Πολυτεχνειο Κρητης
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

[ΠΛΗ 417] - Τεχνητή Νοημοσήνη

Εαρινό Εξάμηνο Ακ. Έτους 2019-2020

$1^{η}$ **Προγραμματιστική Εργασία**
Βάρος 38% βαθμού μαθήματος + **2%** πιθανό bonus

Παράδοση: 24 Απριλίου 2020, 23:55

Διδάσκων: *Γεώργιος Χαλκιαδάκης*

Βοηθοί: *Αντώνης Βογιατζής*, *Νίκος Τρίγκας*

Οδηγίες

Ηλεκτρονική παράδοση συμπιεσμένου αρχείου μέσω του ιστοχώρου του μαθήματος (συμπεριλαμβανομένου του κώδικα). Φροντίστε να είναι καλά οργανωμένο το παραδοτέο σας σε υποκαταλόγους με ευνόητα ονόματα. Τεκμηριώστε τον κώδικά σας επαρκώς. Συνοδέψτε την εργασία με μια το πολύ 5-σέλιδη (για το πρώτο μέρος της) και με μια το πολύ 8-σέλιδη (για το δεύτερο μέρος της) αναφορά, όπου εξηγείτε τις επιλογές σας και περιγράφετε τα αποτελέσματά σας. Θα βαθμολογηθείτε και για την ποιότητα των αναφορών σας - φροντίστε λοιπόν να είναι κατανοητές, να απαντάνε στα ερωτήματα των εκφωνήσεων, και να αντιστοιχούν σε/ περιγράφουν ορθά τον κώδικά σας. Η υλοποίηση μπορεί να γίνει σε οποιαδήποτε γενικής χρήσης γλώσσα προγραμματισμού εσείς επιλέξετε. *Μπορείτε ελεύθερα να χρησιμοποιήσετε «έτοιμο» κώδικα* για επιμέρους κομμάτια των προγραμμάτων σας, αρκεί να αναφέρετε επακριβώς τις πηγές σας*!

Οι ομάδες εργασίας είναι το πολύ 2 ατόμων.
* Κατά τα λοιπά, αντιγραφή συνεπάγεται άμεσο μηδενισμό *στο μάθημα*.

Μέρος Α: Πειραματισμός με μεθόδους αναζήτησης

Βάρος 15% Βαθμού Μαθήματος

1 Αλγόριθμοι BFS, DFS και A* [10%]

Στο πρώτο μέρος της προγραμματιστικής εργασίας καλείστε να υλοποιήσετε κάποιες μεθόδους αναζήτησης. Το πρόβλημα που πρέπει να λύσετε είναι το εξής: κινούμενοι σε έναν κόσμο (grid-world), ξεκινώντας από την αφετηρία πρέπει να φτάσετε στο σημείο τερματισμού ελαχιστοποιώντας το κόστος διαδρομής, όπου ο πράκτορας μπορεί να κινείται απο τετράγωνο σε τετράγωνο είτε κάθετα είτε οριζόντια (και μόνο), εφόσον η κίνηση του δεν παρεμποδίζεται από εμπόδια, τα οποία σημειώνονται στις εικόνες ως "τείχη".

Στο περιβάλλον η κίνηση μπορεί να γίνει είτε επί "ξηρού" εδάφους (τετράγωνα μπεζ χρώματος), είτε μέσα από δάση (πράσινα τετράγωνα). Η κίνηση μέσα από τα δάση περιορίζεται από την βλάστηση - και έχει αυξημένο κόστος. Συγκεκριμένα, το κόστος κίνησης επί ξηρού εδάφους = κόστος διάσχισης ενός μπεζ τετραγώνου ορίζεται να είναι ίσο με 1, ενώ το κόστος κίνησης μέσα από δάσος = κόστος διάσχισης πράσινου τετραγώνου ορίζεται να είναι ίσο είτε με (a) 2, είτε με (b) 10.

