους στο δακτύλιο και έτσι μια αναζήτηση μπορεί να δρομολογηθεί και προς τις δύο κατευθύνσεις στον κυκλικό χώρο αναζήτησης, ανάλογα με το που βρίσκεται το κλειδί που αναζητούμε. Για την εύρεση ενός κλειδιού, αρχικά, ελέγχεται αν το κλειδί ανήκει στο διάστημα ανάμεσα στον κόμβο n και τον επόμενο του s (successor) ή στον προηγούμενο του p (predecessor) και στον κόμβο n. Οπότε επιστρέφεται ο επόμενος s στην μία περίπτωση και ο ίδιος ο n στην άλλη. Σε διαφορετική περίπτωση, πραγματοποιείται αναζήτηση στον πίνακα δεικτών του κόμβου και επιλέγεται ο πιο κοντινός δείκτης τον οποίο γνωρίζει ο n και σε αυτόν προωθείται η αναζήτηση. Ο αλγόριθμος αυτός συνεχίζεται μέχρι να εντοπιστεί το κλειδί ή μέχρι να αποτύχει η αναζήτηση.

Στην τροποποίηση του S-Chord, όπου εφαρμόζουμε το γενετικό μηχανισμό, χρησιμοποιείται πάλι ίδιος πίνακας δεικτών με δείκτες προς προηγούμενους και προς επόμενους κόμβους, αλλά μεταβάλλεται ο αλγόριθμος αναζήτησης. Αρχικά, αν ο τρέχων κόμβος έχει το κλειδί τότε απλά το επιστρέφει. Διαφορετικά, χρησιμοποιώντας τα διαστήματα ευθύνης που προκύπτουν από τον πίνακα δεικτών, επιλέγει την κατεύθυνση (προς τη φορά του ρολογιού ή αντίθετα από τη φορά του ρολογιού) που θα

κινηθεί, χρησιμοποιώντας το αποτέλεσμα που έχει προκύψει από το γενετικό μηχανισμό για τον εντοπισμό του προς αναζήτηση κλειδιού. Σε κάθε βήμα στόχος είναι να πλησιάζουμε πιο κοντά στον κόμβο που έχει προτείνει το γενετικό πρόγραμμα, επιλέγοντας τους κατάλληλους επόμενους κόμβους του δικτύου. Όπως και στην περίπτωση της παραλλαγής του Chord, είναι πιθανό ο γενετικός μηχανισμός να μην μπορέσει να μας κατευθύνει σωστά. Κάτι τέτοιο συμβαίνει, αν για παράδειγμα "κινούμαστε" με φορά αντίθετη από αυτή των δεικτών του ρολογιού, με βάση την τιμή που έχει δώσει ο γενετικός μηχανισμός και ξεπεράσουμε το προς αναζήτηση κλειδί. Σε μια τέτοια περίπτωση, θα πρέπει να αλλάξει η φορά αναζήτησης και να χρησιμοποιήσουμε τον αλγόριθμο αναζήτησης όπως στην περίπτωση του κλασικού αλγορίθμου αναζήτησης του S-Chord. Η αλλαγή αυτή στη φορά αναζήτησης εγγυάται την ορθή ολοκλήρωση της αναζήτησης (είτε με επιτυχία αν υπάρχει το κλειδί είτε με αποτυχία αν δεν υπάρχει το κλειδί) αλλά εισάγει ένα επιπλέον κόστος στην αναζήτηση, δεδομένου ότι μπορεί να χρειαστεί να επισκεφτούμε κάποιο κόμβο για δεύτερη φορά. Η συνολική απόδοση του γενετικού μηχανισμού πρέπει να είναι τέτοια ώστε να υπερκερνά τέτοιες περιπτώσεις.

Στη συνέχεια ακολουθεί μια αναζήτηση, στο ίδιο δίκτυο με το προηγούμενο παράδειγμα αλλά για το κλειδί K37 αυτή τη φορά, ξεκινώντας από τον κόμβο N57 χρησιμοποιώντας τον αλγόριθμο του S-Chord και την τροποποίηση του με το γενετικό μηχανισμό. Και στις δύο αναζητήσεις οι αλγόριθμοι διαθέτουν τον ίδιο πίνακα δρομολόγησης.



Εικόνα 8.3: Δίκτυο S-Chord

Στην παραπάνω εικόνα απεικονίζονται οι κόμβοι που συμμετέχουν στην αναζήτηση και οι πίνακες δρομολόγησης τους. Χρησιμοποιώντας τον αλγόριθμο αναζήτησης του S-Chord, αρχικά, από τον κόμβο N57, η ερώτηση προωθείται στον κόμβο N40, χρησιμοποιώντας τους δείκτες με ανάποδη φορά και τα διαστήματα ευθύνης κάθε δείκτη. Στη συνέχεια, από τον κόμβο N40, η ερώτηση προωθείται στον κόμβο N36 και από τον κόμβο N36 στον κόμβο N38, ο οποίος απαντά πως διαθέτει τα δεδομένα που αντιστοιχούν στο κλειδί K37. Το μονοπάτι για την εύρεση του κλειδιού είναι το $N57, N40, N36, N38$. Ο γενετικός μηχανισμός επιστρέφει τον κόμβο N38 ως υπεύθυνο για το κλειδί K37. Ο τροποποιημένος γενετικός μηχανισμός κατευθύνει την αναζήτηση προς τον κόμβο N38, που έχει το κλειδί. Αρχικά, από τον κόμβο N57, η ερώτηση προωθείται στον κόμβο N40, αφού στο διάστημα ευθύνης του ανήκει ο κόμβος N38. Στη συνέχεια, από τον κόμβο N40 (δείκτη N40-1) η ερώτηση φτάνει στον κόμβο N38 που έχει το κλειδί. Το μονοπάτι που ακολουθήθηκε είναι το $N57, N40, N38$. Συγκρίνοντας το μονοπάτι από τους δύο αλγορίθμους, παρατηρούμε πως το μονοπάτι που χρησιμοποιεί το γενετικό μηχανισμό επισκέπτεται ένα κόμβο λιγότερο για να εντοπίσει το κλειδί.

Στον παρακάτω πίνακα συνοψίζονται τα συνολικά αποτελέσματα (συνολικός και μέσος όρος βημάτων) από την αναζήτηση όλων των κλειδιών ξεκινώντας από κάθε κόμβο του δικτύου, χρησιμοποιώντας και τις τέσσερις παραλλαγές των αλγορίθμων αναζητήσεις. Συνολικά, πραγματοποιούνται 1458 αναζητήσεις κλειδιών (27 κόμβοι * 54 κλειδιά = 1458).

Πίνακας 8.2: Αποτελέσματα Προσομοιώσεων

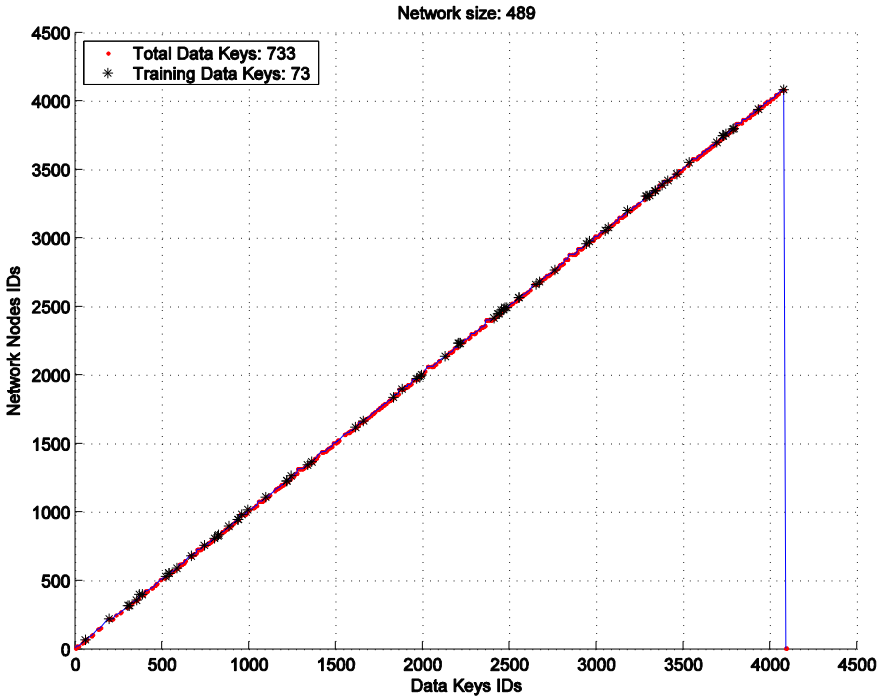
Αλγόριθμος	Πλήθος βημάτων	Μέσος όρος
Chord	4367	2.9951989026
Genetic Chord	3944	2.7050754458
S-Chord	3289	2.2558299039
Genetic S-Chord	3227	2.2133058984

ΚΕΦΑΛΑΙΟ 9
ΠΡΟΣΟΜΟΙΩΣΕΙΣ-ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό θα παρουσιαστούν τα αποτελέσματα από τις προσομοιώσεις που πραγματοποιήθηκαν για το σύστημα που περιγράφηκε στο προηγούμενο κεφάλαιο. Οι παράμετροι που μεταβάλλονται είναι το μέγεθος του δικτύου, το πλήθος των κλειδιών του δικτύου, το μέγεθος του δείγματος εκπαίδευσης και οι αλγόριθμοι επιλογής και επαναεισαγωγής. Στα αποτελέσματα θα αναλυθεί πως επηρεάζεται η συνάρτηση καταλληλότητας ανάλογα με τις παραπάνω παραμέτρους, αλλά και πως μεταβάλλεται το κόστος αναζήτησης στα Chord και S-Chord πρωτόκολλα με την προσθήκη του γενετικού μηχανισμού.

