Υπολογιστική Νοημοσύνη

Ιωάννης Γ. Τσούλος

Τμήμα Πληροφορικής και τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων

2025

Περίληψη

Παράλληλος υπολογισμός καταλληλότητας

1. Είναι η πιο απλή περίπτωση χρήσης πολλών επεξεργαστών σε γενετικούς αλγόριθμους.
2. Σε αυτήν την τεχνική ο υπολογισμός της καταλληλότητας κάθε χρωμοσώματος ανατίθεται σε διαφορετικό επεξεργαστή
3. Εν συνεχεία οι τιμές αυτές συγκεντρώνονται σε έναν κεντρικό διακομιστή.

Ενδεικτικός αλγόριθμος

1. **Αρχικοποίηση πληθυσμού**.
2. **Υπολογισμός καταλληλότητας**.
   1. Ο master node αποστέλλει σε κάθε slave node το χρωμόσωμα που του αντιστοιχεί
   2. Υπολογισμός της αντίστοιχης καταλληλότητας.
   3. Επιστροφή στον μάστερ της τιμής.
3. **Εκτέλεση γενετικών τελεστών**., στον κεντρικό κόμβο.
4. **Έλεγχος κριτηρίου τερματισμού**, στον κεντρικό κόμβο.
5. **Μετάβαση** στο βήμα 2.

Προβλήματα

1. Θα πρέπει κάθε κόμβος να ξέρει όλο το πρόβλημα για να υπολογίσει την καταλληλότητα.
2. Οι κόμβοι επεξεργασίας θα πρέπει να είναι υπολογιστικά όμοιοι.
3. Αν τα χρωμοσώματα είναι πάρα πολλά ή μεγάλα σε μέγεθος , τότε θα δαπανηθεί αρκετός χρόνος για την επικοινωνία μεταξύ των κόμβων.
4. Αν τα χρωμοσώματα είναι περισσότερα από τους κόμβους επεξεργασίας , τότε κάθε κόμβος επεξεργασίας θα αναλάβει περισσότερους από έναν υπολογισμούς.
5. Συγχρονισμός τυχαίων αριθμών.
6. Θα πρέπει ο υπολογισμός της καταλληλότητας να έχει νόημα και να είναι πολύ δαπανηρός χρονικά.

Παράλληλη επιλογή

1. Στην περίπτωση της επιλογής με ρουλέτα απαιτείται ταξινόμηση των καταλληλοτήτων και επιλογή χρωμοσωμάτων ή ακόμα και δημιουργία ενός ενδιάμεσου πίνακα.
2. Τέτοιες ενέργειες μπορούν εύκολα να υλοποιηθούν παράλληλα σύμφωνα με την σχετική βιβλιογραφία, όπου κανείς μπορεί να βρει παράλληλες τεχνικές ταξινόμησης.
3. Φυσικά μια τέτοια διαδικασία θα έχει νόημα αν το πλήθος των χρωμοσωμάτων είναι αρκετά μεγάλο.

Παράλληλη διασταύρωση

1. Η διασταύρωση συνήθως είναι μια διαδικασία στην οποία γίνεται ανταλλαγή γενετικού υλικού μεταξύ χρωμοσωμάτων.
2. Αυτό προγραμματιστικά σημαίνει πράξεις πινάκων.
3. Αυτές οι πράξεις (αν τα χρωμοσώματα είναι μεγάλα σε μέγεθος) μπορούν να εκτελεστούν παράλληλα σύμφωνα με την σχετική βιβλιογραφία.

