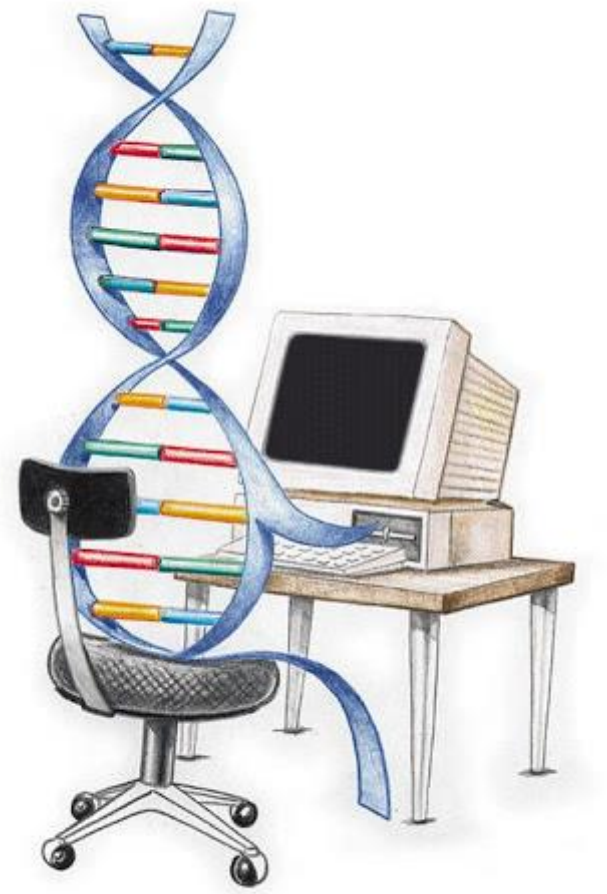
Υπολογιστική νοημοσύνη

Report γεννητικών αλγορίθμων

Λιάπης Σάββας 57403



Υπεύθυνοι Διδάσκοντες :