9.1 Προσομοίωση Συνάρτησης Καταλληλότητας

Στην παράγραφο αυτή θα παρουσιαστεί ένα παράδειγμα προσομοίωσης δικτύου δίνοντας έμφαση στον υπολογισμό της συνάρτησης καταλληλότητας.

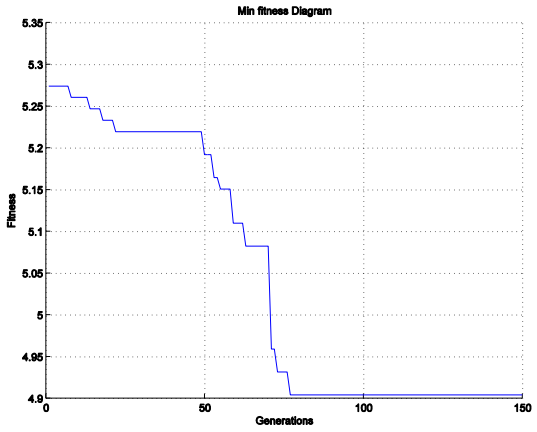


Εικόνα 9.1: Κατανομή Κλειδιών στους Κόμβους του Δικτύου Προσομοίωσης

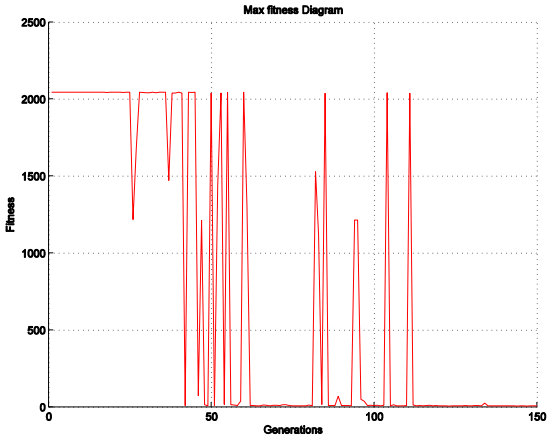
Στην προσομοίωση αυτή χρησιμοποιείται ένα δίκτυο αποτελούμενο από 489 κόμβους (το μέγιστο πλήθος κόμβων). Οι κόμβοι αυτοί είναι υπεύθυνοι για 733 κλειδιά. Το μέγεθος του δείγματος εκπαίδευσης είναι σταθερό και είναι το 10% του συνόλου των κλειδιών. Τα κλειδιά που απαρτίζουν το δείγμα εκπαίδευσης επιλέγονται ομοιόμορφα από το σύνολο των κλειδιών που υπάρχουν στο δίκτυο. Οι προσδιοριστές κόμβων και

κλειδιών παίρνουν τιμές στο $0, 4095$. Στις δοκιμές χρησιμοποιήθηκαν όλοι οι αλγόριθμοι επαναεισαγωγής και όλοι οι αλγόριθμοι επιλογής εκτός από την εκθετική επιλογή με διαβάθμιση. Το πλήθος γενεών προσομοίωσης είναι 150. Στην Εικόνα 9.1, τα κλειδιά που έχουμε εισάγει στο δίκτυο καθώς και το δείγμα εκπαίδευσης που αποτελείται από 73 κλειδιά. Με κόκκινες τελείες απεικονίζονται τα κλειδιά, ενώ με μαύρα αστέρια οι τιμές που αντιστοιχούν στο δείγμα εκπαίδευσης. Αξίζει να αναφέρουμε πως τα κλειδιά με προσδιοριστές 4092, 4094 είναι αποθηκευμένα στον κόμβο με προσδιοριστή 0 (τέλος του εικονικού δακτυλίου). Κάτι τέτοιο είναι πιθανό να δυσκολέψει τη γενετική συνάρτηση να προσδιορίσει σωστά τον υπεύθυνο κόμβο για αυτά.

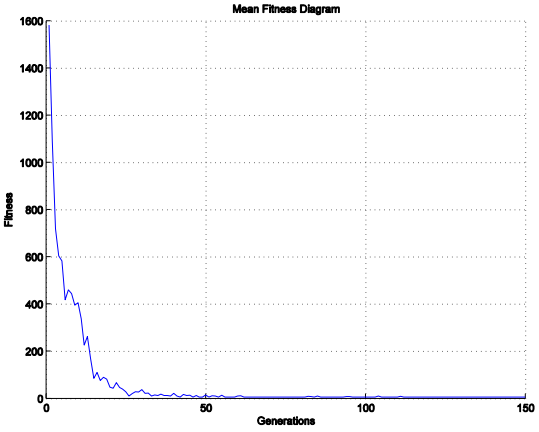
Στις παρακάτω εικόνες (9.2 έως 9.5), παρουσιάζονται στοιχεία που προκύπτουν κατά τον υπολογισμό της συνάρτησης καταλληλότητας. Για τις προσομοιώσεις αυτές, η μέθοδος επιλογής είναι το πρωτάθλημα (tournament) με μέγεθος 3 και η μέθοδος επαναεισαγωγής ο ελιτισμός (elitism).



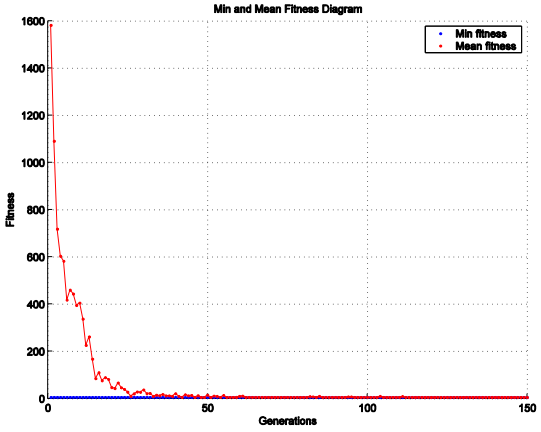
Εικόνα 9.2: Ελάχιστη τιμή Καταλληλότητας ανά γενιά



Εικόνα 9.3: Μέγιστη τιμή Καταλληλότητας ανά γενιά

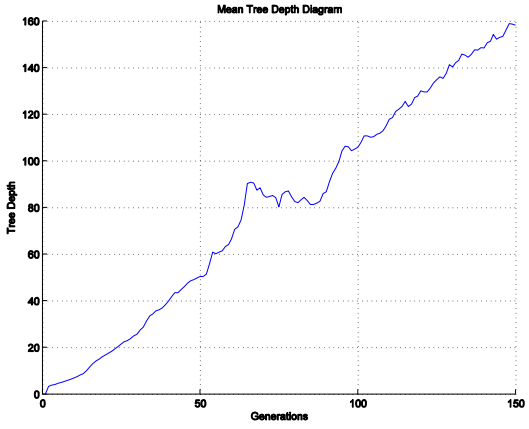


Εικόνα 9.4: Μέση τιμή Καταλληλότητας ανά γενιά

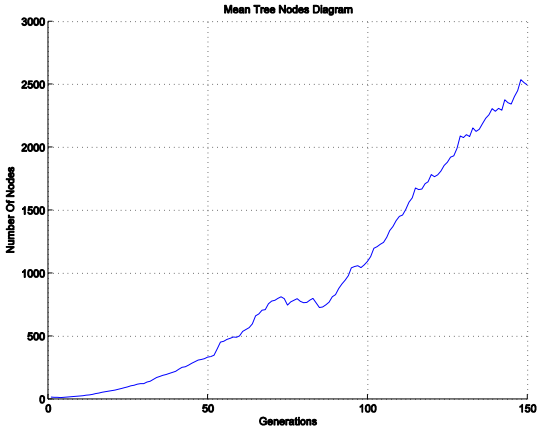


Εικόνα 9.5: Μέση και Ελάχιστη Τιμή Καταλληλότητας ανά γενιά

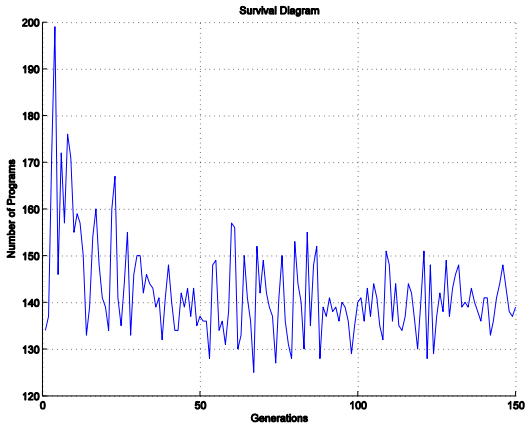
Στις εικόνες αυτές απεικονίζεται η ελάχιστη (Εικόνα 9.2), μέγιστη (Εικόνα 9.3) και μέση (Εικόνα 9.4) τιμή καταλληλότητας σε κάθε γενιά προσομοίωσης. Οι τιμές καταλληλότητας είναι κανονικοποιημένες ως προς το μέγεθος του δείγματος εκπαίδευσης, δηλαδή ως προς το πλήθος των κλειδιών με τα οποία έχει υπολογιστεί. Στο διάγραμμα της ελάχιστης καταλληλότητας παρατηρούμε πως η ελάχιστη τιμή σταθεροποιείται στην τιμή 4,9 στη γενιά 76 (στη μέση περίπου των γενεών προσομοίωσης). Η τιμή 4,9 εκφράζει τη μέση απόκλιση (απόσταση στο χώρο του δακτυλίου) ανάμεσα στην τιμή του προσδιοριστή που υπολογίζει η γενετική συνάρτηση για ένα κλειδί και στην πραγματική τιμή του προσδιοριστή του κόμβου. Στο διάγραμμα της ελάχιστης μαζί με τη μέση καταλληλότητα (Εικόνα 9.5) παρατηρούμε πως η μέση τιμή προσεγγίζει σταδιακά την ελάχιστη τιμή. Αυτό σημαίνει πως σταδιακά ο γενετικός πληθυσμός συγκλίνει προς κάποια τιμή.