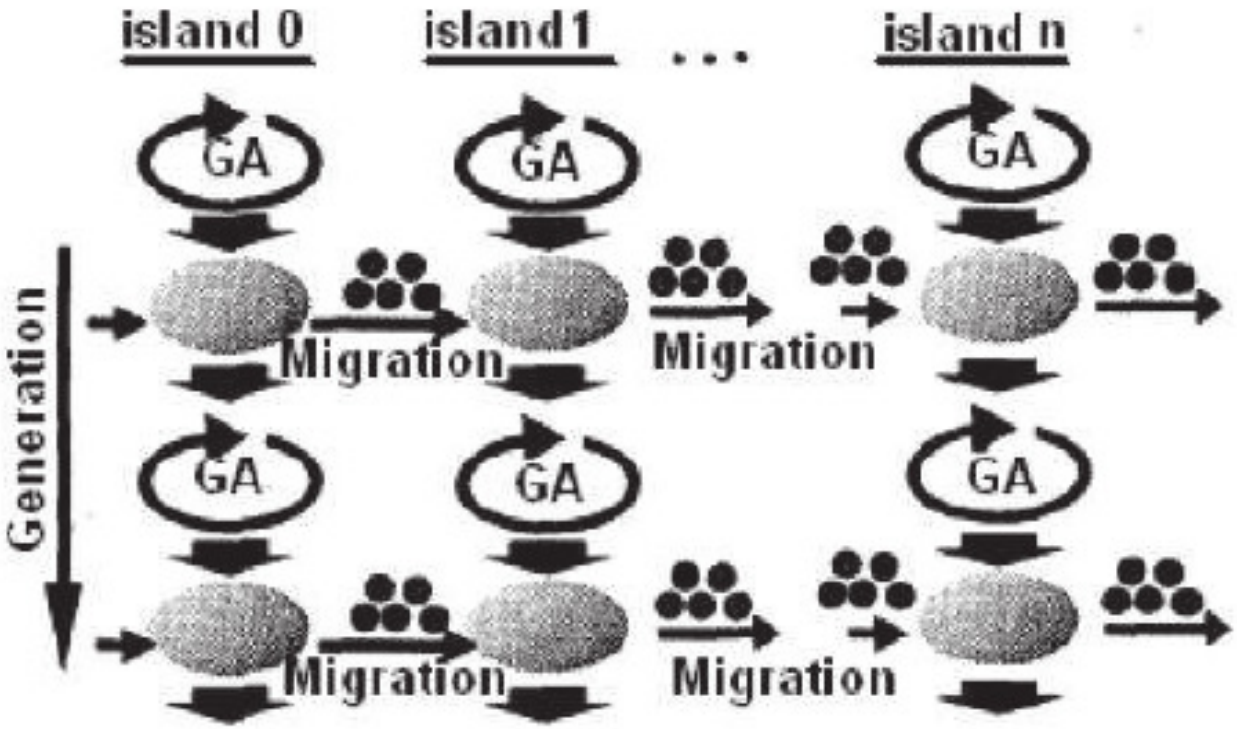
Βασικές αρχές

1. Στην βιβλιογραφία έχουν αναπτυχθεί διάφορες αρχιτεκτονικές για παράλληλους γενετικούς που εκμεταλλεύονται τις σύγχρονες παράλληλες υπολογιστικές δομές.
2. Οι κυριότερες από αυτές είναι οι αλγόριθμοι
    1. ISLAND
    2. CELLULAR.

Αλγόριθμοι ISLAND

1. Στα μοντέλα αυτά ο συνολικός πληθυσμός χωρίζεται σε ανεξάρτητα τμήματα που ονομάζονται νησιά.
2. Το κάθε τμήμα εξελίσσεται ανεξάρτητα από κάθε άλλο μέλος του πληθυσμού και η εκτέλεση γίνεται σε διαφορετικό επεξεργαστή.
3. Ανά τακτά χρονικά διαστήματα τα νησιά ανταλλάσσουν μηνύματα στα οποία συνήθως μπαίνει το καλύτερα ή τα καλύτερα χρωμοσώματα κάθε νησιού.
4. Αυτά τα μηνύματα είτε στέλνονται απευθείας από νησί σε νησί είτε στέλνονται πρώτα σε ένα κεντρικό κόμβο επεξεργασίας και στην συνέχεια από αυτό στέλνονται στα νησιά.

Αλγόριθμοι ISLAND(σχήμα)



Αλγόριθμοι ISLAND (Μέγεθος πληθυσμού)

1. Θα πρέπει να καθοριστεί το μέγεθος κάθε υποπληθυσμού.
2. Αν αυτό το μέγεθος είναι μικρό, τότε τα νησιά μπορεί να εγκλωβιστούν σε τοπικά ελάχιστα και να μην υπάρχει εξέλιξη.
3. Αν είναι πολύ μεγάλο κάθε νησί θα αργεί να προχωρήσει στην εξέλιξη του συνολικού αποτελέσματος.

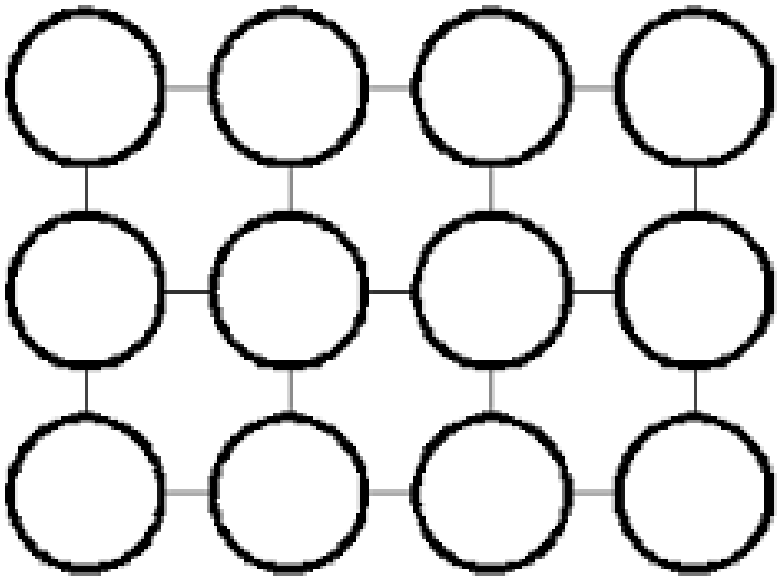
Αλγόριθμοι ISLAND (μηνύματα)

1. Θα πρέπει να καθοριστεί κάθε πότε θα γίνεται ανταλλαγή μηνυμάτων.
2. Αν η ανταλλαγή είναι πολύ συχνή, τότε θα χάνεται πολύτιμος χρόνος για την μετάδοση δεδομένων μέσω του διαδικτύου.
3. Αν γίνεται αραιά, τότε δεν θα υπάρχει σχεδόν κανένα όφελος από την χρήση παράλληλων τεχνικών, αφού κάθε νησί θα είναι σχεδόν ανεξάρτητο.
4. Θα πρέπει να καθοριστεί αν στα μηνύματα θα υπάρχει μόνον το καλύτερο χρωμόσωμα ή τα N καλύτερα χρωμοσώματα.
5. Θα πρέπει να καθοριστεί αν τα χρωμοσώματα που έρχονται με ανταλλαγή μηνυμάτων θα αντικαθιστούν κάποιο χρωμόσωμα στον τοπικό πληθυσμό.

Αλγόριθμοι Κυψέλης

1. Ένα από τα βασικά θέματα που έχουν οι γενετικοί αλγόριθμοι νησιού είναι πως δεν υπάρχει αυστηρά καθορισμένη σειρά επικοινωνίας.
2. Στην υλοποίηση αυτή μπορούμε να θεωρήσουμε πως οι κόμβοι συνδέονται μεταξύ τους σαν να βρίσκονται σε πλέγμα ή σε κάποιο γράφημα.

Αλγόριθμοι Κυψέλης (σχήμα)



Μέγιστος αριθμός γενιών

Τερματισμός όταν εξαντληθεί ένας προκαθορισμένος αριθμός γενιών. Τα προβλήματα σε αυτήν την προσέγγιση είναι τα ακόλουθα:

1. Δεν χρησιμοποιεί καμιά πληροφορία από το πρόβλημα.
2. Πιθανότητα πρόωρης διακοπής του αλγορίθμου.
3. Πιθανή σπατάλη γενιών.