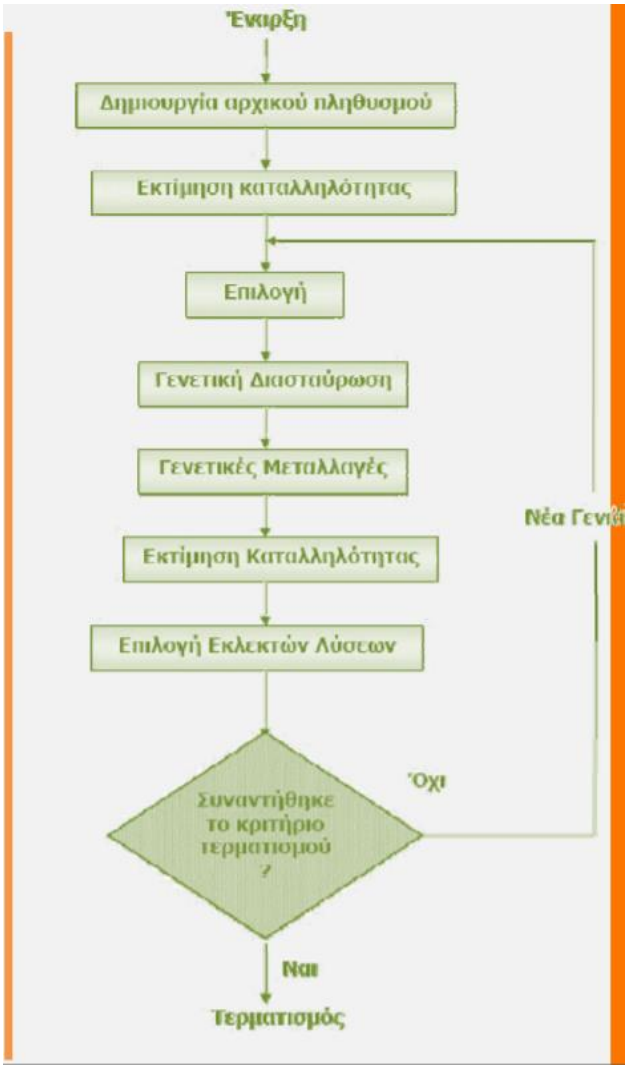
Ιωάννης Μπούταλης

Σπυρίδων Πλακιάς

Απρίλιος 2020

Αρχές των Γενετικών Αλγορίθμων

Οι γενετικοί αλγόριθμοι είναι αλγόριθμοι αναζήτησης λύσεων, που βασίζονται στους μηχανισμούς της φυσικής επιλογής, της γενετικής και της εξέλιξης. Βασικά δομικά στοιχεία για την ανάλυση των προβλημάτων που λύνονται με γενετικούς αλγόριθμους είναι στοιχεία όπως τα χρωμοσώματα που αποτελούν οργανικές «συσκευές», που κωδικοποιούν τη δομή των ζωντανών οργανισμών, καθώς και διαδικασίες φυσικής επιλογής, όπως η αναπαραγωγή, η μετάλλαξη και η διασταύρωση. Η διαδικασία της φυσικής επιλογής προκαλεί τα χρωμοσώματα που κωδικοποιούν επιτυχείς δομές να αναπαράγονται συχνότερα από τα άλλα. Η μετάλλαξη μπορεί να έχει ως αποτέλεσμα τα χρωμοσώματα των απογόνων να είναι διαφορετικά από αυτά των βιολογικών τους γονιών. Η διαδικασία της διασταύρωσης μπορεί επίσης να προκαλέσει διαφορετικά χρωμοσώματα στους απογόνους, συνδυάζοντας υλικό από τα χρωμοσώματα των δύο γονιών τους. Αυτά τα χαρακτηριστικά της φυσικής εξέλιξης ενέπνευσαν την ανάπτυξη των Γ.Α. Δίπλα φαίνεται το λογικό διάγραμμα ενός γενετικού αλγορίθμου



Περιγραφή της Άσκησης

Το ΑΜ μου είναι περιττός αριθμός. Συνεπώς μου έχει ανατεθεί η δεύτερη εργαστηριακή άσκηση **Το πρόβλημα του διαχωρισμού καρτών**.

Εκφώνηση :

Έχουμε 15 κάρτες αριθμημένες από το 1 μέχρι το 15. Θέλουμε να τις χωρίσουμε ισομερώς σε τρεις στοίβες, έτσι ώστε το άθροισμα των καρτών της πρώτης να είναι 49, το άθροισμα των καρτών της δεύτερης να είναι 33 και το γινόμενο της τελευταίας στοίβας να είναι 12600. Να υλοποιηθεί πρόγραμμα σε MATLAB το οποίο να λύνει το παραπάνω πρόβλημα χρησιμοποιώντας γενετικούς αλγόριθμους.

Διαδικασία επίλυσης:

Για την κωδικοποίηση του προβλήματος θα χρησιμοποιήσω ένα χρωμόσωμα 15 μοναδικών γονιδίων με τιμές (1-15). Με άλλα λόγια κάθε χρωμόσωμα θα είναι ένα διάνυσμα 15 θέσεων και θα περιέχει τιμές από το 1 εως το 15 εκ των οποών η κάθε μία θα εμφανίζεται μία φορά.

Αρχικοποίηση πληθυσμού

Η αρχικοποίηση του πληθυσμού είναι αρκετά απλή. Δημιουργούμε μία συνάρτηση η οποία θα πάρει ως ορίσματα τον αριθμό των γονιδίων (NVARS) που θα περιέχει κάθε χρωμόσωμα και την συνάρτηση προσαρμογής. Ως έξοδο θα λαμβάνουμε τον αρχικό πληθυσμό. Ο πληθυσμός των χρωμοσωμάτων καθορίζεται από τις options οι οποίες θα καθοριστούν είτε από έναν συνοδευτικό κώδικα είτε στο ειδικό παράθυρο που καλείται από το command window με την εντολή optimtool. Η έξοδος είναι στην ουσία ένα πίνακας που έχει έναν πληθυσμό μεγέθους totalPopulationSize από 15-μελή διανύσματα καθένα από τα οποία περιέχουν μία διάταξη μοναδικών αριθμών 1-15 . Με μία επανάληψη γεμίζουμε αυτά τα κελία με τυχαίο τρόπο. Εξασφαλίζουμε ωστόσο ώστε σε κάθε δεκαπενταμελές κελί κάθε αριθμός από το 1-15 θα εμφανίζεται μία ακριβώς φορά με την συνάρτηση randperm(αριθμός γονιδίων).

```
function InitialPopulation = cards_create_permutations(NVARS,FitnessFcn,...
    options)
%the total population will be determined in the options
totalPopulationSize = sum(options.PopulationSize);
%the NVARS will be determinde by the options and is the number of the
%the varieables of the problem (genes)
n =NVARS;
%the initial population is a set of as many chromosomes as
%the population size
InitialPopulation = cell(totalPopulationSize,1);
for i = 1:totalPopulationSize
    %with the randperm function we ensure that every number 1-15 will be
    %used one and only time.
    InitialPopulation{i} = randperm(n);
end
```