Εικόνα 9.6: Μέσο Βάθος Γενετικών Προγραμμάτων



Εικόνα 9.7: Μέσο Πλήθος Κόμβων Γενετικών Προγραμμάτων

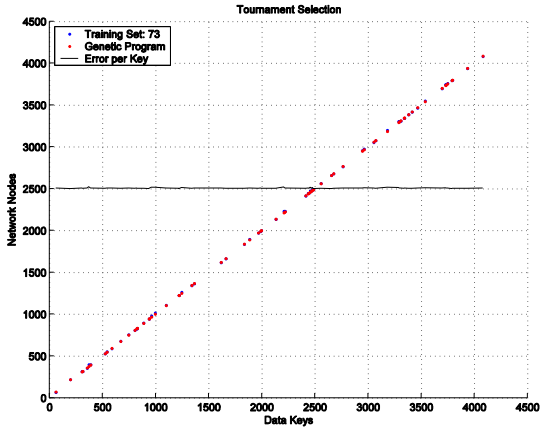


Εικόνα 9.8: Πλήθος Προγραμμάτων που επιβιώνουν

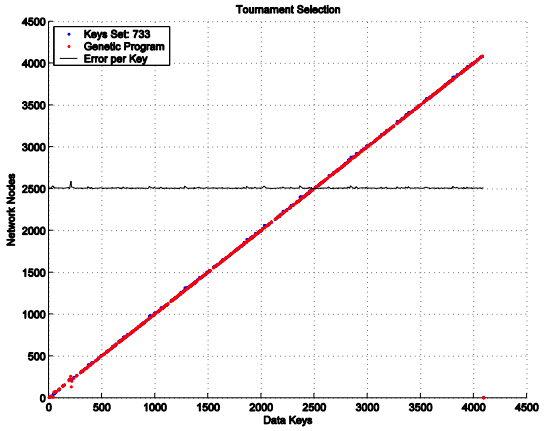
Στις παραπάνω εικόνες, απεικονίζονται ο μέσος αριθμός κόμβων (Εικόνα 9.7) και το μέσο βάθος (Εικόνα 9.6) των γενετικών προγραμμάτων ανά γενιά προσομοίωσης. Όπως είναι αναμενόμενο, τα μεγέθη αυτά μεγαλώνουν με το πέρασμα των γενεών.

Τέλος, απεικονίζεται και το πλήθος των προγραμμάτων τα οποία δεν επιβιώνουν από γενιά σε γενιά (Εικόνα 9.8).

Στα επόμενα δύο διαγράμματα απεικονίζεται το σφάλμα από τον υπολογισμό της γενετικής συνάρτησης. Στην Εικόνα 9.9 υπολογίζεται με βάση το δείγμα εκπαίδευσης, ενώ στην Εικόνα 9.10 με βάση το σύνολο των κλειδιών που υπάρχουν στους κόμβους. Οι μπλε κουκίδες είναι οι κόμβοι που έχουν τα κλειδιά και οι κόκκινες οι κόμβοι που επιστρέφει η γενετική συνάρτηση για τα κλειδιά. Η μαύρη γραμμή στον κατακόρυφο άξονα στην τιμή 2500, αντιστοιχεί στο σφάλμα στον υπολογισμό των κόμβων που έχουν τα κλειδιά. Παρατηρούμε πως το σφάλμα αυτό είναι πολύ μικρό σε σχέση με το εύρος τιμών που λαμβάνουν οι κόμβοι και τα κλειδιά. Το μεγαλύτερο σφάλμα εντοπίζεται στον υπολογισμό του κόμβου που έχει το κλειδί K212. Η γενετική συνάρτηση επιστρέφει τον κόμβο με προσδιοριστή N134, ενώ το κλειδί βρίσκεται στον κόμβο με προσδιοριστή N219.



Εικόνα 9.9: Σφάλμα στο Δείγμα Εκπαίδευσης



Εικόνα 9.10: Σφάλμα στο Σύνολο Κλειδιών

Στη συνέχεια, παρουσιάζεται η τιμή καταλληλότητας υπολογισμένη με συνδυασμό των αλγορίθμων επιλογής και των αλγορίθμων επαναεισαγωγής. Στον Πίνακας 9.1 πρώτο πίνακα παρατίθενται το αποτέλεσμα υπολογισμένο από τα κλειδιά του δείγματος εκπαίδευσης (εδώ 73 κλειδιά), ενώ στον Πίνακας 9.2 το αποτέλεσμα υπολογισμένο με το σύνολο των κλειδιών που υπάρχουν στο δίκτυο (εδώ 733 κλειδιά). Οι τιμές στο δεύτερο πίνακα αντικατοπτρίζουν την ικανότητα της συνάρτησης καταλληλότητας στο σύνολο του δικτύου. Και στις δύο περιπτώσεις οι τιμές καταλληλότητας είναι κανονικοποιημένες ως προς το πλήθος του δείγματος εκπαίδευσης και το πλήθος των κλειδιών του δικτύου αντίστοιχα. Ακόμα, υπολογίζεται και ο μέσος όρος καταλληλότητας ως προς τους αλγορίθμους επιλογής και ως προς τους αλγορίθμους επαναεισαγωγής.

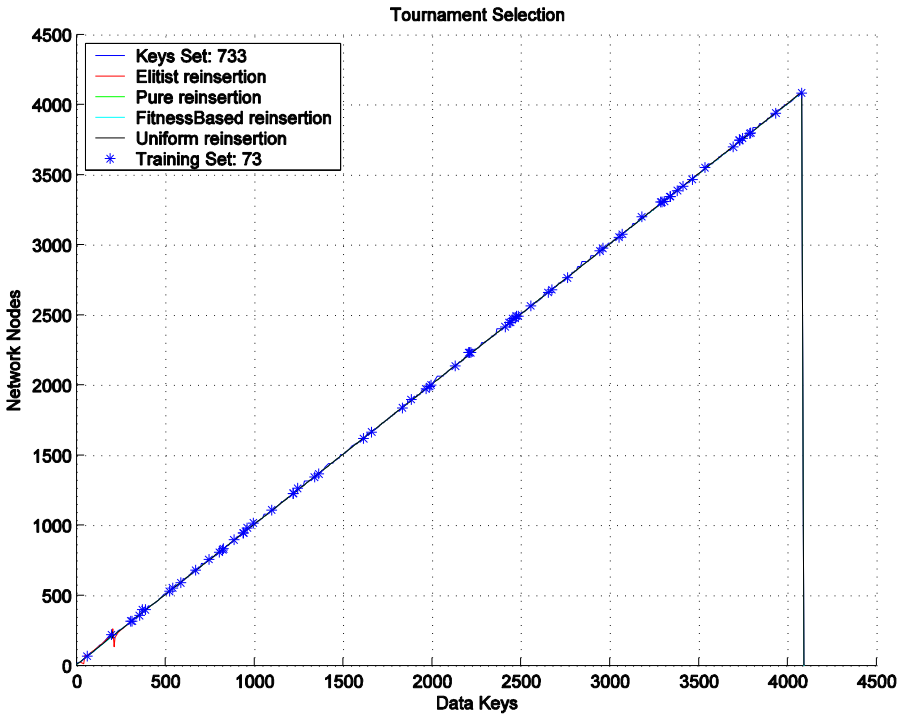
Στα αποτελέσματα που προκύπτουν, δεν υπάρχουν μεγάλες διαφορές ανάμεσα στις τιμές καταλληλότητας που υπολογίζονται με συνδυασμό των διαφορετικών αλγορίθμων επιλογής και επαναεισαγωγής.

Πίνακας 9.1: Καταλληλότητα ως προς το σύνολο του δείγματος εκπαίδευσης

Αλγόριθμος Επιλογής	Αλγόριθμος Επαναεισαγωγής				
	Ελιτιστική	Fitness-Based	Απόλυτη	Ομοιόμορφη	Μέσος Όρος
Γραμμική	4,86	5,12	5,22	5,27	5,12
Ανάλογη	4,84	4,51	5,34	5,27	4,99
Πρωτάθλημα	4,90	4,77	4,88	4,84	4,85
Αποκοπή	4,92	5,22	5,14	5,18	5,11
Μέσος Όρος	4,88	4,90	5,14	5,14	-

Πίνακας 9.2: Καταλληλότητα ως προς το σύνολο των κλειδιών

Αλγόριθμος Επιλογής	Αλγόριθμος Επαναεισαγωγής				
	Ελιτιστική	Fitness-Based	Απόλυτη	Ομοιόμορφη	Μέσος Όρος
Γραμμική	5,07	5,47	5,02	5,03	5,15
Ανάλογη	5,03	5,88	5,06	5,05	5,26
Πρωτάθλημα	5,32	5,01	5,03	5,04	5,10
Αποκοπή	5,68	5,02	5,02	5,07	5,20
Μέσος Όρος	5,27	5,35	5,03	5,05	-



Εικόνα 9.11: Σύγκριση Μεθόδων Επαναεισαγωγής

Στην Εικόνα 9.11**:** Σύγκριση Μεθόδων Επαναεισαγωγής απεικονίζεται η σύγκριση των αποτελεσμάτων των γενετικών συναρτήσεων που προκύπτουν χρησιμοποιώντας ως μέθοδο επιλογής το πρωτάθλημα και ως μεθόδους επαναεισαγωγής όλες τις μεθόδους που έχουν υλοποιηθεί. Οι συναρτήσεις παρόλο που υπολογίστηκαν με διαφορετικούς αλγορίθμους, δεν εμφανίζουν μεγάλες διαφοροποιήσεις.

