Technical University of Crete

# Τεχνητή Νοημοσύνη

## Αναφορά 1ης Εργαστηριακής άσκησης(Μέρος Β)

**Φοιτητές**

Στέφανος Καλογεράκης | 2015030064

**Διδάσκων**

Χαλκιαδάκης Γ.

# Εισαγωγή

Ως δεύτερος σκέλος της πρώτης εργαστηριακής άσκησης του μαθήματος, μας ζητήθηκε η υλοποίηση ενός γενετικού αλγορίθμου, ο οποίος επιλύει το πρόβλημα του χρονοπρογραμματισμού του προσωπικού ενός οργανισμού. Ο αλγόριθμος εκκινεί από έναν τυχαία σχεδιασμένο πληθυσμό, και καταλήγει σε κάποιον που ικανοποιεί όσο καλύτερα γίνεται τους περιορισμούς του προβλήματος, μέσα από συνεχής εξέλιξη όπως πραγματοποιείται σε έναν γενετικό αλγόριθμο.

# Υλοποίηση

Η υλοποίηση πέρασε από αρκετά στάδια και άλλαξε αρχικά προκειμένου να φτάσει στην τελική μορφή μετά από συνεχή πειραματισμό και επικοινωνία με τον βοηθό του εργαστηρίου. Στις επόμενες ενότητες γίνεται η προσπάθεια να παρουσιαστούν όσο πιο διεξοδικά γίνεται όλα αυτά τα στάδια εξέλιξης. Αξίζει επίσης να επισημανθεί ότι ο κώδικας περιέχει αναλυτικά σχόλια σε όλα τα στάδια, και εξηγούν αναλυτικά όλη την υλοποίηση.

## Δημιουργία αρχικού πληθυσμού

Η δημιουργία του πληθυσμού είναι το πρώτο βήμα για την υλοποίηση του γενετικού αλγορίθμου. Στο πρόβλημα μας δημιουργούμε αρχικό πληθυσμό ενώ το κάθε χρωμόσωμα που δημιουργούμε ως κομμάτι του πληθυσμού έχει διαστάσεις 14 * 30(14 είναι οι μέρες και 30 είναι οι υπάλληλοι). Οι διαθέσιμες τιμές κάθε κελιού είναι 0-3 που αναφέρονται στην εκάστοτε βάρδια του κάθε εργαζόμενου όπως έχουν οριστεί από την εκφώνηση.

Η αρχική προσέγγιση, έγινε με την χρήση με εντελώς τυχαία ανάθεση για όλες τις βάρδιες με χρήση της συνάρτησης Random, για την ανάθεση ψευδοτυχαίων ομοιόμορφων ακολουθιών από αριθμούς. Σύντομα αυτή η προσέγγιση αποδείχθηκε προβληματική,  αφού παρουσίασε πρόβλημα στο επόμενο αμέσως στάδιο της αξιολόγησης με τα χρωμοσώματα να μην είναι συνεπή, δηλαδή να μπορούν να ικανοποιούν τα hard constraints του πίνακα κάλυψης,  ακόμα και για αρκετά μεγάλο πληθυσμό. Σε επόμενη ενότητα γίνεται περαιτέρω ανάλυση της συνέπειας και το κομμάτι της αξιολόγησης.

Προκειμένου να επιλυθεί το πρόβλημα, οφείλαμε με κάποιο τρόπο να είμαστε σίγουροι οτι θα δημιουργηθούν ορισμένα συνεπή χρωμοσώματα, πάντα όμως με τυχαίο τρόπο. Αρχικά, χρησιμοποιώντας τον πίνακα καλυψης, το κάθε χρωμόσωμα γεμίζει με τις όλες τις απαραίτητες τιμές σειριακά.(initMatrix συνάρτηση). Η τυχαιότητα εισάγεται στην συνέχεια, όπου σε χρήση με την συνάρτηση Random πραγματοποιούνται τυχαία shuffle στον πίνακα με τέτοιο τρόπο ώστε να μην επηρεάζονται οι περιορισμοί. Τέλος, για να αυξήσουμε τον δείκτη τυχαιότητας βεβαιωνόμαστε ότι δεν είναι εφικτές όλες οι λύσεις και ανά ένα ποσοστό κάνουμε ανάθεση τιμής που επηρεάζει τον πίνακα περιορισμών. Με την τρέχουσα δομή του κώδικα για έναν πληθυσμό 5000 χρωμοσωμάτων, γύρω στα 2200-2300 περνάνε από τον έλεγχο συνέπειας.