Με αυτά τα δεδομένα, ποια είναι η λύση που δίνουν οι BFS, DFS, και A* για κάθε μία από τις 2 αυτές περιπτώσεις (διαφορετικό κόστος για κάθε 5-αδα κόσμων); Παρουσιάστε τις λύσεις με κάποιον ευνόητο τρόπο της επιλογής σας στην αναφορά σας, μαζί με μια συζήτηση-επεξήγηση των τυχόν διαφορών τους. Φροντίστε να αναφέρετε επακριβώς το κόστος κίνησης που σχετίζεται με την κάθε λύση (το οποίο να προσφέρεται και ως έξοδος του κάθε αλγορίθμου, όπως και η λύση). Φροντίστε επίσης να αναφέρετε και να αιτιολογήσετε σημαντικές επιλογές σας - όπως τις ευρετικές συναρτήσεις που θα χρησιμοποιήσετε. Καλό είναι -αν και αυτό δεν απαιτείται αυστηρά- ο κώδικάς σας να είναι όσο πιο γενικός γίνεται.

Σας δίνεται βοηθητικός κώδικας ώστε να μην αναλωθείτε σε σημεία μη σχετικά με τον σκοπό της εργασίας, όπως είναι η δημιουργία του κόσμου. Συγκεκριμένα σας δίνονται η υλοποίηση του grid-world, ένα σύνολο από 5 διαφορετικούς κόσμους (διαφορετικής δυσκολίας), και υλοποίηση για το visualization του grid-world. Παρακάτω σας παρουσιάζουμε τις κλάσεις, και με ποιο τρόπο να τις χρησιμοποιήσετε.

Οι κόσμοι δίνονται με κόστος κελιού-δάσος 2, εσείς πρέπει να ελέγξετε αυτούς του κόσμους και για κόστος 2 και για κόστος 10.

2 Online Αλγοριθμος LRTA* [5%]

Υλοποιήστε και εκτελέστε μια παραλλαγή του αλγορίθμου online search LRTA* για να φτάσετε από την αφετηρία στο τέρμα σε κάθε έναν από τους πέντε κόσμους που σας δίνονται, θεωρώντας ότι το **κόστος διάσχισης για κελιά-δάσος είναι 10**, αλλά ότι **οι υπολογισμοί γίνονται online**: το «κόστος διάσχισης» ενός οποιουδήποτε κελιού γίνεται γνωστό **μόνο μετά την πρώτη διάσχισή του** (υπάρχει δηλαδή αβεβαιότητα για την συνάρτηση κόστους διάσχισης g(), η οποία αίρεται μετά την πρώτη επίσκεψη στο κελί - στον αλγόριθμό σας πρέπει να αποφασίσετε πώς να αντιμετωπίσετε την αρχική αυτή αβεβαιότητα), και μπορούμε να κάνουμε μόνο εντελώς τοπικές κινήσεις (εξετάζουμε μόνο τους άμεσους γείτονες).

Διαθέσιμες Κλάσεις

Grid.java Αποτελείται από ένα πίνακα διαστάσεων $N \times M$ τύπου *Cell*, έναν πίνακα που περιέχει τις θέσεις των κελιών-τοίχος μέσα στο grid, έναν πίνακα που περιέχει τις θέσεις των κελιών-δάσος, τις θέσεις της αφετηρίας και του τερματισμού, καθώς και το κόστος για κάθε κελί-δάσος. Για να δημιουργηθεί ένα object τύπου *Grid*:

- Να δημιουργήσετε ένα τυχαίο κόσμο διαστάσεων 13×9 με τον constructor *new Grid()*.
- Να δημιουργήσετε ένα τυχαίο κόσμο διαστάσεων $N \times M$ με τον constructor *new Grid(N,M)*.
- Να φορτώσετε ένα υπάρχον κόσμο από αρχείο με τον constructor *new Grid(path2file)*.

Επίσης είναι υλοποιημένες οι εξής μέθοδοι:

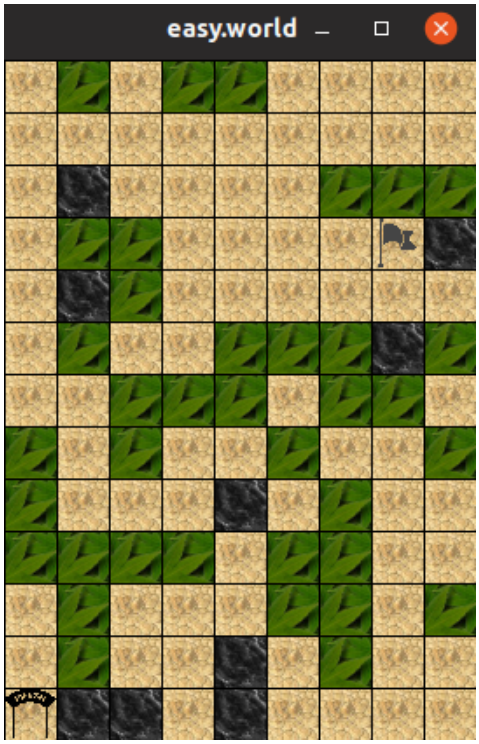
- getCell(int i, int j): επιστρέφει το κελί τύπου *Cell* του grid στη θέση (i,j).
- getStart(): επιστρέφει ένα πίνακα τύπου int με την θέση του κελιού *αφετηρία* στη μορφή (i,j).
- getTerminal(): επιστρέφει ένα πίνακα τύπου int την θέση του κελιού *τερματισμός* στη μορφή (i,j).
- getStartidx(): επιστρέφει την θέση του κελιού *αφετηρία* στη μορφή $i * M + j$.
- getTerminalidx(): επιστρέφει την θέση του κελιού *τερματισμός* στη μορφή $i * M + j$.
- getWalls() επιστρέφει έναν πίνακα τύπου int με τις θέσεις των κελιών-τοίχος στη μορφή $i * M + j$.
- getGrass(): επιστρέφει έναν πίνακα τύπου int με τις θέσεις των κελιών-δάσος στη μορφή $i * M + j$.

Cell.java Αποτελείται από τον τύπο του κελιού ('W' για κελί-τοίχο, 'G' για κελί-δάσος, 'L' για κελί-ξηρά), ένα boolean indicator για το αν είναι το κελί *αφετηρία*, ένα boolean indicator για το αν είναι το κελί *τερματισμός*, και το κόστος του κελιού–διαφέρει ανάλογα με τον τύπο του κελιού. Χρήσιμες μέθοδοι που είναι υλοποιημένες:

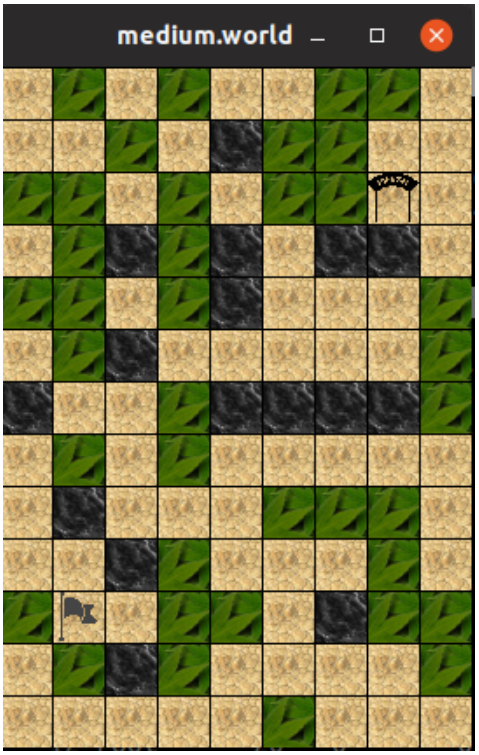
- isWall(): επιστρέφει true αν είναι κελί-τοίχος.
- isGrass(): επιστρέφει true αν είναι κελί-δάσος.
- isLand(): επιστρέφει true αν είναι κελί-ξηρά.
- isStart(): επιστρέφει true αν είναι το κελί *αφετηρία*.
- isTerminal(): επιστρέφει true αν είναι το κελί *τερματισμός*.
- getCost(): επιστρέφει έναν int με το κόστος του κελιού–σε περίπτωση που το κελί είναι τοίχος η τιμή έχει οριστεί ως Integer.MAX_VALUE.
- getCellType(): επιστρέφει τον char με τον χαρακτήρα του κελιού.

Drawing.java Δημιουργεί ένα frame όπου απεικονίζεται ο grid-world με ξηρά, τοίχους, δάση, αφετηρία και τερματισμό.