Καθορισμός Συνάρτησης Προσαρμοστικότητας :

Ένα από τα πιο ζωτικά -αν όχι το πιο ζωτικό χαρακτηριστικό ενός γεννητικού αλγορίθμου είναι η συνάρτηση προσαρμοστικότητας (fitness function). Η συνάρτηση προσαρμοστικότητας προσομοιώνει το περιβάλλον των χρωμοσωμάτων και βάσει αυτής καθορίζεται πόσο καλό είναι το κάθε χρωμόσωμα ώστε να έχει περισσότερες πιθανότητας να επιλέγει. Στο συγκεκριμένο πρόβλημα θα μπορούσαμε να επιτύχουμε αυτά που μας ζητάει η άσκηση τοποθετώντας τους αριθμούς που θέλουμε να βγάζουν άθροισμα 49 στις πρώτες 5 θέσεις του κάθε χρωμοσώματος, τους αριθμούς που θέλουμε να δίνουν άθροισμα 33 στις επόμενες 5 θέσεις και τους αριθμούς, των οποίων το γινόμενο θέλουμε να δίνει 12600 στις τελευταίες 5 θέσεις. Συνεπώς το κάθε χρωμόσωμα θέλουμε να αξιολογείται κατά πόσο οι αριθμοί που βρίσκονται στις πρώτες θέσεις δίνουν άθροισμα κοντά ή ίσο του 49. Οπότε αθροίζουμε τους πρώτους 5 αριθμούς με την συνάρτηση sum και βρίσκουμε την απόλυτη διαφορά τους από

τον αριθμό 49 με την συνάρτηση abs. Ούτω καθεξής και για τους αριθμούς που βρίσκονται στις επόμενες 5 θέσεις και για αυτούς που βρίσκονται στις τελευταίες 5.Η τελική τιμή της συνάρτησης είναι ένα άθροισμα των 3 «αποστάσεων από τους στόχους» όσο μικρότερο το άθροισμα τόσο καλύτερο το χρωμόσωμα με βέλτιστη τιμή το 0. Η συνάρτηση cards_fitness επιστρέφει εν τέλη ένα διάνυσμα fitness_Value που περιέχει τις «αξίες» του κάθε χρωμοσώματος.

```
1   function fitnessValue = cards_fitness(x)
2
3   %the fitness Values of each chromosome is contained into the vector fitness
4   %Value wich is empty in the begining.
5   fitnessValue=zeros(size(x,1),1);
6
7   %the loop runs as many times as the population of the chromosomes
8   for i=1:size(x,1)
9       %we check every chromosome of the population
10      CurrentChromosome = x{i};
11      %finding the sum of the first 5, the second 5 and the third 5 genes
12      sum49=sum(CurrentChromosome(1:5));
13      sum33=sum(CurrentChromosome(6:10));
14      prod12600=prod(CurrentChromosome(11:15));
15
16      %cheking how close is the sum of the first 5 numbers to the number 49
17      score1=abs(49-sum49);
18      %cheking how close is the sum the second 5 numbers to the number 33
19      score2=abs(33-sum33);
20      %cheking how close is the product of the last five numbers to the 12600
21      score3=abs(12600-prod12600);
22
23      %fitness Value of each chromosome is the sum of the 3 scores
24      fitnessValue(i)= score1+score2+score3;
25  end
```

Επιλογή των γονέων :

Η επιλογή των γονέων θα γίνει με τυχαίο τρόπο μέσω μίας συνάρτησης των γενετικών αλγορίθμων (εμείς θα επιλέξουμε συνάρτηση ρουλέτας). Κάθε μέλος του πληθυσμού θα έχει την ευκαιρία να γίνει γονέας όμως ανάλογα με την καταλληλόλητα του θα έχει περισσότερες πιθανότητες (Προσοχή, αυτό δεν αποκλείει την επιλογή ακατάλληλων γονέων σε μία γενιά καθώς η επιλογή γίνεται τυχαία. Ωστόσο γνωρίζουμε ότι με το πέρασμα των γενεών θα παραμείνουν αυτοί που έχουν περισσότερες πιθανότητες).