## Αξιολόγηση-Συνάρτηση Καταλληλότητας

Σε αυτό το στάδιο πραγματοποιείται αξιολόγηση του κάθε χρωμοσώματος, οποία διέρχεται από δύο στάδια, την συνάρτηση συνέπειας και καταλληλότητας. Το θέμα της συνέπειας επισημάνθηκε και στην προηγούμενη ενότητα, είναι ο πιο αυστηρός έλεγχος όπου μόνο τα συνεπή χρωμοσώματα που περνάνε τον έλεγχο του πίνακα κάλυψης μπορούν να χρησιμοποιηθούν στα επόμενα βήματα.

Η συνάρτηση καταλληλότητας, πραγματοποιείται με την ανάθεση μιας βαθμολογίας σε κάθε χρωμόσωμα, απο τον πίνακα περιορισμών που έχει δοθεί και πάλι από την εκφώνηση.

Τόσο οι συνάρτηση συνέπειας, όσο και η συνάρτηση καταλληλότητας δεν παρουσιάζουν κάποια δυσκολία στην κατανόηση, καθώς αποτελούνται από απλούς ελέγχους εγκυρότητας και αναθέσεις τιμών. Τόσο οι αναθέσεις του κάθε χρωμοσώματος όσο και το σκορ γίνεται σε custom object(Statistics). Τέλος, το τελικό αποτέλεσμα εκτέλεσης αυτών των δύο συναρτήσεων (γίνεται μέσω της constraintChecker συνάρτησης) επιστρέφει μια λίστα με όλες τις συνεπείς λύσεις με αύξουσα συνάρτηση σκορ. Να επισημάνουμε σε αυτό το σημείο ότι οι τιμές που προσθέτουμε στο σκορ κάθε χρωμοσώματος είναι αρνητικό βάρος οπότε το καλύτερο χρωμόσωμα είναι αυτό με το μικρότερο σκορ.

## Επιλογή

Επόμενο στάδιο είναι αυτό της επιλογής.  Υπάρχουν αρκετές υλοποιήσεις και τρόποι να πορευτεί σε αυτό το στάδιο. Αρχικά η λύση προσεγγίστηκε αποκλειστικά με επιλογή με χρήση ρουλέτας(roulette wheel selection). Σε αυτό τον τρόπο, ένας τροχός χωρίζεται σε κομμάτια ανάλογα με την βαθμολογία των χρωμοσωμάτων και άρα είναι φανερό ότι όσο μεγαλύτερο το σκορ τόσο μεγαλύτερο κομμάτι του δίσκου κατέχει. Επειδή όμως όπως προαναφέρθηκε, το σκορ είναι αρνητικό θέλουμε το μικρότερο σκορ να είναι αυτό που κατέχει το μεγαλύτερο κομμάτι. Για αυτό τον λόγο δουλεύουμε για το κλάσμα 1/score για να παρουμε το επιθυμητό αποτέλεσμα(συνάρτηση rouletteWheelSelectionF) με την συνάρτηση σε κάθε κλήση της να επιστρέφει τον γονιό χρωμόσωμα που έχει προκύψει.