Συνήθως χρησιμοποιείται σε συνδυασμό με άλλες πιο έξυπνες τεχνικές.

Κριτήριο στόχου

1. Τερματισμός όταν το καλύτερο χρωμόσωμα φτάσει σε ένα στόχο, πχ στα τεχνητά νευρωνικά δίκτυα να φτάσουμε στο 0 σφάλμα στο σύνολο εκπαίδευσης.
2. Απαιτεί εκ των προτέρων γνώση του προβλήματος.

Κριτήριο ομοιότητας

1. Τερματισμός όταν το χρωμοσώματα είναι αρκετά κοντά μεταξύ τους.
2. Ποσοτικοποίηση του αρκετά κοντά:

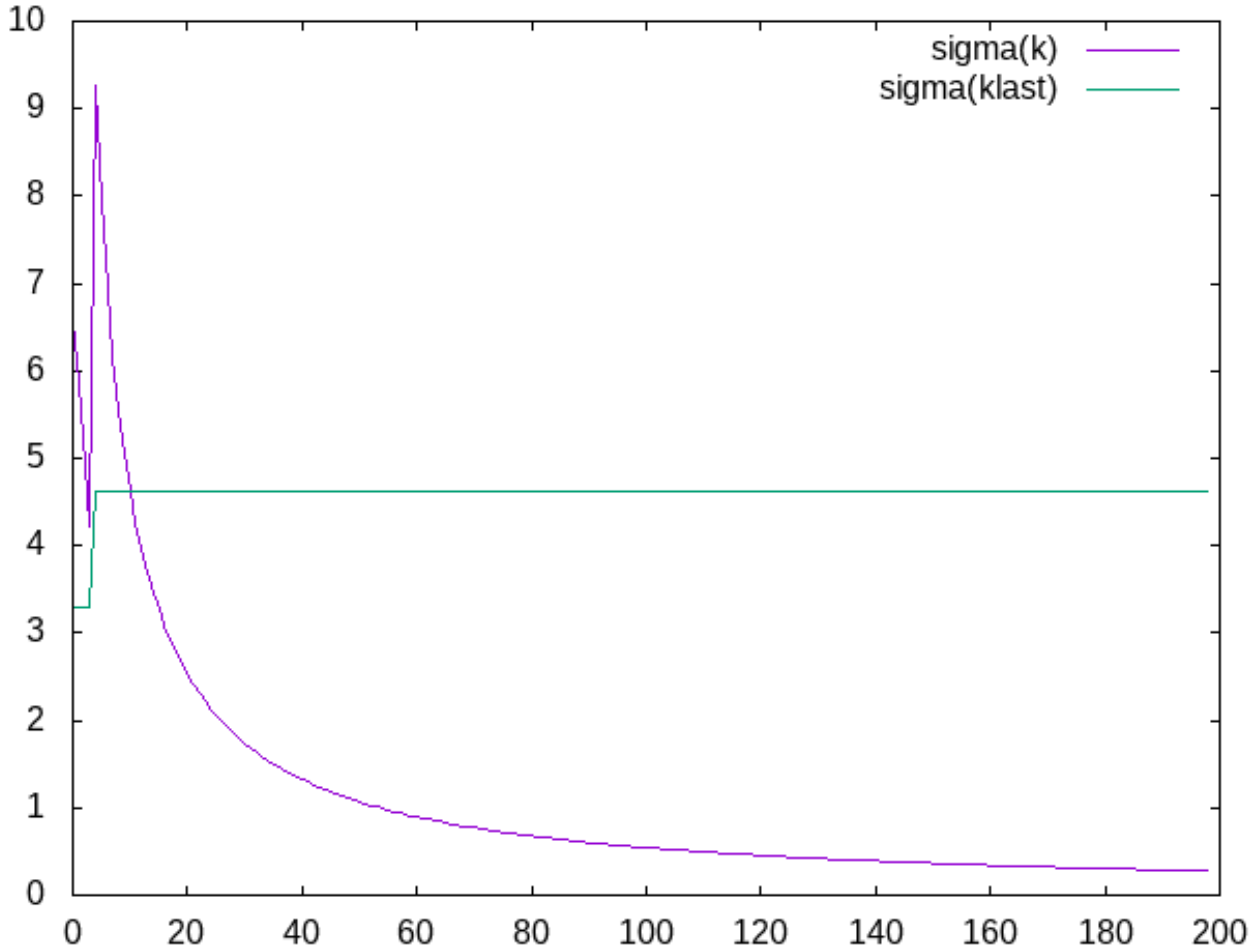
$$\sum_{i=1}^{M} \sum_{j \neq i, j=1} \|x_i - x_j\| \leq e \tag{1}$$

   όπου e ένας προκαθορισμένος μικρός θετικός αριθμός.
3. Είναι αρκετά γενική διαδικασία.
4. Μπορεί να μην επιτευχθεί ποτέ.
5. Έχει αργή υλοποίηση για μεγάλο πλήθος χρωμοσωμάτων.

Διαφορά καλύτερου - χειρότερου

1. Ο αλγόριθμος τερματίζει όταν είναι κοντά το καλύτερο και το χειρότερο μέλος του πληθυσμού.
2. Δηλαδή
$$|f_{max} - f_{min}| \leq e \tag{2}$$
3. Χρησιμοποιείται κυρίως σε δεκαδικές αναπαραστάσεις.
4. Χρησιμοποιείται και σε συνδυασμό με άλλα κριτήρια.