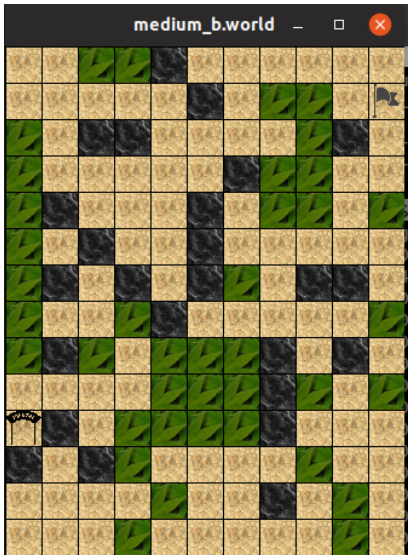
Figure 1: World examples



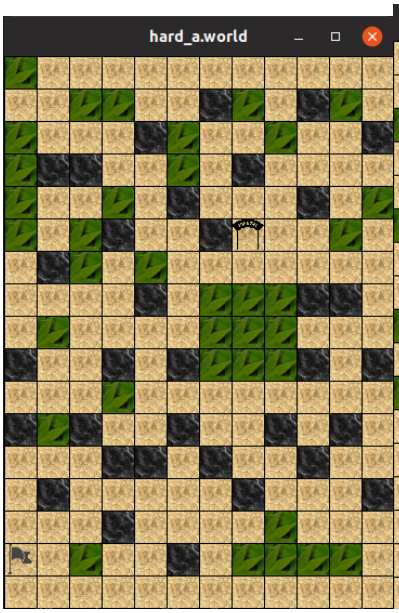
(a) world 1



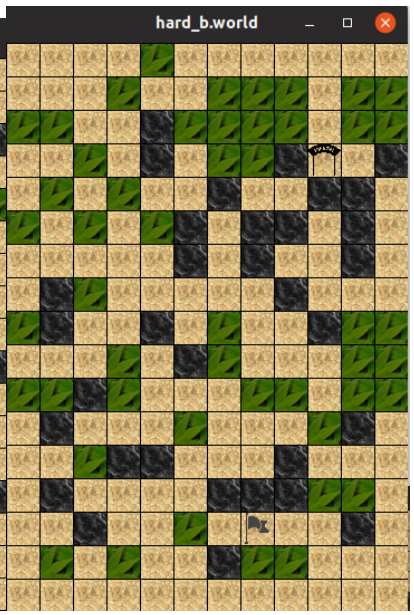
(b) World 2



(c) World 3



(d) World 4



(e) World 5

GridGenerator.java Περιέχει την *main*.

Διαθέσιμοι Κόσμοι

Στην Εικόνα 1 απεικονίζουμε τους διαθέσιμους κόσμους.

Προσοχή, η υλοποίησή σας πρέπει να δουλεύει
για **οποιονδήποτε** *κόσμο, όχι μόνο για αυτούς που σας δίνονται.*

Κάθε κόσμος δίνεται σε αρχείο "*filename.world*" στο φάκελο example_worlds. Κάθε αρχείο περιέχει

- μια γραμμή με τις διαστάσεις του grid.
- μια γραμμή με το πλήθος των κελιών-τοίχος, και τις θέσεις των κελιών-τοίχος στη μορφή $i * M + j$.
- μια γραμμή με το πλήθος των κελιών-δάσος, και τις θέσεις των κελιών-δάσος στη μορφή $i * M + j$.
- μια γραμμή με το κελί *αφετηρία* στη μορφή $i * M + j$.
- μια γραμμή με το κελί *τερματισμός* στη μορφή $i * M + j$.
- μια γραμμή με το κόστος για κάθε κελί-δάσος.

Παρακάτω δίνεται ένα παράδειγμα της δομής του αρχείου για τον default κοσμο:

```
dimensions:13x9
walls:14:0,9,13,22,31,40,49,50,51,54,74,83,99,108
grass:35:25,26,34,35,43,44,47,52,53,56,57,58,59,60,61,62,68,69,70,71,77,78,79,80,
86,87,88,89,97,101,102,106,110,111,115
start_idx:96
terminal_idx:42
grass_cost:2
```

Δεν είναι απαραίτητο τα στοιχεία να ακολουθούν την παραπάνω σειρά.

Κλήση GridGenerator

```
java GridGenerator
java GridGenerator -d 15 8
java GridGenerator -i example_worlds/world2.world
```

Η πρώτη κλήση δημιουργεί έναν random world διαστάσεων 13×9.
Η δεύτερη κλήση δημιουργεί έναν random world διαστάσεων $\times M$, όπου $N = 15$ και $M = 8$.
Η τρίτη κλήση φορτώνει το κόσμο *world2.world*.