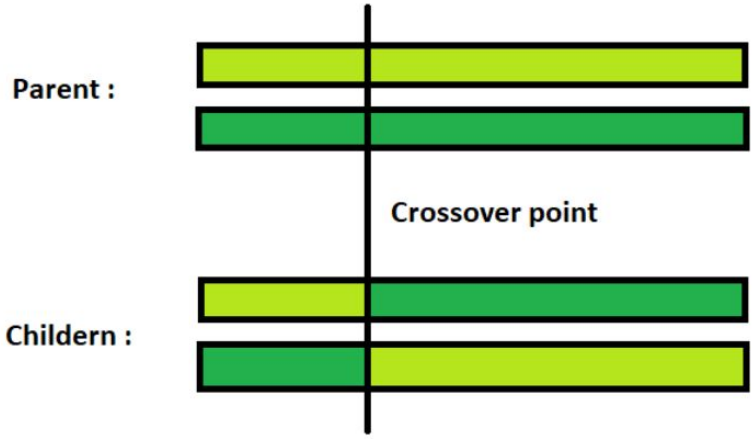
Βελτίωση της υλοποίησης για καλύτερα αποτελέσματα έγινε σε συνδυασμό της παραπάνω μεθόδου με ελιτισμό. Κατά τον ελιτισμό, επιλέγονται τα καλύτερα χρωμοσώματα του πληθυσμού για να περάσουν στην επόμενη γενιά. Πραγματοποιήθηκαν αρκετές δοκιμές αναφορικά με τον βέλτιστο συνδυασμό των δύο μεθόδων όπως για παράδειγμα μισός πληθυσμός με χρήση ελιτισμού και μισός με χρήση roulette κ.α.. Ο συνδυασμός με τα βέλτιστα αποτελέσματα ήταν απλά το πέρασμα ενός 10% των καλύτερων χρωμοσωμάτων ακριβώς όπως είναι στην επόμενη γενιά και roulette wheel για όλο τον πληθυσμό. Στην υλοποίηση μας το συγκεκριμένο κομμάτι ήταν αρκετά απλό καθώς υπήρχε λίστα με ταξινομημένα τα σκορ οπότε δεν χρειάστηκαν πολλές αλλαγές προκειμένου να στείλουμε το 10% των χρωμοσωμάτων ως έχειν.

## Διασταύρωση

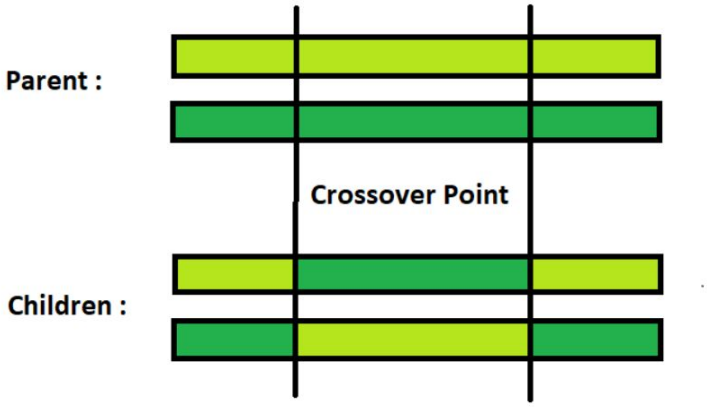
Στο στάδιο της διασταύρωσης (crossover) ζητήθηκε να παρέχουμε δύο διαφορετικές μεθόδους. Και οι δύο μέθοδοι είναι παρόμοιες και ακολουθούν την ίδια λογική υλοποίησης, με ονομασία singlePointCrossover και multiplePointCrossover.