Χρήση διακύμανσης (σχήμα)



Χρήση διακύμανσης (αλγόριθμος)

1. Σε κάθε επανάληψη k υπολογίζεται η διακύμανση της $f\left(x^{*}\right)$.
2. Ορίζουμε αυτήν την διακύμανση ως $\sigma^{(k)}$.
3. Αν δεν υπάρχει αλλαγή για ένα αριθμό γενιών, τότε είναι πιθανόν ότι εντοπίστηκε το ζητούμενο ελάχιστο.
4. Ο αλγόριθμος τερματίζει όταν

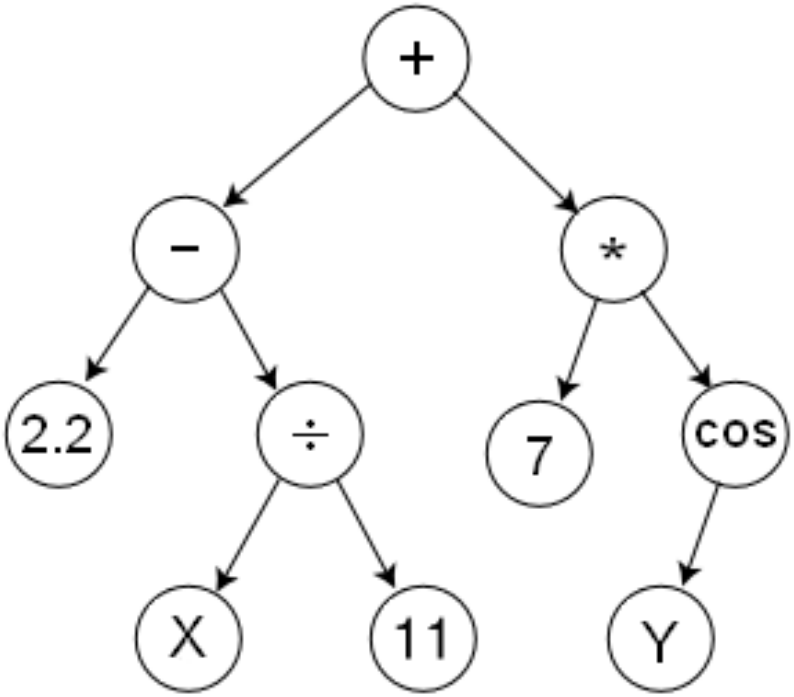
$$\sigma^{(k)} \leq \frac{\sigma^{(klast)}}{2} \tag{3}$$

όπου $klast$ είναι η τελευταία γενιά στην οποία υπήρξε αλλαγή στην καταλληλότητα.

Γενικά στοιχεία

1. Ο γενετικός προγραμματισμός επινοήθηκε από τον Koza και είναι μια ειδική περίπτωση γενετικών αλγορίθμων που στοχεύουν στην παραγωγή προγραμμάτων σε κάποια γλώσσα προγραμματισμού με την χρήση γενετικών τελεστών.
2. Στην αρχική μορφή ο γενετικός προγραμματισμός δημιουργούσε προγράμματα σε μορφή συντακτικών δένδρων
3. Στον γενετικό προγραμματισμό (στην αρχική του μορφή) η διασταύρωση μεταξύ χρωμοσωμάτων θα μπορούσε να γίνει με ανταλλαγή υπόδενδρων και η μετάλλαξη με τυχαία παραγωγή υπόδενδρων και αντικατάσταση στο αρχικό δένδρο.

Γενετικός προγραμματισμός με δένδρα (σχήμα)



$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

Χρησιμοποιούμενα σύνολα

1. Τα τερματικά σύμβολα που συμβολίζονται με το γράμμα T. Στο σύνολο αυτό ανήκουν χαρακτηριστικά που είναι τα φύλλα των δένδρων παραγωγής όπως πχ οι αριθμοί και οι μεταβλητές. Αυτά τα σύμβολα λέγονται τερματικά γιατί δεν μπορούν να αναλυθούν παραπάνω.
2. Τα μη τερματικά σύμβολα που συμβολίζονται με F. Στο σύνολο αυτό ανήκουν στοιχεία που είναι εσωτερικοί κόμβοι στα δένδρα παραγωγής όπως για παράδειγμα αριθμητικοί τελεστές και συναρτήσεις.
3. Τα παραπάνω σύνολα μαζί είναι το σύνολο $C = T \cup F$ των συμβόλων της γραμματικής.

Κλειστότητα

1. Η εφαρμογή οποιασδήποτε συναρτήσεως του συνόλου Φ σε έναν αριθμό από σύμβολα θα επιστρέψει έγκυρη έκφραση.
2. Για παράδειγμα έστω πως $C = \{AND, OR, NOT, x, y, TRUE\}$. Οποιαδήποτε εφαρμογή αυτών των τελεστών στα τερματικά σύμβολα θα δώσει σαν αποτέλεσμα έγκυρη έκφραση.
3. Από την άλλη αν θεωρήσουμε πως έχουμε το σύνολο $C = \{+, -, *, /, x, y, 1, 0\}$ με τα τερματικά σύμβολα $x, y, 1, 0$ και τους μη τερματικούς τελεστές +,-,*,/ θα διαπιστώσουμε πως δεν έχει την ιδιότητα της κλειστότητας, καθώς μπορεί να παραχθεί και η έκφραση $\frac{x}{0}$ όπως και η έκφραση $\frac{x-y}{y-y}$. Και στις δύο περιπτώσεις η παραγόμενη έκφραση οδηγεί σε διαίρεση με 0 και δεν είναι έγκυρη.

Επίλυση προβλημάτων κλειστότητας

1. Για να αποφευχθούν τέτοια προβλήματα εισάγεται η έννοια των προστατευομένων συναρτήσεων.
2. Οι συναρτήσεις αυτές κάνουν έλεγχο για την εγκυρότητα ενός αποτελέσματος και επιστρέφουν διαφορετική τιμή αν παραβιάζεται.
3. Πχ μια προστατευόμενη εκδοχή της διαίρεσης θα μπορούσε να επιστρέφει έναν πολύ μεγάλο αριθμό όταν παρουσιάζεται διαίρεση με το 0. Με αυτόν τον τρόπο το παραγόμενο χρωμόσωμα θα θεωρηθεί άκυρο και δεν θα έχει πολλές πιθανότητες να συμμετάσχει σε διασταύρωση ή μετάλλαξη.