Εκτέλεση διασταυρώσεων :

Ο αλγόριθμος βελτιστοποιείται με την συνεχή εκτέλεση διασταυρώσεων. Η διαδικασία αυτή πραγματοποιείται με την συνάρτηση cards_crossover_permutations. Η συνάρτηση αυτή αρχικοποιεί ένα σύνολο crossoverKids, διαιρώντας τον αριθμό των γονέων δια 2 (οι γονείς εισάγονται σαν όρισμα στην συνάρτηση), στο οποίο θα αποθηκευτούν τα χρωμοσώματα των νέων παιδιών-μελών του πληθυσμού. Έπειτα επιλέγουμε έναν από τους γονείς. Από το

χρωμόσωμα του γονέα αυτού επιλέγουμε 2 γονίδια. Είναι σημαντικό να τηρηθεί η μοναδικότητα των 15 γονιδίων του κάθε χρωμοσώματος, οπότε για να μην παραβιαστεί αυτό, πρώτα φροντίζουμε το δεύτερο γονίδιο να βρίσκεται σε διαφορετική περιοχή του χρωμοσώματος από το πρώτο ώστε να μην υπάρχει η πιθανότητα να επιλεγεί το ίδιο γονίδιο και έπειτα αλλάζουμε αντικαθιστούμε το ένα γονίδιο στην θέση του άλλου με την συνάρτηση fliplr το νέο παιδί που προκύπτει το αποθηκεύουμε στο σύνολο με τα χρωμοσώματα των νέων παιδιών. Αυτή η διαδικασία επαναλαμβάνεται τόσες φορές όσα είναι και τα παιδιά. Η συνάρτηση επιστρέφει το σύνολο των χρωμοσωμάτων των νέων παιδιών crossoverKids.

```
function crossoverKids  = cards_crossover_permutation(parents,options,...
    NVARS,FitnessFcn,thisScore,thisPopulation)

%one child comes from 2 parents so the nuber of children will be
%half of the selected parents number
nunmberOfKids = length(parents)/2;
%crossoverKids will be set of new chromosomes (the kids).
crossoverKids = cell(nunmberOfKids,1);
index = 1;

for i=1:nunmberOfKids
    %we choose one of every 2 parents
    parent = thisPopulation{parents(index)};
    %the index jumps to the next 2 parents and chooses one
    index = index + 2;

    %we choose 2 genes gene1 and gene2 randomly of the chosen parent
    gene1 = ceil((length(parent) -1) * rand);
    %we should reserve that the gene 2 will be in greater place of
    %gene1
    gene2 = gene1 + ceil((length(parent) - gene1- 1) * rand);
    %we create the new childs choromosome which is identical to the parent
    child = parent;
    %we perform the flip of the two genes and get the new chromosome
    child(gene1:gene2) = fliplr(child(gene1:gene2));
    %the new child gets stored in the vector with all the new children
    crossoverKids{i} = child;
end
```

Ύπαρξη μεταλλάξεων :