Σημαντικό στατιστικό της προσομοίωσης αποτελεί η γενιά όπου ο γενετικός μηχανισμός συγκλίνει στην τελική τιμή ελάχιστης καταλληλότητας. Μια τέτοια μέτρηση είναι πιθανό να μας δώσει μια ένδειξη για το πλήθος των γενιών που πρέπει να τρέξει η προσομοίωση. Στον

Πίνακας 9.3: Γενιές Σύγκλισης είναι συγκεντρωμένες οι τιμές της γενιάς στην οποία συγκλίνει ο υπολογισμός της συνάρτησης καταλληλότητας ανάλογα με τον αλγόριθμο επιλογής και τον αλγόριθμο επαναεισαγωγής όπου επιλέγεται. Ενώ οι τιμές καταλληλότητας δεν παρουσιάζουν μεγάλη διαφοροποίηση με τη χρήση διαφορετικών αλγορίθμων επιλογής και επαναεισαγωγής, στην περίπτωση της γενιάς σύγκλισης παρατηρούνται διαφορές. Παρατηρούμε πως στη συγκεκριμένη προσομοίωση, η ελάχιστη τιμή καταλληλότητας σταθεροποιείται πιο νωρίς στην περίπτωση όπου αλγόριθμος επιλογής είναι το πρωτάθλημα (tournament) και ο αλγόριθμος επαναεισαγωγής είναι ο ανάλογος με την τιμή καταλληλότητας.

Πίνακας 9.3: Γενιές Σύγκλισης

Αλγόριθμος Επιλογής	Αλγόριθμος Επαναεισαγωγής				
	Ελιτιστική	Fitness-Based	Απόλυτη	Ομοιόμορφη	Μέσος Όρος
Γραμμική	99	112	147	76	108,5
Ανάλογη	124	92	149	149	128,5
Πρωτάθλημα	76	83	39	51	62,25
Αποκοπή	140	41	148	147	119
Μέσος Όρος	109	82	120,75	105,75	-

Στη συνέχεια παρουσιάζεται μια σύγκριση στην τιμή καταλληλότητας που υπολογίζεται σε σχέση με το δείγμα εκπαίδευσης που χρησιμοποιείται. Η μέθοδος επιλογής είναι και πάλι το πρωτάθλημα, ενώ χρησιμοποιούνται όλοι οι αλγόριθμοι για την επαναεισαγωγή. Το μέγεθος του δείγματος εκπαίδευσης είναι 24, 48, 73 και 97 κλειδιά και η τιμή καταλληλότητας είναι κανονικοποιημένη ως προς το μέγεθος του δείγματος εκπαίδευσης. Στον Πίνακας 9.4 παρατίθεται η καταλληλότητα υπολογισμένη με τα κλειδιά του δείγματος εκπαίδευσης, ενώ στον Πίνακας 9.5 με το σύνολο των κλειδιών

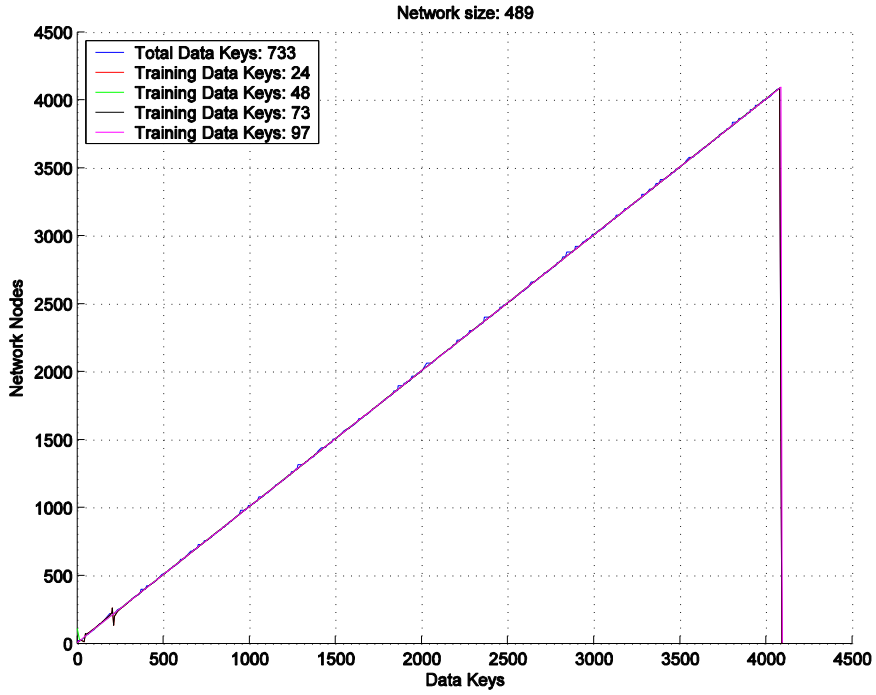
του δικτύου. Παρατηρούμε πως στη συγκεκριμένη προσομοίωση, ο υπολογισμός της καταλληλότητας με βάση τα κλειδιά του δείγματος εκπαίδευσης εμφανίζει καλύτερα αποτελέσματα με μέγεθος δείγματος 48 κλειδιά, ενώ ο υπολογισμός σε ολόκληρο το σύνολο κλειδιών παρουσιάζει καλύτερα αποτελέσματα για το δείγμα εκπαίδευσης 73 κλειδιών.

Πίνακας 9.4: Δείγμα Εκπαίδευσης

Μέγεθος Δείγματος Εκπαίδευσης	Πρωτάθλημα				
	Ελιτιστική	Fitness-Based	Απόλυτη	Ομοιόμορφη	Μέσος Όρος
24	3,96	4,13	3,83	3,83	3,94
48	3,98	3,79	4,00	3,79	3,89
73	4,90	4,77	4,88	4,84	4,85
97	5,01	4,32	4,98	5,03	4,84

Πίνακας 9.5: Σύνολο Κλειδιών

Μέγεθος Δείγματος Εκπαίδευσης	Πρωτάθλημα				
	Ελιτιστική	Fitness-Based	Απόλυτη	Ομοιόμορφη	Μέσος Όρος
24	22,05	21,63	22,30	21,65	21,91
48	13,54	14,09	13,56	13,57	13,69
73	5,32	5,01	5,03	5,04	5,10
97	9,19	11,91	11,42	9,23	10,44



Εικόνα 9.12: Κατανομή Κλειδιών για Διαφορετικού Μεγέθους Δείγματα Εκπαίδευσης

9.2 Κόστος Αναζήτησης

Στην παράγραφο αυτή θα εξεταστεί πως μεταβάλλεται το κόστος αναζήτησης με την προσθήκη του γενετικού μηχανισμού στους αλγορίθμους του Chord και S-Chord, που παρουσιάστηκαν σε προηγούμενο κεφάλαιο.

