ΓΕΩΡΓΙΟΣ ΣΕΪΜΕΝΗΣ – Π19204

πανεπιστημιο πειραιωσ – τμημα πληροφορικησ | ΜΑΪΟΣ ~ ΙΟΥΝΙΟΣ 2022

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ & ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ

εργασια στουσ γενετικουσ αλγοριθμουσ

Περιεχόμενα

[Εκφώνηση 2](#_Toc104043150)

[Constraint Satisfaction Problem 2](#_Toc104043151)

[Υλοποίηση σε C# 3](#_Toc104043152)

[Η λύση της εργασίας 3](#_Toc104043153)

[Πώς αναπαρίστανται οι λύσεις; 3](#_Toc104043154)

[Πώς λειτουργεί η συνάρτηση καταλληλότητας; 4](#_Toc104043155)

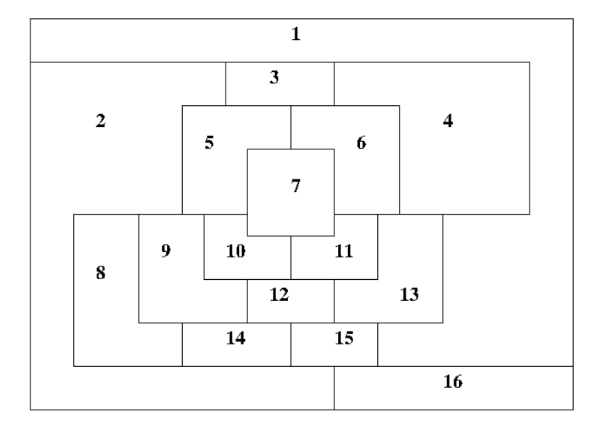
[Πως ορίζεται ο αρχικός πληθυσμός; 4](#_Toc104043156)

[Πώς γίνεται η αναπαραγωγή; 4](#_Toc104043157)

# Εκφώνηση

## Ως εργασία, ζητήθηκε να βρεθεί λύση στην άσκηση τύπου «Constraint Satisfaction Problem», με τη χρήση γενετικών αλγορίθμων.

Η άσκηση είχε τον παρακάτω γράφο με διάφορα κουτάκια να ενώνονται με τυχαία σειρά. **Τα κουτάκια πρέπει να χρωματιστούν με τρόπο τέτοιον, ώστε κανένα κουτάκι να έχει ίδιο χρώμα με αυτά που γειτονεύει.**

****

Οι φοιτητές έπρεπε να:

* Χρησιμοποιήσουν γενετικούς αλγορίθμους.
* Ορίσουν αρχικό πληθυσμό με τυχαίο αριθμό πλήθους.
* Φτιάξουν συνάρτηση καταλληλότητας.
* Φτιάξουν συνάρτηση επιλογής γονέων και συνάρτηση διασταύρωσης ενός σημείου.
* Φτιάξουν συνάρτηση μετάλλαξης και ανανέωσης πληθυσμού.

# Υλοποίηση σε C#

## Η λύση της εργασίας έγινε σε εφαρμογή τύπου Windows Forms, ώστε να υπάρχει γραφικοποίηση στον χρωματισμό του γράφου. Ο χρήστης, με το πάτημα του κουμπιού, μπορεί να βλέπει το πώς χρωματίζονται τα κουτάκια.

Το συγκεκριμένο πρόβλημα CSP (Constraint Satisfaction Problem) έχει πολλές βέλτιστες λύσεις, οπότε ο στόχος μας είναι να φτάσουμε σε μία από αυτές. Στην κύρια ροή, το πρόγραμμα παράγει έναν αρχικό πληθυσμό, μέσα στον οποίο υπάρχουν χρωμοσώματα με τυχαίες λύσεις. Χρησιμοποιώντας την συνάρτηση καταλληλότητας, γίνεται ταξινόμηση των χρωμοσωμάτων, κρατώντας, ως πρώτο, το καλύτερο χρωμόσωμα (δηλαδή αυτό που είναι πιο κοντά σε μία βέλτιστη λύση). Αυτό που καταφέρνουμε, κρατώντας το χρωμόσωμα αυτό, είναι ότι το πρόγραμμα παράγει κι άλλες λύσεις, βασιζόμενο σε αυτό. Έτσι, το πρόγραμμα φτάνει γρηγορότερα σε μία βέλτιστη λύση, χρησιμοποιώντας ταυτόχρονα μετάλλαξη, σε περίπτωση που κωλύεται. **Το πρόγραμμα δεν σταματάει, μέχρι να βρει μία βέλτιστη λύση.**

### Πώς αναπαρίστανται οι λύσεις;

Οι λύσεις πρέπει να αναπαρασταθούν σε δυαδική μορφή. Από τη στιγμή, λοιπόν, που έχουμε **τέσσερα πιθανά χρώματα,** κατανείμονται ως εξής:

* 00 🡪 μπλε
* 01 🡪 κόκκινο
* 10 🡪 πράσινο
* 11 🡪 κίτρινο

Έπειτα, έχουμε 16 κουτάκια να χρωματίσουμε. Αυτό σημαίνει ότι έχουμε 16 × 2 = 32 bits για να ορίσουμε μία λύση. Οπότε, μία ομάδα των δύο bit καθορίζει και τη θέση κάθε κουτιού. Δηλαδή η πρώτη δυάδα καθορίζει το χρώμα του πρώτου κουτιού, η δεύτερη δυάδα του δεύτερου κουτιού, και ούτω καθεξής. Για παράδειγμα, η λύση 00000000000000000000000000000000 σημαίνει ότι όλα τα κουτάκια έχουν το χρώμα μπλε. Ωστόσο, η λύση 11010101010101010101010101010101 δίνει στο πρώτο κουτάκι το χρώμα κίτρινο και σε όλα τα υπόλοιπα το χρώμα κόκκινο.

### Πώς λειτουργεί η συνάρτηση καταλληλότητας;

Χρησιμοποιώντας την κλάση Genome, αναπαρίσταται ένα χρωμόσωμα. Το χρωμόσωμα έχει δύο ιδιότητες: την λύση (όπως αυτή επεξηγήθηκε παραπάνω) και το σκορ καταλληλότητας. Η λύση έχει τύπο λίστας αλφαριθμητικών και **το σκορ καταλληλότητας είναι ένας ακέραιος αριθμός**. **Αυτός ο αριθμός ανεβαίνει κατά 1, όταν ένα κουτάκι έχει διαφορετικό χρώμα με κάποιον γείτονά του**. Άρα, όταν όλα τα κουτάκια έχουν το ίδιο χρώμα, το σκορ καταλληλότητας είναι 0. Αν όλα τα κουτάκια έχουν διαφορετικά χρώματα με τους γείτονές του, ο αριθμός φτάνει στο μέγιστο, που είναι το 42. **Οπότε, μία βέλτιστη λύση θα πρέπει να έχει σκορ καταλληλότητας 42.**

### Πως ορίζεται ο αρχικός πληθυσμός;

Ο καλύτερος αριθμός, που ταιριάζει στον αρχικό πληθυσμό, είναι 15 χρωμοσώματα. Βέβαια, ο χρήστης μπορεί να πειραματιστεί, αλλάζοντάς τον με την χρήση ενός numeric up-down. **Περιέργως, η πρόσθεση χρωμοσωμάτων στον αρχικό πληθυσμό, δεν σημαίνει απαραίτητα ότι το πρόγραμμα θα φτάσει στην εύρεση μιας βέλτιστης λύσης γρηγορότερα.** Υπενθυμίζεται ότι βασιζόμαστε κυρίως στην τύχη. Έχοντας περισσότερα χρωμοσώματα, ναι μεν αυξάνει τις πιθανότητες να βρεθεί βέλτιστη λύση, αλλά σημαίνει και περισσότερη κατανάλωση υπολογιστικής ισχύος.

### Πώς γίνεται η αναπαραγωγή;

Στην κύρια ροή, αφού παραχθούν οι λύσεις, τις ταξινομούμε με φθήνουσα σειρά βάσει του σκορ καταλληλότητάς τους. Κρατώντας απείραχτο το καλύτερο χρωμόσωμα, αναπαράγουμε λύσεις βάσει αυτού του χρωμοσώματος, διασταυρώνοντας σε ένα σημείο με τις υπόλοιπες λύσεις. Το σημείο στο οποίο γίνεται διασταύρωση είναι πάντοτε τυχαίο. Έχοντας, προηγουμένως, ταξινομήσει τα χρωμοσώματα, **φτάνουμε γρηγορότερα στην εύρεση μία λύσης, μια που διασταυρώνουμε πάντοτε τις καλύτερες λύσεις.**

### Πόσο συχνά γίνεται κάποια μετάλλαξη;

**Μία μετάλλαξη έχει πιθανότητα 30% να συμβεί σε έναν απόγονο.** Η μετάλλαξη δεν γίνεται στο καλύτερο χρωμόσωμα. Αυτή η μετάλλαξη συμβαίνει μόνο σε ένα bit. Όταν λέμε «30% πιθανότητα μετάλλαξης», εννοούμε ότι υπάρχει 30% πιθανότητα να αλλάξει ένα τυχαίο bit και 70% πιθανότητα να παραμείνει ίδια η λύση. **Βέβαια, υπάρχει η πιθανότητα, στην κύρια ροή, το πρόγραμμα να κωλύεται σε ένα τοπικό μέγιστο (local maxima).** Για να αποφύγουμε αυτό το ενδεχόμενο, ελέγχουμε αν το καλύτερο χρωμόσωμα παραμένει το ίδιο, για παραπάνω από οκτώ φορές στη σειρά. Σε αυτήν την περίπτωση παίρνουμε όλα τα χρωμοσώματα και κάνουμε **ολική μετάλλαξη με 50% πιθανότητα**.Αυτό σημαίνει ότι για κάθε bit υπάρχει 50% πιθανότητα να αλλάξει ή να παραμείνει το ίδιο.