Στην υλοποίηση single point crossover επιλέγουμε ένα τυχαίο σημείο από τον άξονα x (ημέρες εργασίας) μιας και οι περιορισμοί είναι με βάση την ημέρα και δεν θέλουμε να τους παραβιάσουμε, και παίρνουμε μέχρι αυτό το σημείο εγγραφές του

ενός γονέα και μετά μέχρι το τέλος από τον άλλο γονέα όπως φαίνεται στην παρακάτω εικόνα




Η υλοποίηση multiple point crossover, όπως προείπαμε ακολουθούν την ίδια λογική, με την διαφορά όμως ότι σε αυτή την περίπτωση ο διαχωρισμός γίνεται σε περισσότερα σημεία όπως φαίνεται στην εικόνα. Στην περίπτωση μας έχουν χρησιμοποιηθεί 4 σημεία διαχωρισμού στον άξονα x

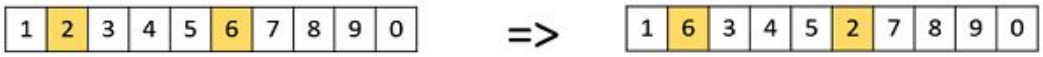



Όπως παρατηρείται και στις δύο περιπτώσεις παράγονται δύο παιδιά από τους γονείς και εμείς απλώς επιλέγουμε τυχαία πιο απο τα δύο θα επιστρέψουμε

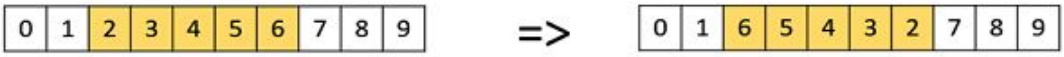
## Μετάλλαξη

Ο τελεστής της μετάλλαξης χρησιμοποιείται για να εμποδίσει την ομογενοποίηση του πληθυσμού των χρωμοσωμάτων που ίσως προκληθεί σε επόμενες γενιές απο την διασταύρωση. Και σε αυτή την περίπτωση ζητήθηκε να μελετηθούν 2 μέθοδοι αλλά υλοποιήθηκαν 4 όπως θα εξηγηθεί παρακάτω.

Οι δυο μέθοδοι που τελικά είναι αυτοί που εξυπηρετούν την υλοποίηση μας είναι οι μέθοδοι με ονομασία twoPointSwappingSameRow και InverseDays. Η πρώτη μέθοδος, απλά επιλέγει τυχαία μια σειρά του άξονα x και τυχαία δύο σημεία από την ίδια γραμμή και τους αλλάζει θέση όπως φαίνεται παρακάτω




Η μέθοδος InverseDays, επιλέγει και αυτή τυχαία μια σειρά του άξονα χ και τυχαία δύο σημεία από την ίδια γραμμή. Αυτή την φορα αντί να τους αλλάζει θέση, αντιστρέφει για το διάστημα αυτό τις θέσεις των τιμών όπως φαίνεται παρακάτω




Στον κώδικα για λόγους πληρότητας παραθέτουμε δύο μεθόδους ακόμα οι οποίες δεν χρησιμοποιούνται με την ένδειξη LEGACY. Χωρίς να αναλυθούν υπερβολικά, η μια μέθοδος πραγματοποιούσε αντιστροφή δύο τυχαίων ολόκληρων σειρών μεταξύ τους, και η άλλη ήταν η αντιστροφή δύο σημείων σε εντελώς τυχαίες όμως συντεταγμένες του πίνακα. Το πρόβλημα και των δυο μεθόδων και δεν επιλέχθηκαν είναι ότι στις περισσότερες περιπτώσεις θα άλλαζαν το χρωμόσωμα με τέτοιο τρόπο που δεν θα περνούσε εκ νέου από έλεγχο συνέπειας. Θα οφείλαμε λοιπόν συνεχώς να εισάγουμε καινούργια νέα χρωμοσώματα όταν είχαμε μετάλλαξη και θα απαιτούσε παραπάνω λογική στον κώδικα μας, την οποία με την τωρινή υλοποίηση αποφεύγουμε

## Επανάληψη- Κριτήρια τερματισμού

Στην περίπτωση μας, επιλέχθηκε να πραγματοποιείται τερματισμός μετά από ένα πεπερασμένο αριθμό επαναλήψεων. Πραγματοποιήθηκαν αρκετές δοκιμές αναφορικά με τον βέλτιστο αριθμό επαναλήψεων, αλλά τελικά παρόλο που σε μερικά σημεία δείχνει να μην υπάρχει εξέλιξη, όσο προχωράει ο αλγόριθμος δίνει και καλύτερα αποτελέσματα. Στο κομμάτι αξιολόγηση επισημαίνονται περισσότερες λεπτομέρειες αναφορικά με τις τιμές που τελικά επιλέχθηκαν για αξιολόγηση.