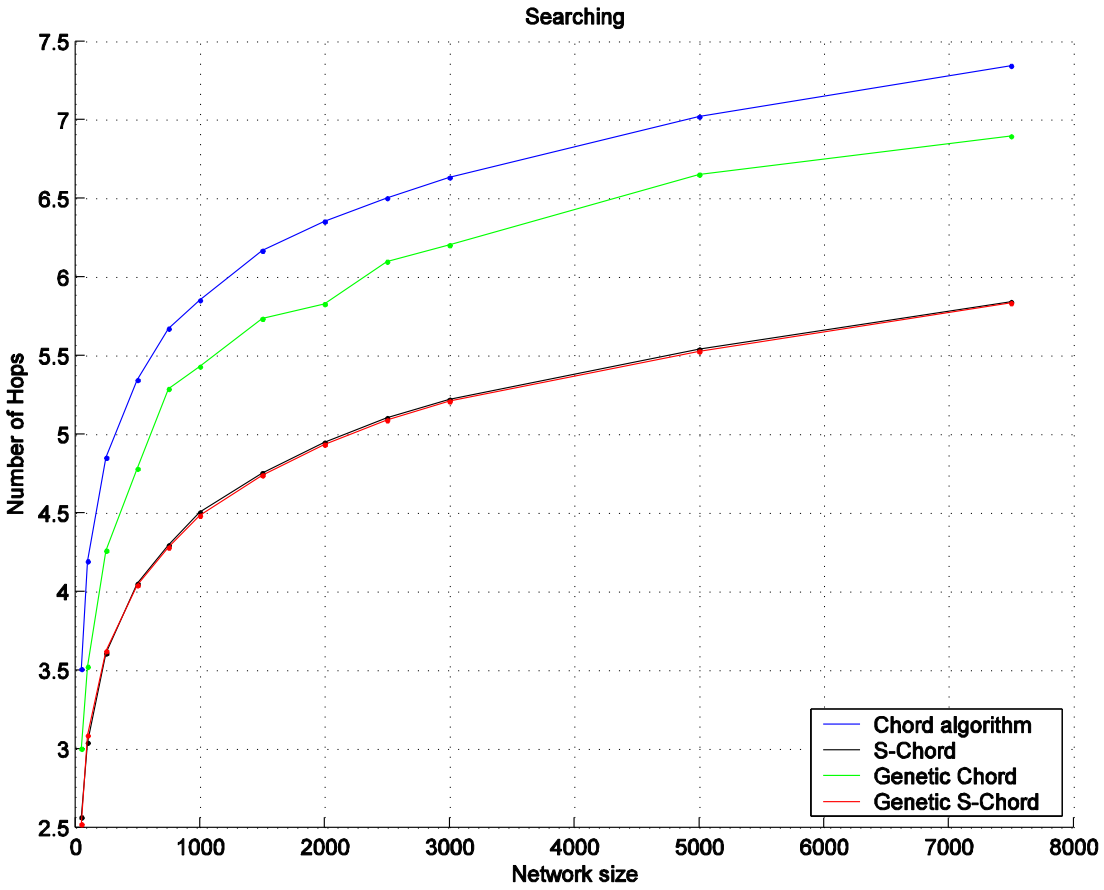
Για τις ανάγκες της διπλωματικής υλοποιήθηκαν οι αλγόριθμοι αναζήτησης τόσο στο Chord όσο και στο S-Chord. Οι αλγόριθμοι υλοποιήθηκαν με αναδρομικό τρόπο, δηλαδή κάθε ενδιάμεσος κόμβος στην αναζήτηση προωθεί την επερώτηση σε επόμενο κόμβο μέχρι να εντοπιστεί ο κόμβος που έχει το κλειδί. Μια αναζήτηση θεωρείται επιτυχημένη, εφόσον εντοπισθεί ο επόμενος κόμβος του κλειδιού που αναζητούμε. Οι μετρήσεις που πραγματοποιούμε αφορούν το μήκος των μονοπατιών αναζήτησης, αφού η απόδοση των πρωτοκόλλων εξαρτάται από τους κόμβους που πρέπει να επισκεφτούμε ώστε να απαντηθεί μια επερώτηση. Το μήκος των μονοπατιών αναζήτησης εξετάζεται σε ένα στιγμιότυπο του δικτύου. Στα παρόν αποτελέσματα δεν αναφερόμαστε πως επηρεάζεται η αναζήτηση με τη μεταβολή του δικτύου. Το ίδιο στιγμιότυπο του δικτύου χρησιμοποιείται για όλους τους αλγορίθμους αναζήτησης που συγκρίνονται.

Για την πραγματοποίηση των δοκιμών προσομοιώνονται δίκτυα με πλήθος κόμβων από 250 έως 7500 κόμβους, ενώ τα κλειδιά στο δίκτυο ποικίλουν από το μισό πλήθος των κόμβων του δικτύου μέχρι διπλάσιο. Οι κόμβοι κατανέμονται ομοιόμορφα στο χώρο αναζήτησης, ενώ τα κλειδιά ανατίθενται στους κόμβους με συνεπή κατακερματισμό. Το πλήθος των κλειδιών που κατανέμεται σε κάθε κόμβο ποικίλει και οι διακυμάνσεις αυτές μεταβάλλονται όσο αυξάνεται το πλήθος των κλειδιών στο δίκτυο. Κάθε κόμβος είναι υπεύθυνος για ένα υποσύνολο του δακτυλίου (και για τα κλειδιά στο υποσύνολο αυτό), το οποίο καθορίζεται από την απόσταση του κόμβου με τον προηγούμενο του. Κάθε κόμβος πραγματοποιεί αναζητήσεις προς όλα τα κλειδιά που έχουν ανατεθεί στο δίκτυο και σε κάθε βήμα υπολογίζεται το μήκος του μονοπατιού. Στις μετρήσεις αυτές δεν πραγματοποιούνται αναζητήσεις προς κλειδιά που δεν υπάρχουν στο δίκτυο, άρα όλες οι αναζητήσεις θα είναι επιτυχημένες. Οι αλγόριθμοι συγκρίνονται με βάση τον μέσο μήκος του μονοπατιού. Στον Πίνακας 9.6 συγκεντρώνονται οι μέσοι όροι των αποτελεσμάτων αναζήτησης για το Chord, το S-Chord και τις τροποποιήσεις τους με την προσθήκη του γενετικού μηχανισμού. Οι τροποποιήσεις που εξετάζουμε είναι: αλγόριθμος με οπισθοδρόμηση (rollback) χρησιμοποιώντας πίνακες δρομολόγησης όπως το Chord (Genetic Chord) και αλγόριθμος που χρησιμοποιεί πίνακες δρομολόγησης όπως το S-Chord (Genetic S-Chord). Το μέγεθος των πινάκων

δρομολόγησης σε όλες τις περιπτώσεις είναι ίδιο και ίσο με 14 (Οι προσδιοριστές παίρνουν τιμές στο $\left[0, 2^{14}-1\right]$).

Πίνακας 9.6: Μέσο Μήκος Μονοπατιού Αναζήτησης

Μέγεθος Δικτύου	Αλγόριθμοι Αναζήτησης			
	Chord	S-Chord	Genetic Chord	Genetic S-Chord
50	3,50640	2,56320	2,99960	2,51880
100	4,19170	3,03720	3,52110	3,08430
250	4,84976	3,60715	4,25971	3,61970
500	5,34404	4,04817	4,78054	4,03858
750	5,67215	4,29432	5,28916	4,27900
1000	5,85211	4,50269	5,42980	4,48092
1500	6,16595	4,75228	5,73298	4,73863
2000	6,35334	4,94636	5,82745	4,93363
2500	6,50116	5,10311	6,09720	5,08757
3000	6,63242	5,21949	6,20125	5,20807
5000	7,01953	5,53892	6,65175	5,52641
7500	7,34324	5,84091	6,89496	5,83299



Εικόνα 9.13: Σύγκριση Μέσου Μονοπατιού Αναζήτησης

Στην Εικόνα 9.13**:** Σύγκριση Μέσου Μονοπατιού Αναζήτησης απεικονίζεται το μέσο μήκος για τα μονοπάτια αναζήτησης. Το μήκος των μονοπατιών αυξάνεται λογαριθμικά με το πλήθος των κόμβων του δικτύου. Παρατηρούμε, πως η τροποποίηση του Chord με την προσθήκη του γενετικού μηχανισμού (πράσινη γραμμή) είναι σταθερά καλύτερη από τον κλασικό αλγόριθμο του Chord (μπλε γραμμή), περίπου κατά μισό βήμα. Ακόμα, τόσο το S-Chord όσο και η γενετική του τροποποίηση (μαύρη και κόκκινη γραμμή αντίστοιχα) είναι εμφανώς καλύτερα από τους δύο προηγούμενους αλγορίθμους. Η προσθήκη του γενετικού μηχανισμού στον αλγόριθμο του S-Chord δε μεταβάλλει την απόδοση. Πιο συγκεκριμένα, η τροποποιημένη αναζήτηση είναι σταθερά καλύτερη, αλλά σε πολύ μικρό βαθμό.

Η διαφοροποίηση στο μέσο μήκος μονοπατιού αναζήτησης οφείλεται στους διαφορετικούς τύπους πινάκων δρομολόγησης και στη διαφοροποίηση που εισάγει η προσθήκη του γενετικού μηχανισμού. Στη συνέχεια, ακολουθεί ένα παράδειγμα αναζήτησης όπου παραθέτουμε τα μονοπάτια αναζήτησης για κάθε ένα από τους αλγορίθμους. Η αναζήτηση ξεκινά από τον κόμβο με προσδιοριστή N11 και η ερώτηση αφορά το κλειδί με προσδιοριστή K3424. Στον παρακάτω πίνακα συνοψίζονται τα μονοπάτια για την αναζήτηση με χρήση κάθε αλγορίθμου.

Πίνακας 9.7: Μονοπάτια Αναζήτησης

Αλγόριθμος	**Μονοπάτι**
Chord	$N11, N2069, N3102, N3360, N3401, N3420, N3430$
Genetic Chord	$N11, N2069, N3102, N3360, N3430$
S-Chord	$N11, N3075, N3340, N3410, N3420, N3430$
Genetic S-Chord	$N11, N3075, N3340, N3410, N3430$

Παρατηρούμε, πως οι αλγόριθμοι που χρησιμοποιούν πίνακα δρομολόγησης με δείκτες και προς τις δύο κατευθύνσεις, κινούνται στην αναζήτηση με φορά αντίθετη από αυτή των δεικτών του ρολογιού, επιταχύνοντας έτσι την αναζήτηση. Επιπλέον, οι αλγόριθμοι όπου έχει προστεθεί ο γενετικός μηχανισμός καταφέρνουν να εντοπίσουν το κλειδί, αποφεύγοντας να επισκεφτούν προηγούμενους κόμβους από αυτόν που έχει το κλειδί. Αυτό συμβαίνει γιατί ο γενετικός μηχανισμός μας κατευθύνει προς τον κόμβο που είναι υπεύθυνος για το υποσύνολο του δακτυλίου όπου ανήκει το κλειδί που ψάχνουμε.