Προστατευόμενες συναρτήσεις (πίνακας)

Συνάρτηση	Προστατευόμενη έκδοση
x/y	Αν $y = 0$, 100 αλλιώς x/y
$log(x)$	Αν $x = 0$, 0 αλλιώς $log(x)$
$\sqrt{x}$	$\sqrt{\lvert x \rvert}$

Επάρκεια

1. Η επάρκεια ενός συνόλου C αναφέρεται στην δυνατότητα του συνόλου να εκφράσει με χρήση των μελών του συνόλου την λύση ενός προβλήματος.
2. Παράδειγμα1: $C = \{AND, OR, NOT, x, y, TRUE\}$ και η λύση ενός προβλήματος είναι AND NOT x AND Y, τότε προφανώς το σύνολο αυτό είναι επαρκές για τον υπολογισμό της λύσης.
3. Παράδειγμα2: "$=\{x, +, -, *, /, 2, 1, 0\}$ και η λύση σε ένα πρόβλημα μάθησης συναρτήσεων είναι η $f(x) = x^3 + x^2 + x$, τότε το σύνολο θεωρείται επαρκές για αυτήν την συνάρτηση καθώς μπορεί να την μάθει.
4. Παράδειγμα3: Αν η λύση σε ένα πρόβλημα είναι η $f(x) = \exp(x)$, τότε το σύνολο δεν είναι επαρκές. Πιθανή εκτίμηση με ανάπτυγμα Talylor: $f'(x) = 1 + x + \frac{1}{2} * x * x + \frac{1}{1+2+2+1} * x * x * x$

Προβλήματα ρουλέττας

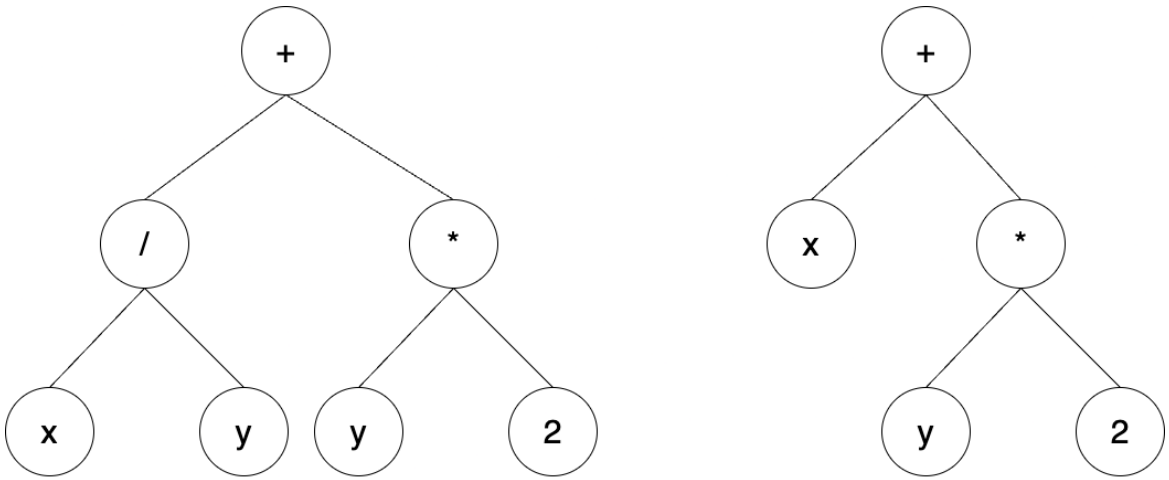
1. Δύσκολη υλοποίηση
2. Απαίτηση για ταξινόμηση καταλληλοτήτων
3. Αργή υλοποίηση
4. Πρόωρη σύγκλιση
5. Στασιμότητα
6. Επίλυση των παραπάνω με τεχνικές κλιμάκωσης

Αρχικοποίηση - μέθοδος full

1. Η μέθοδος αυτή παράγει με αναδρομικό τρόπο δένδρα τα οποία μέχρι βάθους K θα παίρνουν κόμβους μόνο από το σύνολο των μη τερματικών συμβόλων και από εκεί και μετά μόνο από το σύνολο των μη τερματικών συμβόλων.
2. Η μέθοδος αυτή εγγυάται δένδρα σταθερού βάθους στα οποία δεν θα κυριαρχούν τα τερματικά σύμβολα.
3. Για παράδειγμα δεν θα παραχθεί το δένδρο $T = 2$ αφού αυτό περιέχει μόνο τερματικό σύμβολο.

Αρχικοποίηση - μέθοδος Grow

1. Η μέθοδος αυτή παράγει δένδρα με μέγιστο βάθος K
2. Κατά την διάρκεια της δημιουργίας τα σύμβολα λαμβάνονται και από τα τερματικά και από τα μη τερματικά σύμβολα.