### Τοπική Αναζήτηση(Bonus)

Σαν μπόνους κομμάτι της εργασίας, μετά το στάδιο της μετάλλαξης. Η υλοποίηση που ακολουθήθηκε ήταν μια μορφή hill climbing για την εύρεση τοπικού ελαχίστου. Μετά από υπόδειξη για τις δυνατές επιλογές αναζήτησης πραγματοποιήθηκε σύγκριση όλων των δυνατών επιλογών μετά το στάδιο της μετάλλαξης και την εύρεση του ελαχίστου συγκεκριμένα των δύο γονιών πριν το στάδιο της διασταύρωσης(τωρινή κατάσταση), όπως και των δύο παιδιών που προκύπτουν μετά από την εκτέλεση των δύο μεθόδων μετάλλαξης. Το αποτέλεσμα της εκτέλεσης αναφέρεται στην ενότητα της αξιολόγησης.

## Αξιολόγηση

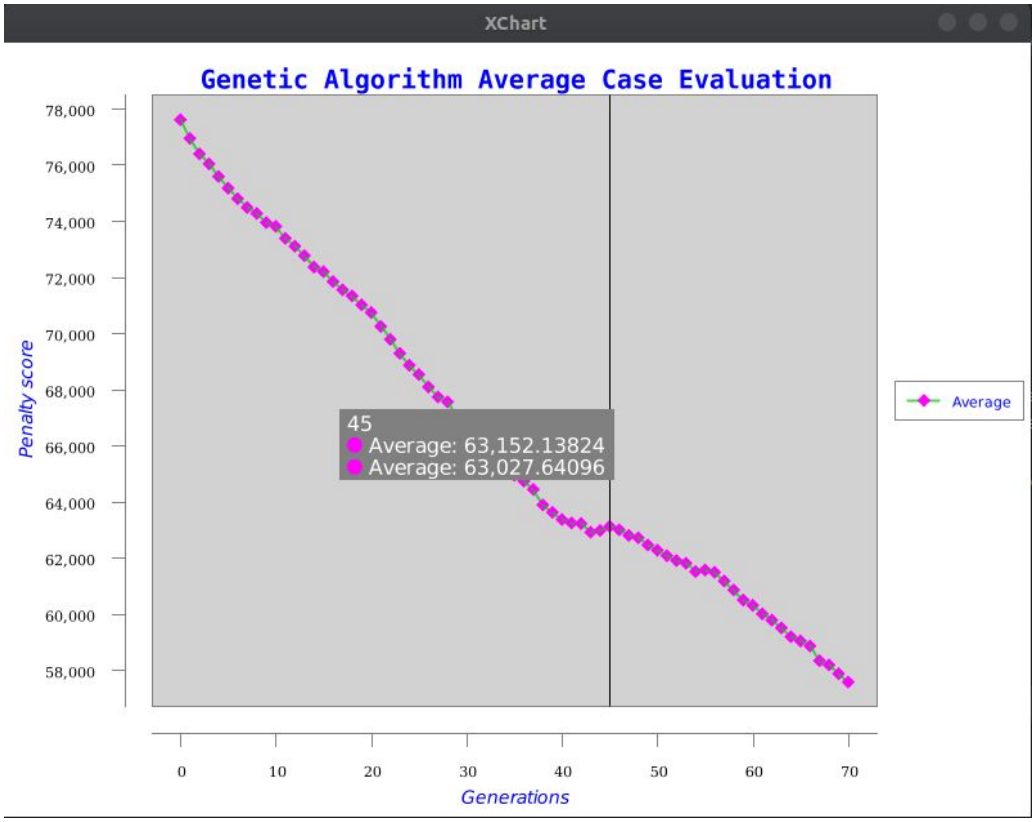
Για την αξιολόγηση όλων μας των αλγορίθμων υπήρχαν πάρα πολλοί συνδυασμοί που δοκιμάστηκαν μέχρι την τελική υλοποίηση. Πληθώρα paper σχετικά με τον γενετικό αλγόριθμο, συντέλεσαν στην τελική επιλογή πιθανοτήτων για τα στάδια των αλγορίθμων αφού επισημανθηκαν οι εξής πληροφορίες:

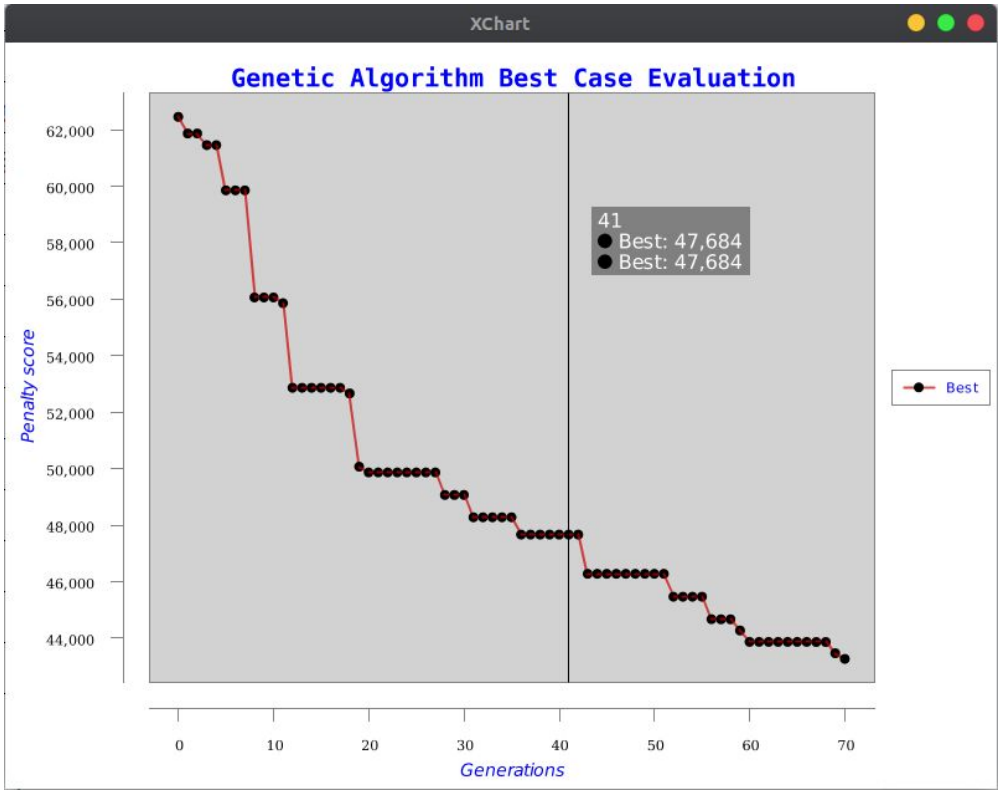
- Μεταλλάξεις δεν πρέπει να συμβαίνουν συχνά σε έναν γενετικό αλγόριθμο, καθώς θα αυτό θα οδηγήσει σε τυχαία αναζήτηση( Best Rates: 0.5-1%)
- Διασταυρώσεις πραγματοποιούνται με την ελπίδα ότι τα νέα χρωμοσώματα θα είναι καλύτερα και θα διατηρηθούν τα καλύτερα παλιά χρωμοσώματα(Best Rates 80-95%)

