ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ
ΠΛΗ417

LAB41740898

1η Εργασία Προγραμματισμού

Φλέγγας Γεώργιος 2014030161

Θεοδωράκη Εμμανουέλα 2014030238

20/04/19

ΗΜΜΥ

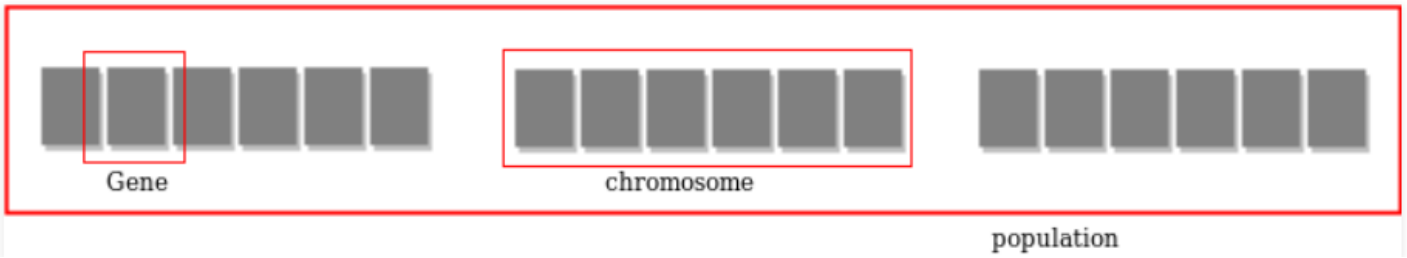
Β. Υλοποίηση ενός Γενετικού Αλγορίθμου για το Πρόβλημα του Χρονοπρογραμματισμού Προσωπικού

Εισαγωγή

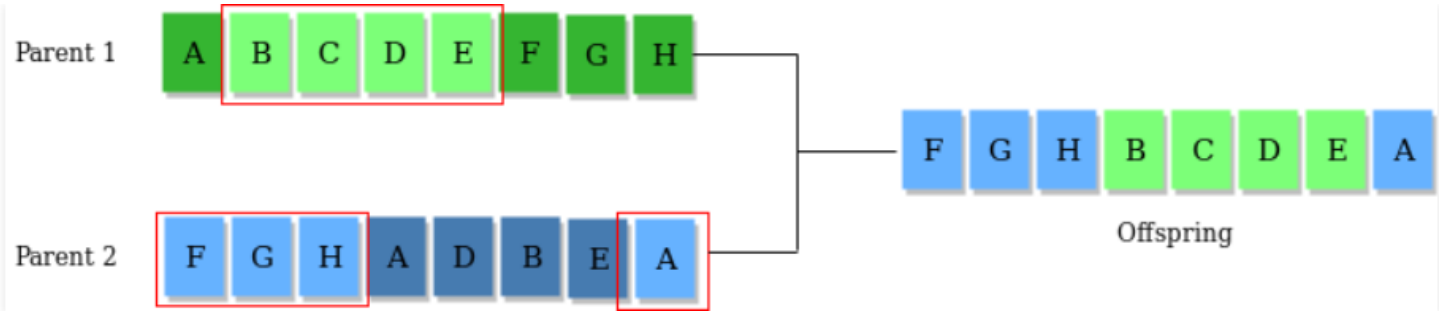
Για το 2ο κομμάτι της εργασίας, ζητήθηκε να υλοποιήσουμε έναν αλγόριθμο βασισμένο στον Γενετικό Αλγόριθμο, ο οποίος θα επιλύει το πρόβλημα του χρονοπρογραμματισμού του προσωπικού κάποιας εταιρίας-οργανισμού. Στόχος του αλγορίθμου μας, είναι ξεκινώντας από κάποιο αρχικό πληθυσμό προγραμμάτων τα οποία τηρούν τους απόλυτους περιορισμούς, να καταλήξει σε κάποιο το οποίο θα ικανοποιεί όσο το δυνατόν καλύτερα γίνεται τους ελαστικούς περιορισμούς, μέσα από μια διαδικασία συνεχούς εξέλιξης που ακολουθεί τα πρότυπα της εξέλιξης των έμβιων οργανισμών και της φυσικής επιλογής.