Στις παραπάνω προσομοιώσεις το πλήθος κλειδιών και κόμβων στο δίκτυο ήταν περίπου το ίδιο. Ακόμα, οι προσδιοριστές παίρνουν τιμές στο $0,16383$ (μήκος 14 bits). Κάτι τέτοιο δεν είναι η γενική περίπτωση. Στα δίκτυα ομότιμων κόμβων, συνήθως, τα κλειδιά είναι περισσότερα από τους κόμβους του δικτύου, ενώ οι προσδιοριστές έχουν μεγαλύτερο μήκος (π.χ. 128, 160 bits - ακολουθίες ψηφίων στο δεκαεξαδικό σύστημα) όπως και οι πίνακες δρομολόγησης. Στη συνέχεια, θα ακολουθήσουν παραδείγματα όπου υπολογίζεται το κόστος αναζήτησης σε δίκτυα όπου το πλήθος κλειδιών ξεπερνά κατά πολύ το πλήθος κόμβων του δικτύου, ενώ και το εύρος δυνατών τιμών για τους προσδιοριστές/κλειδιά (χώρος κατακερματισμού) είναι μεγαλύτερος.

Στον Πίνακας 9.8 συνοψίζονται τα αποτελέσματα για τους αλγορίθμους αναζήτησης όπου το πλήθος κόμβων του δικτύου είναι 300 και 500, ενώ τα κλειδιά είναι 6000 και 5000 αντίστοιχα. Παρατηρούμε, πως η απόδοση των αλγορίθμων δε μεταβάλλεται.

Πίνακας 9.8: Αποτελέσματα Αναζήτησης

Αλγόριθμος	Μέσο μονοπάτι αναζήτησης	
	Κόμβοι/ Κλειδιά: 300/6000	Κόμβοι/ Κλειδιά: 500/5000
Chord	4.973886	5.3197008
S-Chord	3.725662	4.0443608
Genetic Chord	4.634048	4.9218992
Genetic S-Chord	3.702506	4.0210152

Στην περίπτωση, όπου αυξάνεται το εύρος δυνατών τιμών για τους προσδιοριστές/κλειδιά η απόδοση των αλγορίθμων δε μεταβάλλεται. Αυτό, που αλλάζει σε σχέση με τις προηγούμενες προσομοιώσεις είναι η αύξηση στην τιμή της συνάρτησης καταλληλότητας. Η συνάρτηση καταλληλότητας υπολογίζεται με βάση το σφάλμα στο χώρο τιμών των προσδιοριστών. Όσο μεγαλώνει το εύρος δυνατών τιμών για τους προσδιοριστές, αυξάνει και το σφάλμα κατά τον υπολογισμό της γενετικής συνάρτησης. Όπως παρατηρούμε από τα αποτελέσματα, μια τέτοια αύξησης δεν αλλοιώνει την ικανότητα αναζήτησης με χρήση του γενετικού μηχανισμού, αφού όσο μεγαλώνει το εύρος τιμών των προσδιοριστών τόσο μια μεγαλύτερη απόκλιση στην τιμή προσδιοριστή που επιστρέφει ο γενετικός μηχανισμός και στον προσδιοριστή του κόμβου που έχει το κλειδί, είναι αποδεκτή. Για παράδειγμα, αν το εύρος τιμών των προσδιοριστών είναι από 0 έως 100000, τότε μια απόκλιση μεγέθους 100 στον προσδιοριστή του κόμβου που δίνει η γενετική συνάρτηση και στον κόμβο που έχει το κλειδί δεν αποτελεί σημαντικό σφάλμα.

Στη συνέχεια, ακολουθεί ένα ακόμα παράδειγμα προσομοίωσης δικτύου. Το δίκτυο αποτελείται από 2000 κόμβους, ενώ τα κλειδιά στο δίκτυο είναι 80000. Από αυτά τα 4000 χρησιμοποιούνται στο δείγμα εκπαίδευσης για τον υπολογισμό της συνάρτησης καταλληλότητας. Οι προσδιοριστές των κόμβων και των κλειδιών έχουν μέγεθος 24 bits και λαμβάνουν τιμές από 0 έως 167772165. Το πλήθος κλειδιών αλλά και οι τιμές που μπορούν να πάρουν οι προσδιοριστές προσεγγίζει πολύ καλά ένα πραγματικό δίκτυο. Η συνάρτηση καταλληλότητας στην προσομοίωση αυτή υπολογίζεται μετά από 75 γενιές και έχει κανονικοποιημένη τιμή 5654,5. Όπως αναφέρθηκε και παραπάνω, μια τέτοια απόκλιση στον υπολογισμό είναι αποδεκτή δεδομένου του εύρους τιμών των προσδιοριστών. Στον Πίνακας 9.9 παρατίθενται τα αποτελέσματα από 1250000 αναζήτησης.

Πίνακας 9.9: Αποτελέσματα Αναζήτησης

Αλγόριθμος	Μέσο μονοπάτι αναζήτησης
	Κόμβοι/Κλειδιά: 2000/80000
Chord	6.347926
S-Chord	4.945007
Genetic Chord	5.800824
Genetic S-Chord	4.934356

Παρατηρούμε πως η γενετική τροποποίηση του Chord είναι σταθερά καλύτερη από τον κλασικό αλγόριθμο του Chord. Ενώ οι δύο περιπτώσεις όπου χρησιμοποιούνται πίνακες δρομολόγησης όπως του S-Chord έχουν αντίστοιχη συμπεριφορά. Όπως βλέπουμε και από τα αποτελέσματα, η συμπεριφορά των αλγορίθμων παραμένει ίδια με τις προσομοιώσεις που εξετάσαμε παραπάνω και δεν επηρεάστηκε από το μεγάλο πλήθος κλειδιών και το εύρος τιμών των προσδιοριστών.

9.3 Υπολογισμός Κόστους Αναζήτησης με μεταβολή του δικτύου

Στις προηγούμενες παραγράφους, υπολογίστηκε το μήκος του μονοπατιού αναζήτησης για τις διαφορετικές παραλλαγές των αλγορίθμων αναζήτησης. Οι υπολογισμοί αυτοί πραγματοποιήθηκαν σε ένα στιγμιότυπο του δικτύου, όπου η δομή του δικτύου δεν παρουσίαζε μεταβολές. Κάτι τέτοιο δεν είναι η συνηθισμένη περίπτωση στα δίκτυα ομότιμων κόμβων, όπου η δομή του δικτύου μεταβάλλεται συνέχεια, με αφίξεις και αναχωρήσεις κόμβων. Στην περίπτωση μας δεν αλλάξει ο τρόπος λειτουργίας των πρωτοκόλλων, αφού ο γενετικός μηχανισμός μεταβάλλει μόνο τους αλγορίθμους αναζήτησης. Οπότε δε χρειάζεται να εξετάσουμε την ικανότητα τον πρωτοκόλλων να παραμένουν συνεπή (consistent). Αυτό που πρέπει να εξεταστεί, είναι πόσο καλά μπορεί να λειτουργεί η γενετική συνάρτηση εφόσον το δίκτυο για το οποίο

υπολογίστηκε, μεταβάλλεται. Ένας τέτοιος υπολογισμός θα μας δώσει και μια ένδειξη για το πόσο συχνά πρέπει να υπολογίζεται ο γενετικός μηχανισμός.

Για τις ανάγκες της προσομοίωσης αυτής χρησιμοποιείται ένα δίκτυο αποτελούμενο αρχικά από 400 κόμβους, ενώ και τα κλειδιά στο δίκτυο είναι και αυτά 400. Το μέγεθος του δείγματος εκπαίδευσης είναι το 20% του συνόλου των κλειδιών (εδώ 80). Η συνάρτηση καταλληλότητας υπολογίστηκε σε 75 γενιές και είχε κανονικοποιημένη τιμή 30,98. Τη συνάρτηση αυτή θα τη χρησιμοποιήσουμε σε όλη τη διάρκεια της προσομοίωσης για εξετάσουμε πως συμπεριφέρεται ο γενετικός μηχανισμός με τη μεταβολή στη μορφή του δικτύου. Στον Πίνακας 9.10 αποτυπώνεται το μέσο μήκος μονοπατιού αναζήτησης για το δίκτυο, αμέσως μετά τον υπολογισμό της συνάρτησης καταλληλότητας.

Πίνακας 9.10: Αρχικά Αποτελέσματα Αναζήτησης

Αλγόριθμος	Μέσο μονοπάτι αναζήτησης
	Κόμβοι/Κλειδιά: 400/400
Chord	5.1977125
S-Chord	3.9147937
Genetic Chord	4.7965312
Genetic S-Chord	3.8937625

Στη συνέχεια, μεταβάλλουμε τη δομή του δικτύου με την εισαγωγή και απομάκρυνση τόσο κόμβων όσο και κλειδιών. Συνολικά, πραγματοποιούνται 194 αφίξεις κόμβων και 139 αποχωρήσεις με αποτέλεσμα το δίκτυο στο τέλος της προσομοίωσης να αποτελείται από 455 κόμβους, ενώ τα κλειδιά είναι 900. Μετά την ολοκλήρωση της προσομοίωσης 37 κλειδιά από το αρχικό δείγμα εκπαίδευσης και 187 από το αρχικό σύνολο κλειδιών παραμένουν στο δίκτυο. Παρόλο που το δίκτυο έχει αλλάξει σημαντικά ο γενετικός μηχανισμός με την αρχική συνάρτηση καταλληλότητας εξακολουθεί να λειτουργεί ικανοποιητικά όπως παρατηρούμε από τα αποτελέσματα αναζητήσεων στον παρακάτω πίνακα.