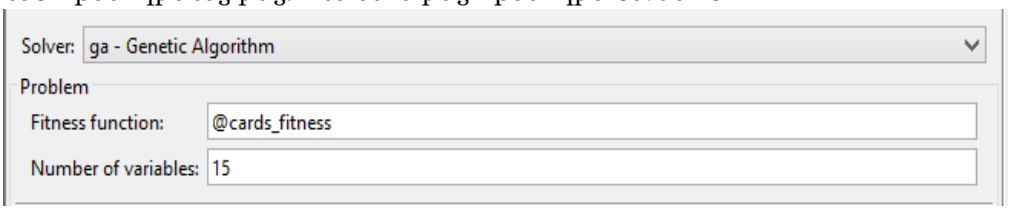
Πρέπει να έχουμε κατά νου ότι οι γεννητικοί αλγόριθμοι μιμούνται την φυσική επιλογή που υπάρχει και στην φύση και όπως γνωρίζουμε σε αυτή την διαδικασία μπορούν να προκύψουν μεταλλάξεις δηλαδή μια τροποποιημένη διάταξη των γονιδίων σε ένα χρωμόσωμα. Μια μετάλλαξη μπορεί να είναι είτε καλή είτε κακή. Αν είναι καλή έχει πιθανότητες να διαιωνιστεί, ειδάλλως είτε αυτή είτε οι απόγονοί της δεν θα επιβιώσουν. Αυτό που μένει για την επίλυση του προβλήματός μας είναι η δημιουργία μίας συνάρτησης που θα πραγματοποιεί μεταλλάξεις. Στην συγκεκριμένη συνάρτηση επιλέγουμε από έναν γονέα (έχουμε εισάγει σαν όρισμα τα χρωμοσώματα των γονέων) 2 τυχαία γονίδια και τα αλλάζουμε θέση. Αυτό επαναλαμβάνεται για το σύνολο των γονέων. Η συνάρτηση με την ολοκλήρωσή της εξάγει το σύνολο των μεταλλαγμένων παιδιών.

```
function mutatedChildren = cards_mutate_permutation(parents ,options,...
    NVARS,FitnessFcn, state, thisScore,thisPopulation,mutationRate)

% Here we swap two elements of the permutation
mutatedChildren = cell(length(parents),1);

for i=1:length(parents)
    %we get every parent
    parent = thisPopulation{parents(i)};
    %from the parents chromosome we choose two random genes and swap them
    p = ceil(length(parent) * rand(1,2));
    child = parent;
    child(p(1))= parent(p(2));
    child(p(2))= parent(p(1));
    mutatedChildren{i} = child;
end
```

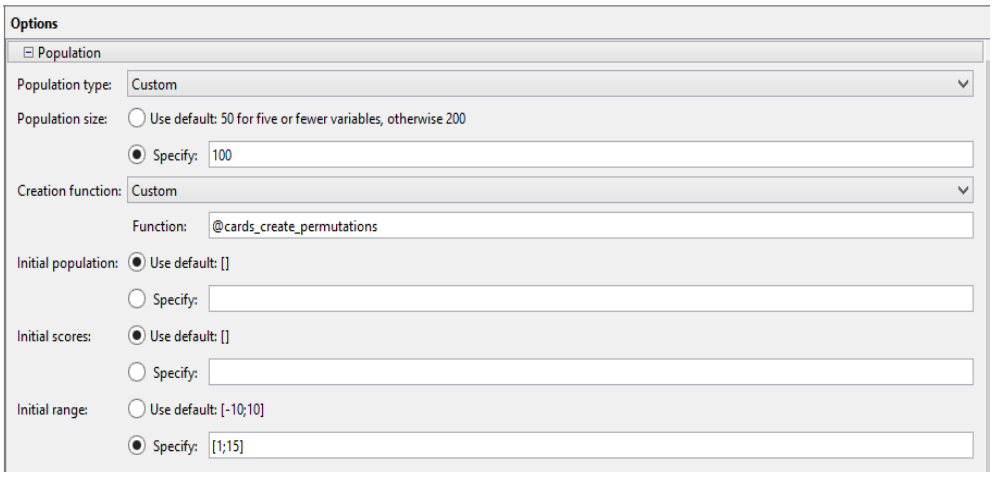
Εκτέλεση του αλγορίθμου για την υλοποίηση της άσκησης :

Για να εκτελέσουμε έναν γεννητικό αλγόριθμο μπορούμε να καλέσουμε το παράθυρο εκτέλεσης πληκτρολογώντας στο command window του matlab optimtool. Τότε εμφανίζεται ένα παράθυρο και ρυθμίζουμε τις παραμέτρους ως εξής :

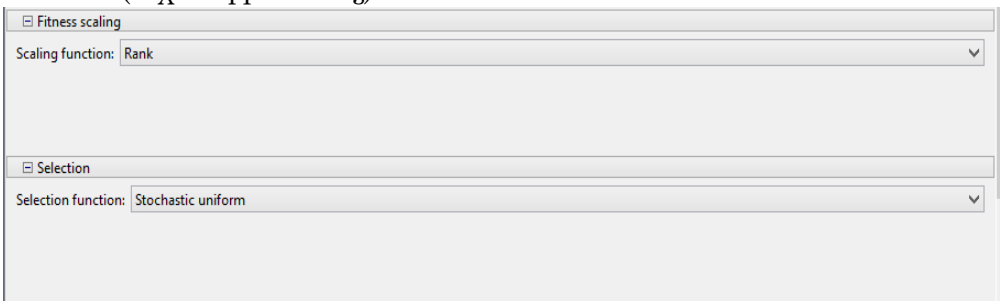
1. επιλέγουμε ως solver ga-Genetic Algorithm
2. ακριβώς από κάτω στο πεδίο Fitness Function βάζουμε το όνομα τις fitness που έχουμε δημιουργήσει με την μορφή ανώνυμης συνάρτησης (με @ από μπροστά). Στην δικό μας πρόβλημα @cards_fitness
3. προσδιορίζουμε τον αριθμό των μεταβλητών-γονιδίων ενός χρωμοσώματος του προβλήματος μας. Στο δικό μας πρόβλημα είναι 15