Συσχετισμός Γενετικού αλγορίθμου και της υλοποίησης μας

Σύμφωνα με τον γενετικό αλγόριθμο, τα χρωμοσώματα του κάθε πληθυσμού που διαθέτουν τα περισσότερα κατάλληλα γονίδια, είναι πιο πιθανό να επιλεχθούν για ανα-παραγωγή από ότι είναι αυτά τα οποία δεν τα διαθέτουν. Ένας πληθυσμός αποτελείται από χρωμοσώματα,τα οποία στην σειρά τους αποτελούνται από γονίδια, που περιγράφουν τα χαρακτηριστικά του χρωμοσώματος .



Ξεκινώντας από έναν αρχικό πληθυσμό, που λαμβάνει τυχαία τιμές από το χώρο αναζήτησης, περνάνε τα χρωμοσώματα από την συνάρτηση καταλληλότητας και βαθμολογούνται τα χαρακτηριστικά των γονιδίων, δίνοντας της μια αντιπροσωπευτική τιμή. Στην συνέχεια με βάση την βαθμολογία αυτή, επιλέγονται τα καλύτερα για διασταύρωση και παραγωγή ενός απόγονου, συνθέτοντας τα γονίδια τους και ταυτόχρονα χάρης την μετάλλαξη φροντίζει να αποφευχθεί η δημιουργία ομοιογένειας στον πληθυσμό.



Ένας γενικός ψευδοκώδικας που απεικονίζει τα παραπάνω είναι ο εξής:

```
Initialize population p
evaluate p
while not end_condition{
      select parents
      Crossover
      Mutation
      evaluate new population
}
```

Στην υλοποίηση μας ο ρόλος του χρωμοσώματος ανατίθεται στο χρονοπρόγραμμα, του οποίου τα γονίδια είναι το πρόγραμμα εργασίας του κάθε εργαζομένου. Το πρόγραμμα του κάθε εργαζομένου χαρακτηρίζεται από την βάρδια στην οποία δουλεύει για κάθε μια από τις μέρες του χρονικού ορίζοντα. Η συνάρτηση αξιολόγησης, θα ελέγχει κατά κύριο λόγο τους ελαφρούς περιορισμούς δίνοντας ένα score σε κάθε πιθανό Schedule, ενώ οι αυστηροί περιορισμοί θα πρέπει να ελέγχονται κατά την δημιουργία ενός νέου προγράμματος. Κατά την διασταύρωση, θα επιλέγονται 2 Schedules και θα δημιουργείται ένα καινούργιο, συνδυάζοντας προγράμματα εργαζομένων που θα προέρχονται και από τους 2 γονείς, ενώ για την μετάλλαξη θα αλλάζουν μέρη του προγράμματος τυχαίων εργαζομένων.

Υλοποίηση

Αρχικοποίηση:
Κατά την αρχικοποίηση του πληθυσμού, αρχικά θέτουμε όλες τις βάρδιες ως '0', δηλαδή άδεια και στην συνέχεια ξεκινάμε να δίνουμε τιμές στον πίνακα των βαρδιών με βάση τις ημέρες. Τυχαία επιλέγουμε έναν από τους 30 εργαζόμενους και εφόσον δεν του έχει ανατεθεί κάποια βάρδια, του αναθέτουμε κατά σειρά προτεραιότητας: πρωινή '1', απογευματινή '2', βραδινή '3' . Όταν συμπληρωθεί ο αριθμός των απαραίτητων ατόμων για την συγκεκριμένη μέρα στην πρωινή βάρδια, προχωράμε στην απογευματινή και με τον ίδιο τρόπο συνεχίζουμε με την βραδινή. Όταν όλες οι βάρδιες έχουν γεμίσει, ξεκινάει η διαδικασία από την αρχή για την επόμενη μέρα, μέχρι να συμπληρωθεί ο αριθμός των ημερών που προβλέπει ο χρονικός ορίζοντας.