Πίνακας 9.11: Τελικά Αποτελέσματα Αναζήτησης

Αλγόριθμος	Μέσο μονοπάτι αναζήτησης
	Κόμβοι/Κλειδιά: 455/900
Chord	5.2397509
S-Chord	3.9799169
Genetic Chord	4.7425445
Genetic S-Chord	3.9754798

Όπως αναφέρθηκε και παραπάνω, ο υπολογισμός της συνάρτησης καταλληλότητας δεν πρέπει να πραγματοποιείται πολύ συχνά, αφού κάτι τέτοιο απαιτεί τόσο επικοινωνία μεταξύ των κόμβων όσο και υπολογιστικό κόστος. Πριν αποφασίσουμε πόσο συχνά πρέπει να υπολογίζεται ξανά η γενετική συνάρτηση πρέπει να λάβουμε υπόψη πως τα κλειδιά κατανέμονται στο δακτύλιο με συνεπή κατακερματισμό. Έτσι, για ένα δίκτυο με Ν κόμβους και Κ κλειδιά (με μεγάλη πιθανότητα) όταν ένας κόμβος συνδέεται ή αποχωρεί από το δίκτυο, $O(K/N)$ κλειδιά μετατοπίζονται προς ή από τον κόμβο που συνδέεται ή αποχωρεί, αντίστοιχα. Επιπλέον, κατά τις αναχωρήσεις και αφίξεις κόμβων τα κλειδιά μετατοπίζονται προς γειτονικούς κόμβους στο δακτύλιο. Λαμβάνοντας υπόψη τις δύο παραπάνω παρατηρήσεις και δεδομένου πως η γενετική συνάρτηση μας κατευθύνει προς μια περιοχή αναζήτησης γίνεται κατανοητό πως εφόσον τα κλειδιά παραμένουν στην ίδια περιοχή του δακτυλίου η γενετική συνάρτηση μπορεί να συνεχίσει να υποστηρίζει τις αναζητήσεις. Κάτι τέτοιο παρατηρήθηκε και από την προσομοίωση που παρουσιάστηκε παραπάνω.

9.4 Επιλογή Μεγέθους Δείγματος Εκπαίδευσης

Στην παράγραφο αυτή θα παρουσιαστεί ένα παράδειγμα υπολογισμού της συνάρτησης καταλληλότητας χρησιμοποιώντας διαφορετικά μεγέθη στο δείγμα εκπαίδευσης.

Για τις ανάγκες του παραδείγματος, χρησιμοποιήθηκαν 400 γενετικά προγράμματα, ενώ το σύνολο των κλειδιών ήταν 400. Ο αλγόριθμος επιλογής είναι το πρωτάθλημα, ενώ ο αλγόριθμος επαναεισαγωγής που χρησιμοποιείται είναι ο ελιτιστικός. Στον παρακάτω πίνακα συγκεντρώνονται οι τιμές της συνάρτησης καταλληλότητας που υπολογίζονται με τη βοήθεια των 400 γενετικών προγραμμάτων, χρησιμοποιώντας διαφορετικό κάθε φορά μέγεθος δείγματος εκπαίδευσης. Τα κλειδιά που αποτελούν το δείγμα εκπαίδευσης είναι ομοιόμορφα κατανεμημένα στο σύνολο των κλειδιών που υπάρχουν στο δίκτυο. Τα παρακάτω δεδομένα αποτελούν τους μέσους όρους από ένα σύνολο 10 προσομοιώσεων. Η καταλληλότητα κανονικοποιείται με βάση το μέγεθος του δείγματος εκπαίδευσης.

Πίνακας 9.12: Καταλληλότητα για Διαφορετικού Μεγέθους Δείγματα Εκπαίδευσης

Δείγμα Εκπαίδευσης	Καταλληλότητα	Κανονικοποιημένη Καταλληλότητα
40	309,1	7,7275
80	673	8,4125
120	1091,1	9,0925
160	1376,6	8,6038
200	1946,4	9,7320

240	2575,3	10,7304
280	2731,9	9,7568
320	2562,7	8,0084
360	3887,1	10,7975
400	4260,3	10,6507

Παρατηρούμε πως δεν παρουσιάζονται σημαντικές διαφορές στην κανονικοποιημένη τιμή καταλληλότητας που υπολογίζεται όσο αυξάνεται το μέγεθος του δείγματος εκπαίδευσης. Τα αποτελέσματα αυτά δεν μας επιτρέπουν να εξάγουμε κάποια συμπεράσματα για πιο ποσοστό του συνόλου των κλειδιών του δικτύου πρέπει να χρησιμοποιηθεί ως δείγμα εκπαίδευσης. Όμως, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο σκοπός αυτής της εργασίας δεν είναι να υπολογιστεί ποιο είναι το κατάλληλο μέγεθος του δείγματος. Οι μετρήσεις αυτές περιλαμβάνονται ως έναυσμα για μελλοντική μελέτη.

Διαισθητικά μπορούμε να αναφέρουμε πως σε ένα δίκτυο που υιοθετείται ο γενετικός μηχανισμός, το δείγμα εκπαίδευσης πρέπει να ανανεώνεται όσο μεταβάλλεται η τοπολογία του δικτύου, αλλά και η κατανομή των κλειδιών. Ενώ, είναι επιθυμητό στον υπολογισμό να χρησιμοποιούνται κλειδιά που αντιπροσωπεύουν όσο το δυνατών περισσότερα από τα διαστήματα ευθύνης που υπάρχουν στο δίκτυο.

ΚΕΦΑΛΑΙΟ 10

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ

10.1 Συμπεράσματα

Στο πλαίσιο αυτής της εργασίας ενσωματώσαμε ένα γενετικά παραγόμενο μηχανισμό πρόβλεψης για την εύρεση του κόμβου που είναι υπεύθυνος για ένα κλειδί, στους αλγόριθμους αναζήτησης των πρωτοκόλλων Chord και S-Chord. Στο προηγούμενο κεφάλαιο παρουσιάστηκαν τα αποτελέσματα από τις προσομοιώσεις των αλγορίθμων αναζήτησης με την προσθήκη του γενετικού μηχανισμού, ενώ στο κεφάλαιο 8 παρουσιάστηκαν οι επιλογές για τον υπολογισμό του γενετικού μηχανισμού, πως ο μηχανισμός αυτός ενσωματώνεται στους αλγορίθμους αναζήτησης καθώς και πως τροποποιούνται οι αλγόριθμοι αυτοί. Στη συνέχεια θα αναφερθούμε στα κυριότερα συμπεράσματα που προέκυψαν.

Η προσθήκη του προτεινόμενου γενετικού μηχανισμού στο πρωτόκολλο Chord βελτιώνει την απόδοση του πρωτοκόλλου μειώνοντας το πλήθος βημάτων που απαιτούνται για την αναζήτηση (σταθερή μείωση μισό βήμα). Η προσθήκη του ίδιου μηχανισμού στο πρωτόκολλο S-Chord οδηγεί σε σταθερή βελτίωση του πρωτοκόλλου αλλά σε πολύ μικρό ποσοστό. Επιπλέον, εμμέσως δίνεται η δυνατότητα να συγκρίνουμε την απόδοση των πρωτοκόλλων Chord και S-Chord, επιβεβαιώνοντας τη βελτίωση σε ποσοστό 25% του S-Chord σε σχέση με το Chord χρησιμοποιώντας πίνακες δρομολόγησης ίδιου μεγέθους.

Όσον αφορά το κομμάτι του γενετικού προγραμματισμού αξίζει να αναφέρουμε πως η εφαρμογή του γενετικού μηχανισμού μπορεί να συνεργαστεί εύκολα με τα p2p πρωτόκολλα αφού στη διαδικασία βελτιστοποίησης χρησιμοποιεί μόνο πληροφορίες του συστήματος (δείγμα εκπαίδευσης) και δεν επεμβαίνει στη δομή των πρωτοκόλλων (π.χ. πίνακες δρομολόγησης). Ακόμα, οι διαφορετικές εκτελέσεις του αλγορίθμου τόσο σε ίδια δίκτυα όσο και σε διαφορετικά, παράγει αποτελέσματα χωρίς μεγάλη διασπορά. Στις προσομοιώσεις για τον υπολογισμό της συνάρτησης καταλληλότητας χρησιμοποιήθηκαν διάφοροι συνδυασμοί αλγορίθμων επιλογής και επαναεισαγωγής στον πληθυσμό. Για το συγκεκριμένο πρόβλημα, δεν παρατηρήθηκαν μεγάλες διαφορές στο αποτέλεσμα της συνάρτησης καταλληλότητας που υπολογίζεται. Η μόνη διαφοροποίηση αφορούσε τη γενιά σύγκλισης όπου ο αλγόριθμος επιλογής με πρωτάθλημα (tournament) είχε την τάση να οδηγεί σε γρηγορότερη σύγκλιση.

Επιπλέον, δείξαμε πως ο υπολογισμός της συνάρτησης καταλληλότητας δε χρειάζεται να πραγματοποιείται πολύ συχνά αφού οι αλλαγές στη δομή του δικτύου δεν οδηγούν σε μεγάλες μετατοπίσεις των κλειδιών από την περιοχή ευθύνης όπου μας κατευθύνει ο γενετικός μηχανισμός.