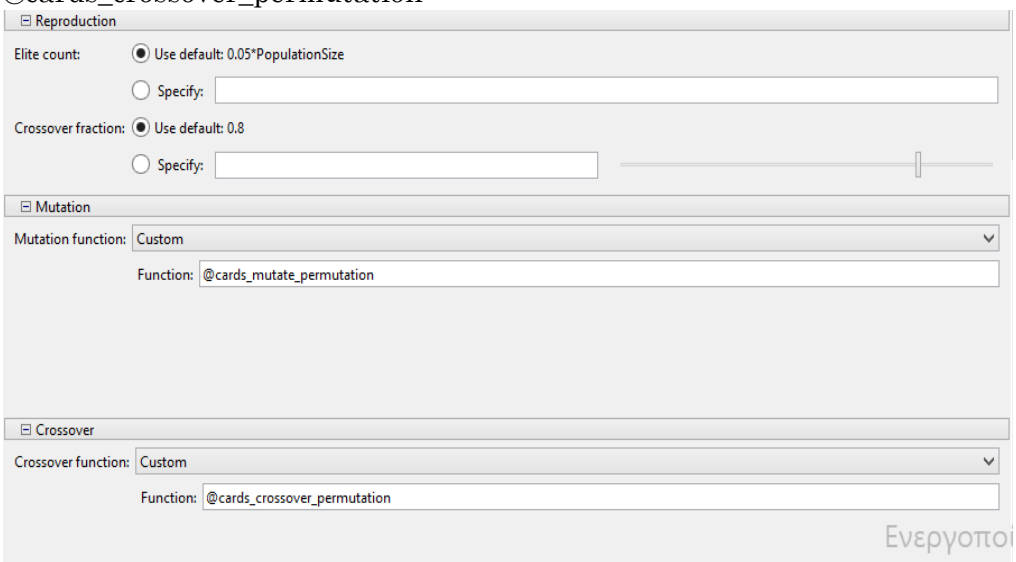
4. πηγαίνουμε στην δεξιά στήλη Options και ρυθμίζουμε τις παραμέτρους
5. Population:
    - Population type : Custom
    - Population size : προσδιορίζουμε το μέγεθος του πληθυσμού που χρειάζεται για το πρόβλημα . Ας πούμε 100.
    - Creation function : επιλέγουμε custom και στο πεδίο Function γράφουμε το όνομα της συνάρτησης αρχικοποίησης του πληθυσμού, που έχουμε φτιάξει. Στο δικό μας πρόβλημα @cards_create_permutations
    - Τα υπόλοιπα πεδία του παραρτήματος Population μπορούμε να τα αφήσουμε deafault η να τα τροποποιήσουμε όπως θέλουμε. Στο πεδίο initial range πρέπει να βάλουμε [1;15] για το συγκεκριμένο πρόβλημα