Έλεγχος συνέπειας- απόλυτων περιορισμών:
Χάρης τον τρόπο με τον οποίο δημιουργήσαμε τον αρχικό μας πληθυσμό, εξασφαλίζουμε την εκπλήρωση των απόλυτων περιορισμών. Όμως δημιουργήσαμε μια ακόμα συνάρτηση, την public int evaluate() , που καλούμε όποτε γίνεται διασταύρωση και μετάλλαξη, ώστε τα νέα χρωμοσώματα να ικανοποιούν με την σειρά τους, αυτούς τους περιορισμούς.

Βαθμολόγηση- ελαστικοί περιορισμοί:
Για την υλοποίηση της συνάρτησης καταλληλότητας, βασιστήκαμε στον πίνακα των soft constraints που μας δόθηκε Δημιουργήσαμε την συνάρτηση public double ScoreCalculation(Employee Empl), η οποία ελέγχει το πρόγραμμα εργασίας ενός εργαζομένων και προσθέτει στην βαθμολογία-fitness του τις απαραίτητες μονάδες σε περίπτωση που δεν τηρείται κάποιος από τους περιορισμούς του. Εν τέλη επιστρέφει το σύνολο των μονάδων αυτών και τις θέτει ως το fitness του προγράμματος αυτού. Η συνάρτηση αυτή καλείται επανειλημμένα μέσα από την public void SchedScore(), ώστε να υπολογιστεί το συνολικό fiteness του χρονοπρογραμματος-χρωμοσώματος.

Μέθοδος επιλογής:
Έχουμε 2 μεθόδους επιλογής τους οποίους αξιοποιούμε στον αλγόριθμο μας.

- Ο πρώτος είναι να ταξινομούμε ως προς το fitness όλα τα μέλη του πληθυσμού και να λαμβάνουμε τον καλύτερο μέσο της public Schedule getFittest(int offset), στην οποία θέτουμε το offset ως 0 για να λάβουμε το καλύτερο χρωμόσωμα.
- Ο δεύτερος είναι με την υλοποίησης της public Schedule rouletteWheelSelection(Population pop), που εκτελεί επιλογή με χρήση ρουλέτας όπως περιγράφτηκε στην εκφώνηση που μας δόθηκε. Επιλέγουμε ένα τυχαίο σημείο του αθροίσματος των fitness του πληθυσμού και στην συνέχεια επαναληπτικά προσθέτουμε τα fitness μέχρι να ξεπεράσουμε το σημείο αυτό. Μόλις συμβεί αυτό, επιστρέφουμε τον χρωμόσωμα το οποίο προσθέσαμε τελευταίο.

Μέθοδοι διασταύρωσης:
Για την διασταύρωση μας ζητήθηκε να δημιουργήσουμε 2 διαφορετικές μεθόδους:

- Η πρώτη public Population crossoverBest(Population population), βασίζεται στην επιλογή των 2 καλύτερων χρωμοσωμάτων μέσω της getFittest και στην συνέχεια να χρησιμοποιεί για το 1ο μισό τα γονίδια του καλύτερου, ενό για το 2ο τα γονίδια του 2ου καλύτερου.
- Η δεύτερη public Population crossoverPopulation(Population population), επιλέγει αρχικά το καλύτερο χρωμόσωμα ως γονέα και στην συνέχεια έχοντας υπόψιν το elitism και το crossover ratio, αξιοποιεί το rouletteWheelSelection, για την επιλογή του 2ου γονέα και εν τέλη επιλέγει τυχαία μισά-μισά από τους γονείς τα γονίδια και δημιουργεί το καινούργιο χρωμόσωμα, εκτελεί έλεγχο της συνέπειας του και αν είναι αποδεκτό το προσθέτει στον νέο πληθυσμό. Σε αντίθετη περίπτωση προσθέτει τον parent1.