3. Σχηματικά:

Full Method

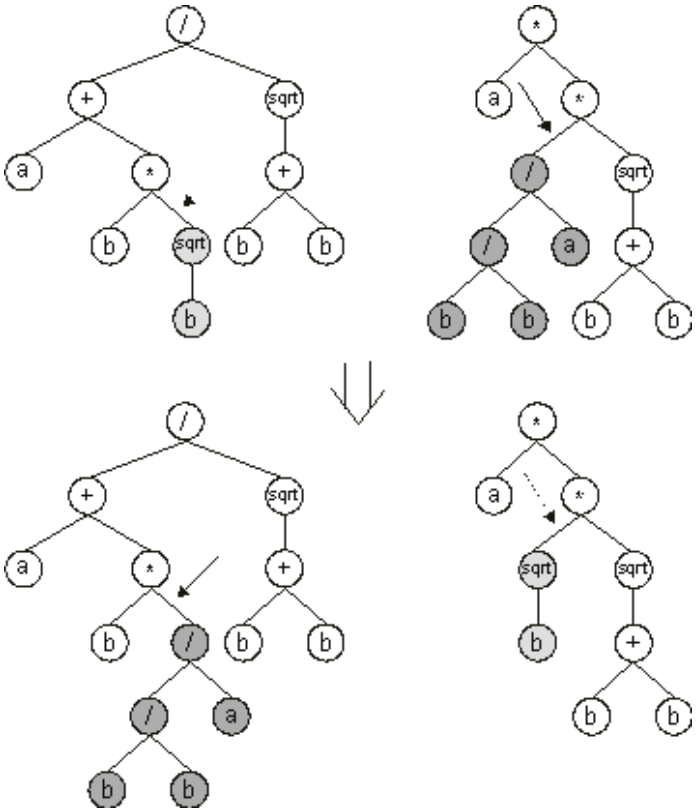
Grow Method

Αρχικοποίηση - **Μέθοδος ramped half-and-half**.

1. Η μέθοδος αυτή παράγει ένα ποσοστό των δένδρων με την μέθοδο full και το υπόλοιπο με την μέθοδο grow.
2. Συνήθως αυτό το ποσοστό είναι 50%.

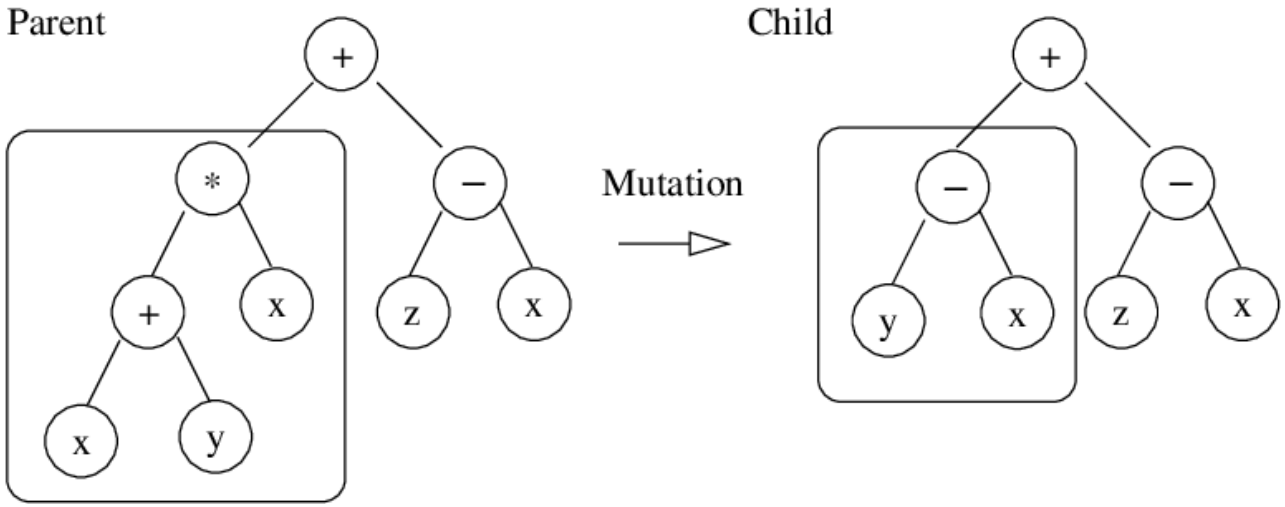
Διασταύρωση δένδρων

Στον γενετικό προγραμματισμό με δένδρα η διασταύρωση συνήθως γίνεται με ανταλλαγή υπόδενδρων μεταξύ συντακτικών δένδρων



Μετάλλαξη

- Η μετάλλαξη μπορεί να πραγματοποιηθεί με τυχαία παραγωγή ενός υπόδενδρου και αντικατάσταση στο συντακτικό δένδρο.



Γενικά

1. Η τεχνική της γραμματικής εξέλιξης (grammatical evolution) είναι μια σχετικά πρόσφατη υλοποίηση γενετικού προγραμματισμού με την χρήση γραμματικών BNF.
2. Σε αυτήν την υλοποίηση γίνεται παραγωγή προγραμμάτων σε οποιαδήποτε γλώσσα με την χρήση της BNF γραμματικής που βρίσκεται πίσω από την προς υλοποίηση γλώσσα.
3. Η μέθοδος αυτή εφαρμόστηκε σε πολλά προβλήματα στην σύγχρονη βιβλιογραφία.

Εισαγωγικές έννοιες

1. Ο αλγόριθμος απαιτεί την γραμματική της αντικειμενικής γλώσσας εκφρασμένη σε BNF μορφή καθώς και την αντίστοιχη συνάρτηση καταλληλότητας.
2. Οι αριθμοί των κανόνων παραγωγής αποκτούν αύξοντα αριθμό για κάθε μη τερματικό σύμβολο της γραμματικής.
3. Τα χρωμοσώματα δεν εκφράζονται σαν συντακτικά δένδρα παραγωγής αλλά σαν διανύσματα ακέραιων αριθμών.
4. Κάθε ακέραιος συμβολίζει και τον αύξοντα αριθμό του κανόνα παραγωγής που θα χρησιμοποιηθεί.
5. Ο αλγόριθμος παραγωγής εκφράσεων ξεκινά από το σύμβολο εκκίνησης της γραμματικής (Start Symbol) και βαθμιαία παράγει την τελική έκφραση αντικαθιστώντας μη τερματικά σύμβολα με τον αντίστοιχο κανόνα παραγωγής.