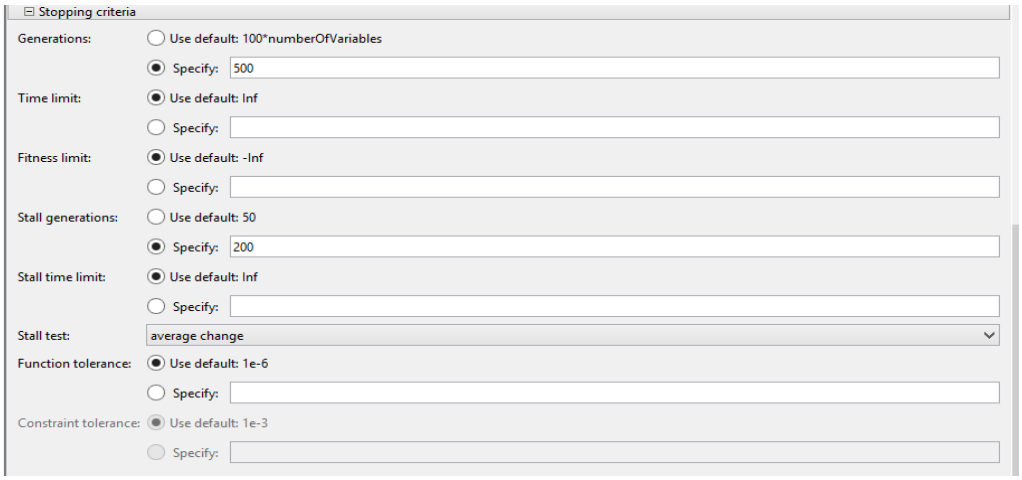
6. Fitness Scaling: Επιλέγουμε με ποιο τρόπο θέλουμε να γίνεται η διαβάθμιση. Σε αυτό το πρόβλημα θέλουμε αναλογικά.
7. Selection: σε αυτό το παράρτημα επιλέγουμε τον τρόπο επιλογής των γονέων από το σύνολο του πληθυσμού. Στο πρόβλημα αυτό επιλέγουμε stochastic uniform (τεχνική ρουλέτας).



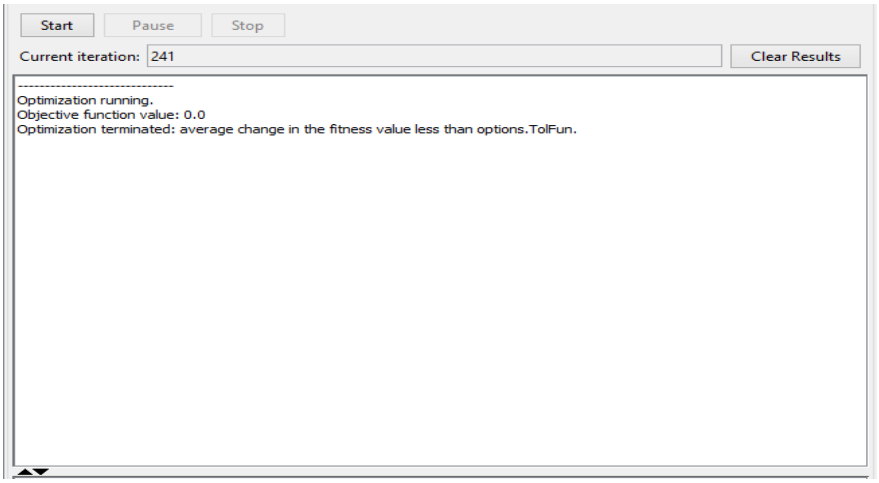
8. Reproduction: Επιλέγουμε τα ποσοστά των χρωμοσωμάτων που θα περάσουν στην επόμενη γενιά χωρίς να μεταλλαχθούν(ελιτισμός). Για το πρόβλημα η τιμές των επιλογών μπορούν να είναι default.
9. Mutation : Εδώ Επιλέγουμε στο crossover function την επιλογή custom και στο πεδίο που υπάρχει γράφουμε το όνομα της συνάρτησης μεταλλάξεων. Εμείς θα βάλουμε @cards_mutate_permutation
10. Crossover : Επιλέγουμε Custom και βάζουμε @cards_crossover_permutation