Απεικόνιση Κόσμου (Grid-World Visualization)

Μέσα στην κλάση GridGenerator υπάρχουν δύο μέθοδοι για την απεικόνιση του grid.

```
VisualizeGrid(frame_name, N, M, walls, grass, start_idx, terminal_idx)
VisualizeGrid(frame_name, N, M, walls, grass, steps, start_idx, terminal_idx)
```

Με την πρώτη μπορείτε απεικονίσετε τον αρχικό κόσμο, ενώ με την δεύτερη δείχνετε και τα βήματα που οδηγούν από το σημείο αφετηρίας στο σημείο τερματισμού, όπως αυτή προκύπτει από την υλοποίηση των μεθόδων αναζήτησης. Τα walls, grass, steps είναι πίνακες από int, όπου κάθε στοιχείο αντιστοιχεί στην θέση του κελιού-τοίχος/κελιού-δάσος/κελί από το οποίο πέρασε στη μορφή $i * M + j$. Αντίστοιχα τα start_idx και terminal_idx είναι στη μορφή $i * M + j$.

Προτείνουμε να χρησιμοποιήσετε την GridGenerator ως την main class. Ωστόσο, εάν αλλάξετε την κυρίως κλάση σας, φροντίστε να υπάρχει δυνατότητα με μενού (όχι hard-coded) να φορτώνουμε τον επιθυμητό κόσμο. Επίσης, φροντίστε να προσθέσετε στο μενού τη δυνατότητα να επιλέγεται ο προς εκτέλεση αλγόριθμος.

Μέρος Β: Υλοποίηση ενός Γενετικού Αλγορίθμου για το Πρόβλημα του Χρονοπρογραμματισμού Προσωπικού WHPP

Βάρος: 23% Βαθμού Μαθήματος + 2% πιθανό bonus

3 Χρονοπρογραμματισμός προσωπικού

Στην παρούσα εργασία θα μας απασχολήσει το πρόγραμμα εργασίας σε οργανισμούς και ιδρύματα, όπου ο αριθμός των εργαζομένων είναι μεγάλος και η χρονική διάρκεια του κάθε σχεδιασμού μεταβαλλόμενη. Η δημιουργία του προγράμματος εργασίας απο-τελεί ένα είδος χρονοπρογραμματισμού των ανθρώπινων πόρων που διαθέτει το ίδρυμα ή ο οργανισμός. Σύμφωνα με τις απαιτήσεις καθε χρονικής περιόδου, ορίζεται ένα πλήθος από βάρδιες για κάθε εργάσιμη ημέρα. Στη συνέχεια επιλέγονται από το σύνολο των εργαζομένων αυτοί που θα δουλέψουν σε κάθε μία. Ζητούμενο είναι η κάλυψη των αναγκών να είναι πλήρης και να ικανοποιούνται κάποιοι κανόνες που ενδεχομένως να έχουν τεθεί από τα συμβόλαια των εργαζομένων.

Χαρακτηριστικό των οργανισμών και ιδρυμάτων, είναι ο μεγάλος αριθμός διαθέσιμων εργαζομένων στις εκάστοτε θέσεις εργασιας. Οι βάρδιες του νοσηλευτικού προσωπικού κάθε ημέρας σε ένα νοσοκομείο, μπορούν να καλυφθούν από διαφορετικούς συνδυασμούς διαθέσιμων εργαζομένων, καθώς αυτοί, φυσιολογικά, είναι περισσότεροι από όσους χρειάζεται το νοσοκομείο εκείνη την ημέρα. Πιο συγκεκριμένα, ένα νοσοκομείο μπορεί να απασχολεί πάνω από δέκα νοσηλεύτριες και νοσηλευτές, παρ' όλο που χρειάζεται μόνο τέσσερις καθημερινά. Κάτι αντίστοιχο συμβαίνει και με τις τηλεφωνήτριες και τους τηλεφωνητές σε έναν τηλεπικοινωνιακό οργανισμό, τους υπαλλήλους των λογιστηρίων και σε πλήθος άλλων περιπτώσεων.