Επιλογή κανόνα παραγωγής

Η επιλογή του κανόνα παραγωγής γίνεται σε δύο βήματα:

1. Επιλογή του επόμενο στοιχείου V από το χρωμόσωμα
2. Επιλέγεται ο αντίστοιχος κανόνας παραγωγής σύμφωνα με το σχήμα: Rule = V mod NR, όπου NR είναι το πλήθος των κανόνων παραγωγής που διαθέτει το μη τερματικό σύμβολο που θα αντικατασταθεί.

Τερματισμός κανόνα παραγωγής

Η διαδικασία θα συνεχιστεί μέχρι να συμβεί ένα από τα ακόλουθα:

1. Να έχει παραχθεί ένα έγκυρο πρόγραμμα στην αντικειμενική γλώσσα προγραμματισμού. Έγκυρο θεωρείται εκείνο το πρόγραμμα το οποίο δεν έχει άλλα μη τερματικά σύμβολα.
2. Να έχουν τελειώσει τα στοιχεία του χρωμοσώματος και να μην έχει ακόμα παραχθεί έγκυρο πρόγραμμα στην αντικειμενική γλώσσα προγραμματισμού. Σε αυτήν την περίπτωση είτε μπορούμε να θεωρήσουμε πως απέτυχε η διαδικασία και να ανατεθεί μια πολύ μεγάλη καταλληλότητα στο χρωμόσωμα ώστε να μην συμμετάσχει σε διαδικασία διασταύρωσης είτε μπορούμε να εφαρμόσουμε αναδίπλωση (wrapping effect) που σημαίνει πως η διαδικασία της παραγωγής ξεκινά και πάλι από την αρχή του χρωμοσώματος. Στις περισσότερες εφαρμογές 1-2 αναδιπλώσεις είναι αρκετές.

Παράδειγμα γραμματικής

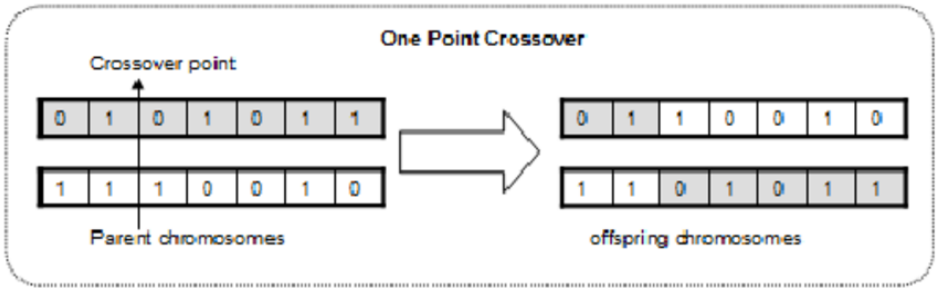
```
<S>::=if(<BEXPR>) CLASS=0 else CLASS=1 (0)
<BEXPR>::=<XLIST><BOOLOP><EXPR>              (0)
         |!(<BEXPR>)                         (1)
         |<XLIST><BOOLOP><EXPR>&<BEXPR>      (2)
         |<XLIST><BOOLOP><EXPR>|<BEXPR>      (3)
<BOOLOP>::= >                                (0)
          | >=                               (1)
          | <                                (2)
          | <=                               (3)

<EXPR>::= (<EXPR><BINARYOP><EXPR>)           (0)
        | <FUNCTION>(<EXPR>)                 (1)
        | <TERMINAL>                         (2)
<BINARYOP>::= +                              (0)
            | -                              (1)
            | *                              (2)
            | /                              (3)
<FUNCTION>::= sin | cos | exp | log           (0-3)
<TERMINAL>::= <XLIST>                        (0)
            | <DIGITLIST>.<DIGITLIST>        (1)
            | (-<DIGITLIST>.<DIGITLIST>)     (2)
 <XLIST>::= x1 | x2 | ...|xD                 (0-D-1)
<DIGITLIST>::=<DIGIT>                        (0)
             | <DIGIT><DIGIT>                (1)
             | <DIGIT><DIGIT><DIGIT>         (2)
<DIGIT>::= 0 | 1 | 2 | 3 | 4 |5 |6 |7 |8 |9    (0-9)
```

Παράδειγμα παραγωγής

string	chromosome	operation
if(<bexpr>) class=0 else class=1	10,65,12,31,28,9,8,6,10,6,2,0,1	
if(<xlist><boolop><expr>) class=0 else class=1	65,12,31,28,9,8,6,10,6,2,0,1	10%4=0
if(x1<boolop><expr>) class=0 else class=1	12,31,28,9,8,6,10,6,2,0,1	65%2=1
if(x1><expr>) class=0 else class=1	31,28,9,8,6,10,6,2,0,1	12%4=0
if(x1><expr><binaryop><expr>) class=0 else class=1	28,9,8,6,10,6,2,0,1	31%3=1
if(x1><function>(<expr)<binaryop><expr>) class=0 else class=1	9,8,6,10,6,2,0,1	28%3=1
if(x1>cos(<expr>)<binaryop><expr>) class=0 else class=1	8,6,10,6,2,0,1	9%4=1
if(x1>cos(<terminal><binaryop><expr) class=0 else class=1	6,10,6,2,0,1	8%3=2
if(x1>cos(<xlist>)<binaryop><expr) class=0 else class=1	10,6,2,0,1	6%3=0
if(x1>cos(x0)<binaryop><expr>) class=0 else class=1	6,2,0,1	10%2=0
if(x1>cos(x0)*<expr>) class=0 else class=1	2,0,1	6%4=2
if(x1>cos(x0)*<terminal>) class=0 else class=1	0,1	2%3=2
if(x1>cos(x0)*<xlist>) class=0 else class=1	1	0%3=0
if(x1>cos(x0)*x1) class=0 else class=1		1%2=1

Διασταύρωση σε Γραμματική Εξέλιξη

1. Συνήθως γίνεται με διασταύρωση ενός σημείου.
2. Σχηματικό παράδειγμα:



Μετάλλαξη στην Γραμματική Εξέλιξη

1. Για κάθε στοιχείο κάθε χρωμοσώματος επιλέγεται ένας τυχαίος αριθμός $r \in [0, 1]$.
2. Το συγκεκριμένο στοιχείο αλλάζει τυχαία σε κάποιον άλλο ακέραιο αριθμό αν $r \leq p_m$, όπου p_m η πιθανότητα μετάλλαξης.