Μέθοδοι μετάλλαξης:
Για την μετάλλαξη μας ζητήθηκε να δημιουργήσουμε 2 διαφορετικές μεθόδους. Και οι 2 είναι βασισμένες τόσο στο Elitism, όσο και στο mutation rate:

- Η πρώτη public Population mutatePopulation(Population pop), όταν ικανοποιηθεί το mutation rate, δημιουργείται ένα τυχαίο πρόγραμμα υπαλλήλου και αντικαθιστά το υπάρχον πρόγραμμα του.
- Η δεύτερη public Population flipMutate(Population pop), όταν ικανοποιηθεί το mutation rate, ανταλλάσσει τις βάρδιες 2 τυχαίων υπαλλήλων για μια συγκεκριμένη μέρα.

Κριτήρια τερματισμού:
Όσον αφορά το κριτήριο με τον οποίο θα τερματίζει την εξέλιξη ο αλγόριθμος, δοκιμάσαμε 2 διαφορετικές εκδοχές.

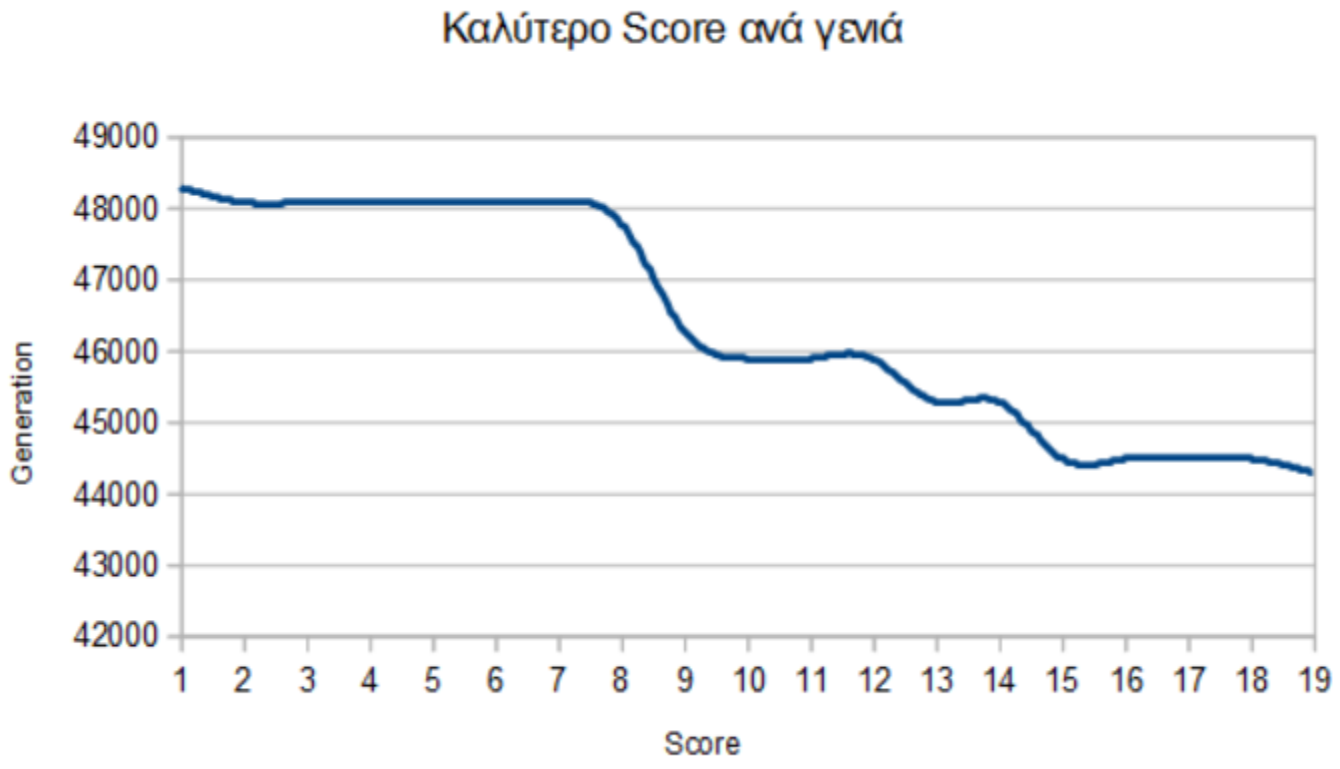
- Η πρώτη, η οποία θα ήταν και η πιο ελκυστική είναι να φτάσει στο σημείο όπου το score θα είναι πάρα πολύ μικρό(<100) και έτσι θα έχουμε ένα σχεδόν τέλειο χρονοπρόγραμμα. Αυτό όμως δεν συνέβη ποτέ και καταλήγαμε να φτάσουμε σε ένα σημείο που το πρόγραμμα σταματούσε να εξελίσσετε.
- Η δεύτερη, την οποία τελικά χρησιμοποιήσαμε για την άντληση των αποτελεσμάτων μας, βασίστηκε στο αριθμό μέγιστων επαναλήψεων. Όταν ο αλγόριθμος μας ξεπεράσει τον αριθμό αυτό, τότε σταματάει και μας επιστρέφει το καλύτερο πρόγραμμα που κατάφερε να δημιουργήσει.