Τα παραπάνω αποτελούν τους παράγοντες, που δείχνουν το βαθμό καταλληλότητας ενός προγράμματος. Αν εκφραστούν σε μορφή περιορισμών, όσοι περισσότεροι από αυτούς ικανοποιηθούν, τόσο πιο κατάλληλο είναι το πρόγραμμα. Συνήθως, σε πραγματικές συνθήκες είναι δύσκολο να μην υπάρχει παραβίαση. Για τον λόγο αυτό, διαχωρίζονται σε απόλυτους και ελαστικούς. Οι απόλυτοι οφείλουν να ικανοποιούνται πάντα, ενώ οι ελαστικοί, όποτε αυτό είναι δυνατό. Ως απόλυτοι περιορισμοί μπορούν να θεωρηθούν η πραγματοποιησιμότητα (feasibility) του προγράμματος και κανόνες των συμβολαίων για τις βάρδιες καθώς και η πληρότητα στην κάλυψη. Οι ελαστικοί διαχωρίζονται περαιτέρω μεταξύ τους, σε περισσότερο και λιγότερο σημαντικούς, με την ανάθεση βαρών. Συνεπώς, το κάθε πρόγραμμα αξιολογείται βάσει περιορισμών και εξάγεται μια ανάλογη βαθμολογία από την συνάρτηση καταλληλότητας. Μπορούμε πλέον να προχωρήσουμε σε μια πιο σαφή περιγραφή του προβλήματος.

3.1 Μαθηματική περιγραφή

Αρχικά ορίζονται τρία σύνολα. Ένα σύνολο από n εργαζομένους $E = \{\epsilon_1, \epsilon_2, ..., \epsilon_n\}$ το οποίο κωδικοποιεί τους υπαλλήλους, ένα σύνολο από p ημέρες $H = \{\eta_1, \eta_2, ..., \eta_p\}$ που δείχνει την περίοδο σχεδιασμού, και ένα σύνολο που περιλαμβάνει τις k βάρδιες στις οποίες χωρίζεται μια μέρα, μαζί με ένα σύμβολο που αναπαριστά οτι ο εργαζόμενος είναι ελεύθερος βάρδιας μια συγκεκριμένη ημέρα (το σύμβολο "0" ="άδεια"): $S = \{0, s_1, s_2, ..., s_k\}$.[1]

Καλούμε ανάθεση εργασίας $v_{\epsilon,\eta}$ την βάρδια που έχει ανατεθεί στον κάθε υπάλληλο την κάθε ημέρα, δηλαδή:

$$v_{\epsilon,\eta} = s \tag{1}$$

όταν ο εργαζόμενος ϵ, την ημέρα η, έχει βάρδια s, $\epsilon \in E, \eta \in H, s \in S$. Το σύνολο των αναθέσεων αποτελεί το πρόγραμμα εργασίας και μπορεί να παρασταθεί με έναν πίνακα V, όπου ο αριθμός των γραμμών αντιστοιχεί στο πλήθος των εργαζομένων, ενώ ο αριθμός στηλών στον χρονικό ορίζοντα σχεδιασμού (ημέρες). Η τιμή σε κάθε θέση του πίνακα υποδεικνύει την ανατεθειμένη βάρδια. Ένα παράδειγμα φαίνεται στον Πίνακα 1.

[1] Στο πρόβλημα που θα κληθείτε να λύσετε, όπου υπάρχουν 3 βάρδιες ανά ημέρα, έχουμε $k = 3$ και $S = \{0, 1, 2, 3\}$.

Πίνακας 1: Παράδειγμα προγράμματος εργασίας με 8 εργαζόμενους και δύο βάρδιες, πρωινή (Π) και απογευματινή (Α).

Α/Α	Δ	Τ	Τ	Π	Π	Σ	Κ	Δ	Τ	Τ	Π	Π	Σ	Κ
#1	-	-	Π	Π	-	Π	Π	Α	-	-	Α	Α	-	-
#2	Π	Π	-	-	Π	Α	Α	-	-	-	Α	Α	-	-
#3	-	-	-	Α	Α	-	-	Π	Α	-	-	Π	Π	Π
#4	-	-	Α	Α	Α	-	-	-	Π	Α	-	-	Π	Π
#5	Π	Π	Α	-	-	-	-	Π	Α	-	-	-	Α	Α
#6	-	-	Π	Π	Π	-	-	-	Π	Α	-	-	Α	Α
#7	Α	Α	-	-	-	Α	Α	-	-	Π	Π	Π	-	-
#8	Α	Α	-	-	-	Π	Π	Α	-	Π	Π	-	-	-