Γενικά στοιχεία

1. Δυναμική δημιουργία αρχιτεκτονικής ΤΝΔ με χρήση Γραμματικής Εξέλιξης.
2. Η Γραμματική παράγει τόσο τις συνδέσεις μεταξύ των κόμβων όσο και τις βέλτιστες τιμές των βαρών.

Χρήση γενετικών αλγορίθμων

1. Χρωμοσώματα: Τυχαίοι δεκαδικοί πίνακες. Κάθε πίνακας είναι ένα σύνολο παραμέτρων για το ΤΝΔ.
2. Καταλληλότητα η συνάρτηση σφάλματος:

$$\mathrm{E}=\sum \left(N\left(x_i\right)-y_i\right)^2$$

3. Διασταύρωση, Δημιουργία χρωμοσωμάτων με τον τύπο:

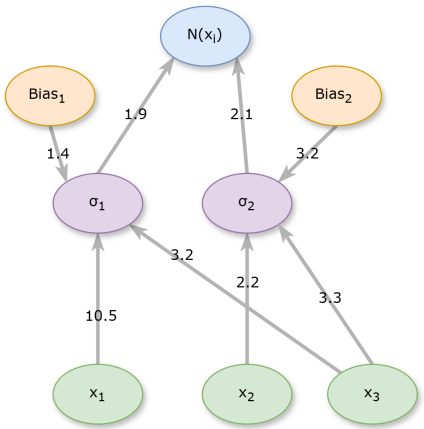
$$x=p_1+\alpha\left(p_1-p_2\right),\ \alpha\in[-0.5,1.5]$$

4. Μετάλλαξη: Αλλαγή των στοιχείων κάθε χρωμοσώματος με τυχαίες μεταβολές σε ένα διάστημα τιμών.

Γραμματική BNF κατασκευαζόμενου ΤΝΔ

S:=<Expression> (0)
<Expression>::=<Neuron> (0)
| <Neuron> + <Expression> (1)
<Neuron>::=<Number>*sig(<Sum>+<Number>) (0)
<Sum>::= <Number>*<Xlist> (0)
| <Sum>+<Sum> (1)
<Xlist>::= x1 (0)
| x2 (1)
.....
| xd (d-1)
<Number>::= (<Dlist>.<Dlist>) (0)
| (-<Dlist>.<Dlist>) (1)
<Dlist>::= <Digit> (0)
| <Digit><Dlist> (1)
<Digit>::= 0 (0)
| 1 (1)
.....
| 9 (9)

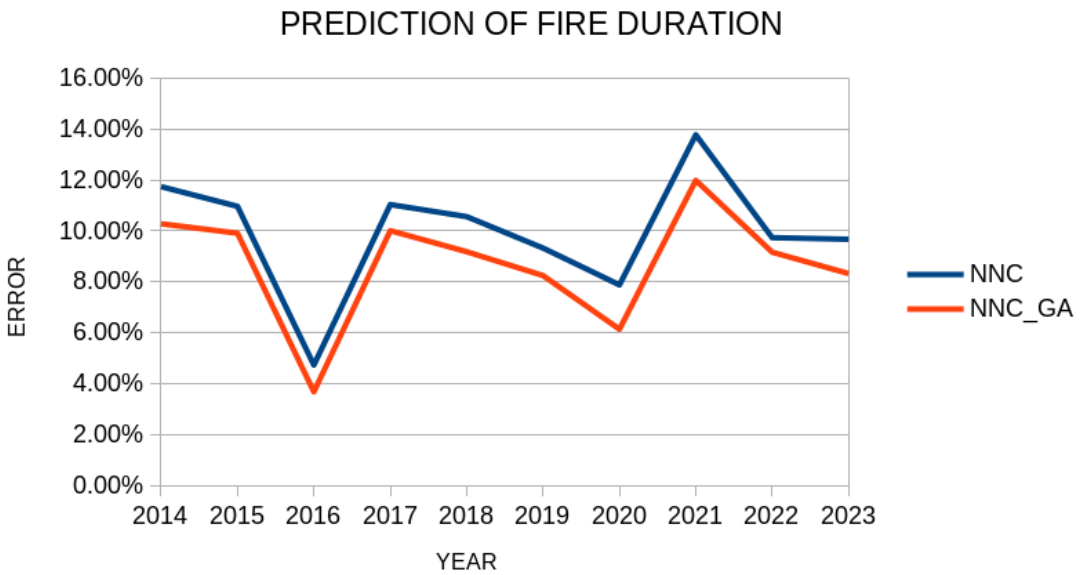
Παράδειγμα κατασκευής

1. 

2. $N(x) = 1.9 \times sig\left(10.5x_1 + 3.2x_3 + 1.4\right) + 2.1 \times sig\left(2.2x_2 - 3.3x_3 + 3.2\right)$

Βελτίωση με χρήση βελτιστοποίησης με Γενετικούς Αλγορίθμους

1. Εφαρμογή Γενετικών Αλγορίθμων για την βελτιστοποίηση των παραγόμενων παραμέτρων από την Γραμματική Εξέλιξη.
2. Παράδειγμα βελτίωσης σε πρακτικό παράδειγμα:



Σύνοψη

- Παράλληλοι αλγόριθμοι
- Τερματισμός
- Γενετικός προγραμματισμός
- Γραμματική εξέλιξη

Βιβλιογραφία I


Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.


Freund, Y. and Schapire, R. E. 1998. Large margin classification using the perceptron algorithm. In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT' 98). ACM Press.