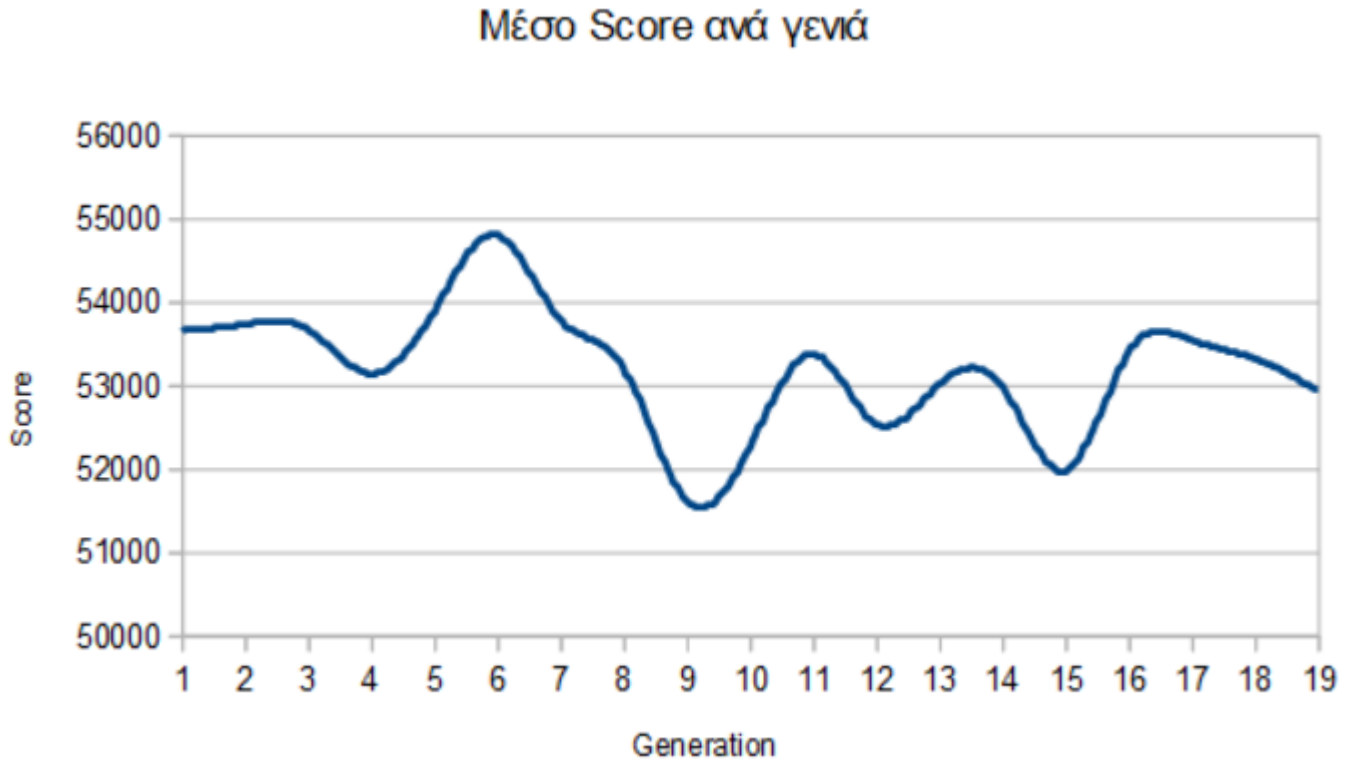
Αξιολόγηση

Αφού πειραματιστήκαμε με διάφορες τιμές στον αρχικό πληθυσμό (pop), με τον μέγιστο αριθμό επαναλήψεων (MaxIter), τις πιθανότητες επιλογής - διασταύρωσης - μετάλλαξης αποφασίσαμε να παρουσιάσουμε τα αποτελέσματα 2 χαρακτηριστικών περιπτώσεων, που φαίνονται παρακάτω :