Όπως προαναφέρθηκε, το τιμή για **itermax** που να σταματήσει η εξέλιξη δεν υπήρχε. Στα πειράματα που απεικονίζονται παρακάτω επιλέχθηκε η τιμή 50(παραδοτέο) και 70 για την κάθε εκτέλεση, προκειμένου να φανεί όσο καλύτερα γίνεται η εξέλιξη παρά το γεγονός ότι ήταν χρονοβόρα. Η τιμή του **population** που επιλέχθηκε ήταν 5000 και 7000(παραδοτέο), με περίπου 2200-2300 και 3100-3200 αντίστοιχα feasible χρωμοσώματα ανά γενιά. Στον πίνακα αναγράφονται επίσης και οι διαφορετικές πιθανότητες επιλογής τοπικής αναζήτησης

| Συνδυασμοί | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | singlePointCross | | | | multiplePointCrossover | | | |
| | *twoPointSwap* | | *InverseDays* | | *twoPointSwap* | | *InverseDays* | |
| *Πληθυσμός* | 5000 | 7000 | 5000 | 7000 | 5000 | 7000 | 5000 | 7000 |
| *Επαναλήψεις* | 50 | 70 | 50 | 70 | 50 | 70 | 50 | 70 |
| *psel* | 99% | 90% | 99% | 90% | 99% | 90% | 99% | 90% |
| *pcross* | 80% | 95% | 80% | 95% | 80% | 95% | 80% | 95% |
| *pmut* | 0.5% | 1% | 0.5% | 1% | 0.5% | 1% | 0.5% | 1% |
| *ploc* | 90% | 70% | 90% | 70% | 90% | 70% | 90% | 70% |
| ***Αρχικό Best*** | **62673.0** | **62476.0** | **64082.0** | **60072.0** | **65082.0** | **63671.0** | **64083.0** | **61580.0** |
| ***Αρχικό Average*** | **77735.5** | **77610.5** | **77740.1** | **77755.9** | **77686.3** | **77555.8** | **77642.7** | **77732.7** |
| ***Τελικό Best*** | **46982.0** | **41480.0** | **46677.0** | **41182.0** | **45482.0** | **48968.0** | **43086.0** | **44477.0** |
| ***Τελικό Average*** | **51823.8** | **58146.2** | **56042.0** | **56202.2** | **63329.2** | **71152.2** | **57572.4** | **64474.9** |

Ενδεικτικά παρατίθενται γραφήματα για το best score και average score ανά γενιά για **multiplePointCross**, **twoPointSwap** για **70** επαναλήψεις και **7000** πληθυσμό. Ανανεωμένα γραφήματα αυτά προκύπτουν κάθε φορά μετά από την εκτέλεση του προγράμματος, όπως και αναλυτικά όλα τα δεδομένα.


XChart
Genetic Algorithm Average Case Evaluation
78,000
76,000
74,000
72,000
70,000
68,000
66,000
64,000
62,000
60,000
58,000
Penalty score
45
Average: 63,152.13824
Average: 63,027.64096
Average
0
10
20
30
40
50
60
70
Generations


XChart
Genetic Algorithm Best Case Evaluation
62,000
60,000
58,000
56,000
54,000
52,000
50,000
48,000
46,000
44,000
Penalty score
41
Best: 47,684
Best: 47,684
Best
0
10
20
30
40
50
60
70
Generations

# Βιβλιογραφία

https://www.youtube.com/watch?v=9JzFcGdpT8E&list=PLea0WJq13cnARQILcbHUPlNYLy1lOSmjH&index=15

https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

https://www.hindawi.com/journals/mpe/2015/906305/

https://github.com/haifengl/smile/blob/master/core/src/main/java/smile/gap/GeneticAlgorithm.java

https://arxiv.org/abs/1303.5909