Στις προσομοιώσεις ο εξελικτικός αλγόριθμος σε καμία περίπτωση δεν μπόρεσε να εντοπίσει την καθολικά βέλτιστη λύση. Οι θεωρητικές αποδείξεις για καθολική σύγκλιση προϋποθέτουν άπειρο υπολογιστικό χρόνο, ενώ η προτεινόμενη λύση προέκυψε σε περιορισμένο υπολογιστικά χρόνο (πλήθος γενεών). Παρόλα αυτά, το σφάλμα που προέκυψε ήταν αποδεκτό για τις ανάγκες του προβλήματος που εξετάστηκε. Γενικά, η απαίτηση του προβλήματος σε υπολογιστική ισχύ δεν είναι πολύ μεγάλη και οφείλεται κυρίως στο μέγεθος του πληθυσμού. Η έννοια του πληθυσμού στο γενετικό προγραμματισμό καθιστά την παραλληλοποίηση του αλγορίθμου υπολογισμού της συνάρτησης καταλληλότητας σχετικά εύκολη, μειώνοντας έτσι το χρόνο εκτέλεσης του αλγορίθμου. Η μεγαλύτερη δυσκολία κατά τον υπολογισμό της συνάρτησης καταλληλότητας (αλλά και γενικότερα στους εξελικτικούς αλγορίθμους) αποτέλεσε η διευθέτηση των παραμέτρων προσομοίωσης (π.χ. το είδος της επιλογής, το είδος του τελεστή διασταύρωσης, το μέγεθος του πληθυσμού, την πιθανότητα εφαρμογής του κάθε τελεστή, η μορφή της συνάρτησης καταλληλότητας).

10.2 Προτάσεις

Τα αποτελέσματα που προέκυψαν από τις προσομοιώσεις δίνουν υποσχέσεις για εφαρμογή ενός τέτοιου μηχανισμού κυρίως στο πρωτόκολλο του Chord. Υπάρχουν αρκετά σημεία που αξίζουν ακόμα διερεύνησης. Το πιο σημαντικό από αυτά αφορά το δείγμα εκπαίδευσης. Όπως αναφέρθηκε και σε μια από τις προσομοιώσεις στο προηγούμενο κεφάλαιο, δεν είναι σαφές πιο πρέπει να είναι το μέγεθος του δείγματος εκπαίδευσης, αλλά και ποια κλειδιά θα παρέχουν μια αρκετά καλή εικόνα για τη μορφή κατανομής κλειδιών στον εικονικό δακτύλιο. Στις προσομοιώσεις που παρουσιάστηκαν θεωρήθηκε πως το δείγμα εκπαίδευσης είναι ένα ομοιόμορφα κατανεμημένο υποσύνολο του συνόλου των κλειδιών (π.χ. 5% - 20%). Επιπλέον, αξίζει να υπολογιστεί πιο είναι το κόστος για τη συλλογή του δείγματος εκπαίδευσης (π.χ. ανταλλαγή μηνυμάτων για τη συλλογή δεδομένων, ανταλλαγή μηνυμάτων για την εκλογή του κόμβου που θα πραγματοποιήσει τον υπολογισμό).

Ακόμα, δεδομένου του κόστους σε υπολογιστική ισχύ ενδιαφέρον παρουσιάζει και η σχεδίαση ενός παράλληλου αλγορίθμου για τον υπολογισμό της συνάρτησης καταλληλότητας. Ένας τέτοιος αλγόριθμος θα εκμεταλλεύεται την έννοια του πληθυσμού

και δεδομένου πως δε χρειάζεται σημαντική επικοινωνία ανάμεσα στα άτομα που συμμετέχουν στους υπολογισμούς, θα είναι σε θέση να μειώσει το χρόνο εκτέλεσης του υπολογισμού και να κατανείμει τις ανάγκες για υπολογιστική ισχύ, μεταξύ περισσότερων κόμβων.

Τέλος, ενδιαφέρον θα είχε μια προσπάθεια για την εφαρμογή του μηχανισμού στην πράξη σε κάποια υλοποίηση των πρωτοκόλλων Chord[8] ή S-Chord, ώστε να μελετηθεί στην πράξη πόσο καλά μπορεί να δουλέψει ένας τέτοιος αλγόριθμος. Ακόμα, ενδιαφέρον θα παρουσίαζε μια μελέτη κατά πόσο είναι εφικτό κάποιος αντίστοιχος μηχανισμός που βασίζεται σε γενετικό προγραμματισμό να εφαρμοστεί σε άλλα πρωτόκολλα δικτύων ομότιμων κόμβων.

[8] π.χ. Open Chord

ΟΡΟΛΟΓΙΑ

Terminal set	Τερματικά σύμβολα
Functional set	Συναρτησιακά σύμβολα
Fitness function	Συνάρτηση καταλληλότητας/αξιολόγησης
Sufficiency	Επάρκεια
Symbolic Regression	Συμβολική Παλινδρόμηση
Crossover	Διασταύρωση
Mutation	Μετάλλαξη
Reproduction	Αναπαραγωγή
Permutation	Μετάθεση
Editing	Επεξεργασία
Encapsulation	Ενθυλάκωση
Raw fitness	Ακατέργαστη καταλληλότητα
Standardized fitness	Τυποποιημένη καταλληλότητα
Adjusted fitness	Προσαρμοσμένη καταλληλότητα
Normalized fitness	Κανονικοποιημένη καταλληλότητα
Fitness case	Δείγμα εκπαίδευσης
Grow method	Μέθοδος εξέλιξης
Full method	Πλήρης μέθοδος
Ramped half and half method	Μισή-Μισή μέθοδος
Fitness based selection	Επιλογή ανάλογα την καταλληλότητα
Tournament selection	Επιλογή με πρωτάθλημα
Truncation selection	Επιλογή με αποκοπή
Ranking selection	Επιλογή με διαβάθμιση
Pure Reinsertion	Καθαρή επαναεισαγωγή
Uniform Reinsertion	Ομοιόμορφη επαναεισαγωγή
Elitist Reinsertion	Ελιτιστική επαναεισαγωγή
Fitness-based Reinsertion	Με βάση την καταλληλότητα επαναεισαγωγή
Purely decentralized	Πλήρως αποκεντρωμένα
Partially centralized	Με μερικό κεντρικό έλεγχο
Hybrid decentralized	Υβριδικά αποκεντρωμένα
Peer-to-peer	Ομότιμος κόμβος
Supernode	Υπερκόμβος
Distributed Hash Tables	Κατανεμημένοι πίνακες κατακερματισμού
Consistent hashing	Συνεπής κατακερματισμός
Finger table	Πίνακας δρομολόγησης

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ

DHT	Distributed Hash Tables
P2P	Peer-to-Peer
GP	Genetic Programming

ΑΝΑΦΟΡΕΣ

1. D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin. “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web”, In Symp. on Theory of Computing (STOC), May 1997, p.p. 654–663.
2. J. R., Koza, “Genetic Programming: On the Programming of Computers by Means of Natural Selection”, The MIT Press, 1992.
3. M., Mitchell, “An introduction to genetic algorithms”, MIT Press, MA, 1999.
4. Timothy, Lai, “Discovery of Understandable Math Formulas Using Genetic Programming”, Computer Science Program, Stanford University.
5. J. R., Koza, “Introduction to Genetic Programming”, Tutorial, Gecco 2004-Seattle.
6. Jaymin, B., Kessler, “Using Genetic Algorithms to Reorganize Superpeer Structure in Peer to Peer Networks”, Thesis (Under the direction of Khaled Rasheed and Budak Arpinar).
7. W. Banzhaf, P. Nordin, R. Keller and Frank D. Francone, “Genetic Programming - An Introduction: On the Automatic Evolution of Computer Programs and its Applications”, Morgan Kaufmann, 1998.
8. Ion, Stoica, et al, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications", SIGCOMM 2001, pages 149–160.
9. S. Androutsellis-Theotokis and D. Spinellis. “A Survey of Peer-to-Peer File Sharing Technologies”, White paper, Electronic Trading Research Unit (ELTRUN), Athens University of Economics and Business, 2002.
10. T., Blickle, and Thiele, L.. “A Comparison of Selection Schemes used in Genetic Algorithms.”, Technical Report No. 11. Gloriastrasse 35, CH-8092 Zurich: Swiss Federal Institute of Technology (ETH) Zurich, Computer Engineering and Communications Networks Lab (TIK), 1995.
11. Tobias, Blickle, “Theory of Evolutionary Algorithms and Application to System Synthesis”, TIK-Schriftenreihe Nr. 17. Zurich, Switzerland: vdf Hochschul Verlag AG and der ETH Zurich. ISBN 3-7281-2433-8, 1997.
12. John, R., Koza. “Genetic programming as a means for programming computers by natural selection”, Statistics and Computing, 4(2):87--112, June 1994.
13. S., Luke, “Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat”, PhD Dissertation. U. Maryland, 2000.
14. P., Felber, “Peer-to-Peer Networks Unstructured vs Structured P2P Systems”, Presentation.
15. V.Mesaros, B.Carton, and P.Van Roy – “S-Chord: Using Symmetry to Improve Lookup Efficiency in Chord”, In Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA’03, June 23-26, 2003, Las Vegas, Nevada, USA.
16. David, Liben-Nowell, Hari Balakrishnan, and David Karger. “Analysis of the Evolution of Peer-to-Peer Systems”. In Proceedings PODC, pages 233–242, 2002.
17. http://www.genetic-programming.org, (A source of information about the field of genetic programming and the field of genetic and evolutionary computation).
18. http://www.genetic-programming.com (Genetic Programming Inc., a privately funded research group that does research in applying genetic programming).

19. John, R., Koza, 1990. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Stanford University Computer Science Department Technical Report STAN-CS-90-1314. June, 1990.
20. http://en.wikipedia.org/wiki/Genetic_Programming (Wikipedia, the free encyclopedia)
21. http://en.wikipedia.org/wiki/Genetic_algorithm (Wikipedia, the free encyclopedia)
22. J. Eggermont and J. I. van Hemert, “Adaptive genetic programming applied to new and existing simple regression problems,” In Proc. 4th European Conf. Genetic Programming, vol. 2038, LNCS, J. F. Miller et al., Eds., Lake Como, Italy, Apr. 18–20, 2001, pp. 23–35.