	Combination							
	crossoverPopulation flipMutate		crossoverPopulation mutatePopulation		crossoverBest flipMutate		crossoverBest mutatePopulation	
Population	20	100	20	100	20	100	20	100
MutationRate	0.001	0.005	0.001	0.005	0.001	0.005	0.001	0.005
CrossoverRate	0.95	0.7	0.95	0.7	0.95	0.7	0.95	0.7
MaxIter	100	1000	100	1000	100	1000	100	1000
ElitismCount	2	5	2	5	2	5	2	5
Initial Score	45,686	49,874	46,674	45,664	47,285	48,680	49,868	49,872
Final Score	43,073	39,883	40,879	39,984	45,373	45,373	47,668	43,681
Final Average	52,799	53,989	53,539	53,650	54,367	54,367	56,727	68,837

Παρακάτω παρουσιάζονται ενδεικτικά οι μεταβολές των τιμών του καλύτερου score και του μέσου για τον συνδυασμό crossoverPopulation και mutatePopulation που αποτελεί και το καλύτερο αποτέλεσμα συνδυασμού από τους αλγορίθμους που υλοποιήσαμε:





Η μορφή του αντίστοιχου προγράμματος θα είναι :

Emp	M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	0	0	2	1	0	1	2	2	3	1	0	2	0	2
2	1	1	2	2	0	3	0	1	1	0	2	3	0	0
3	1	1	2	0	2	0	0	1	1	0	3	3	0	0
4	1	2	0	0	1	1	1	0	0	2	0	3	0	0
5	1	2	3	3	3	2	2	0	0	2	0	2	3	3
6	1	2	0	3	2	0	0	2	3	3	1	1	1	1
7	2	2	0	2	3	0	0	2	1	3	3	0	2	0
8	3	3	0	0	2	3	3	2	2	0	3	0	3	1
9	2	3	0	2	1	0	1	2	3	3	0	3	0	3
10	1	1	1	0	0	2	3	3	1	2	0	0	0	0
11	2	2	2	3	0	2	0	1	2	0	1	2	0	2
12	2	0	3	0	2	0	0	1	2	1	0	2	1	0
13	2	1	0	1	2	1	3	2	1	1	0	0	0	1
14	3	1	0	3	1	0	0	0	2	1	2	2	0	0
15	2	1	0	0	3	3	0	0	2	0	3	0	2	1
16	0	1	0	0	0	0	2	2	2	2	3	1	0	0
17	1	1	2	0	2	0	0	1	1	2	0	1	0	0
18	2	2	3	1	0	0	3	1	3	3	0	1	1	0
19	1	2	1	2	0	3	3	0	2	0	1	0	0	0
20	1	2	0	2	2	0	0	3	1	2	2	0	2	0
21	3	0	1	0	2	3	1	3	2	0	2	2	1	3
22	2	3	3	0	2	0	1	1	1	0	0	2	0	3
23	2	2	2	0	0	0	0	1	0	3	0	0	1	2
24	0	1	3	1	3	0	0	3	1	0	1	2	3	0
25	3	0	2	0	0	0	2	2	0	1	0	3	0	1
26	0	2	2	0	3	0	0	1	2	0	0	2	0	2
27	0	3	2	0	0	1	0	2	3	2	0	0	3	3
28	3	3	1	0	2	2	1	2	1	2	1	1	3	0
29	2	0	2	3	1	2	0	1	0	2	2	2	2	2
30	1	1	1	1	1	1	2	3	2	2	0	0	2	0