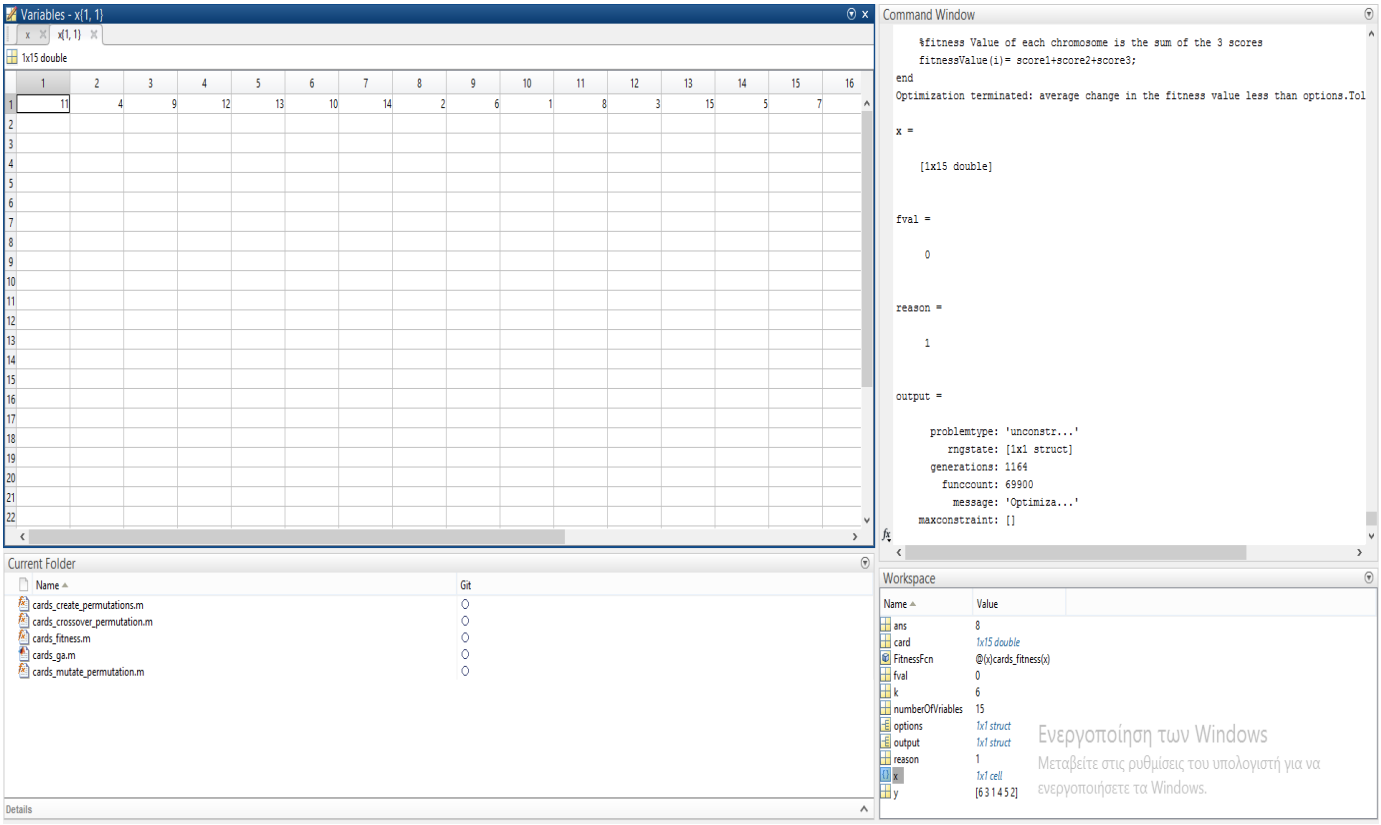
11. Στα πεδία Migration , Constraint parameters , Hybrid function αφήνουμε τις default τιμές τους.
12. Στο παράρτημα Stopping criteria μπορούμε να καθορίζουμε τα κριτήρια ώστε ο αλγόριθμος να σταματάει. Μπορούμε να τα αφήσουμε όλα default ή να αλλάξουμε ότι θέλουμε όπως θέλουμε εμείς θα ορίσουμε :
    - Generations : 500 (να σταματάει μετά από 500 γενιές)
    - Stall Generations : 200 (αν ο αλγόριθμος μετά από 200 γενιές δεν βελτιστοποιείται τότε να σταματήσει)



13. Plot functions : μπορούμε να επιλέξουμε κάποιο γράφημα για να αντλήσουμε πληροφορίες γραφικά για τον αλγοριθμό μας ή να τοποθετήσουμε μία δική μας συνάρτηση αν έχουμε φτιάξει. Ας επιλέξουμε το plot range.
14. Όταν ολοκληρώσουμε τις ρυθμίσεις πατάμε Start που βρίσκεται στην αριστερή στήλη και ο αλγόριθμος εκτελείται. Αφού ολοκληρωθεί επιστρέφει στο άσπρο παραθυράκι την καλύτερη functionValue



Αν θέλουμε να δούμε το αποτέλεσμα μετά το τέλος του αλγορίθμου πηγαίνουμε στο παράρτημα workspace του Matlab κάνουμε διπλό κλικ στο x και μετά ξανά διπλό κλικ στο παράρτημα variables :



Επίσης έχουμε το παρακάτω διάγραμμα μετά το πέρας του αλγορίθμου κατόπιν επιλογής (βήμα 13) :