Στη συνέχεια ορίζουμε την συνάρτηση ελέγχου ανάθεσης, η οποία μας δίνει την δυνατότητα να ελέγξουμε το αν ένας εργαζόμενος ϵ, την ημέρα η, έχει την βάρδια s, με $\epsilon \in E, \eta \in H, s \in S$:

$$f(\epsilon, \eta, s|V_j) = \begin{cases} 1, & v_{\epsilon,\eta} = s \\ 0, & \text{αλλιώς} \end{cases} \tag{2}$$

Ωστόσο, για διευκόλυνση του ελέγχου περιορισμών του τύπου αν δουλεύει κάποιος γενικά (ανεξαρτήτως τύπου βάρδιας, εξαιρουμένης της 0-άδεια), μπορούμε να εισάγουμε ένα επιπλέον σύμβολο, το Any. Το Any αναπαριστά μια οποιαδήποτε από όλες τις βάρδιες εκτός της ελεύθερης. Συνεπώς, όταν $s = Any$, ελέγχουμε αν ο εργαζόμενος δουλεύει (οποιαδήποτε βάρδια) την ζητούμενη ημέρα. Ο έλεγχος ανάθεσης f, μίας βάρδιας s, σε έναν εργαζόμενο ϵ, τη μέρα η, του προγράμματος εργασίας V_j, μπορεί να δωθεί από μια παραλλαγή της συνάρτησης f:

$$f(\epsilon, \eta, s|V_j) = \begin{cases} 1, & \text{εάν } s \neq Any \text{ και } v_{\epsilon,\eta} = s \\ 1, & \text{εάν } s = Any \text{ και } v_{\epsilon,\eta} \neq 0 \\ 0, & \text{αλλιώς} \end{cases} \tag{3}$$

Ο πρώτος κλάδος λειτουργεί όπως ο πρώτος της Εξ. 2, δηλαδή ελέγχει την ανάθεση συγκεκριμένης βάρδιας. Ο δεύτερος κλάδος, όπου το όρισμα s της συνάρτησης f είναι Any, εξασφαλίζει ότι αν έχει ανατεθεί οτιδήποτε εκτός της άδειας-0, η συνάρτηση f πάλι επιστρέφει 1. Τέλος, επιστρέφουμε την τιμή 0 εφόσον η τιμή του ορίσματος s (από τις $0, s_1, s_2, ..., s_k$) δεν έχει ανατεθεί, αλλά και στην περίπτωση που το όρισμα s είναι Any και η ανάθεση είναι $v_{\epsilon,\eta} = 0$.

3.1.1 Απόλυτος περιορισμός

Στην παρούσα εργασία, θα θεωρήσουμε ως απόλυτο περιορισμό την πληρότητα στις ανάγκες κάλυψης. Ο εργοδότης ορίζει για κάθε ημέρα και για κάθε βάρδια έναν ελάχιστο και έναν μέγιστο αριθμό. Όταν το ελάχιστο και το μέγιστο έχουν την ίδια τιμή, τότε συμπεραίνεται ότι χρειάζονται ακριβώς τόσοι εργαζόμενοι. Άρα, δεδομένων $min_{\eta,s}$ και $max_{\eta,s}$, $\forall\eta$, $\forall s$, πρέπει:[2]

$$min_{\eta,s} \leq \sum_{\epsilon=1}^{n} f(\epsilon, \eta, s) \leq max_{\eta,s}, \forall\eta, \forall s \tag{4}$$

[2]Στο πρόβλημα που θα κληθείτε να λύσετε, τα min και max ταυτίζονται, και ο απόλυτος περιορισμός, δηλαδή οι ακριβείς ανάγκες κάλυψης για κάθε ημέρα και βάρδια, καθορίζονται από τον Πίνακα 6.

3.1.2 Ελαστικοί περιορισμοί

Εκτός από τον απόλυτο περιορισμό της κάλυψης των ζητούμενων θέσεων, μπορεί να υπάρξει πληθώρα ελαστικών περιορισμών, οι οποίοι είναι δυνατό να παραβιάζονται προσθέτοντας βαθμούς ποινής στη βαθμολογία του προκύπτοντος προγράμματος. Στην ουσία, ελαστικοί περιορισμοί αποκαλούνται οι απαιτήσεις που αφορούν το σύνολο των εργαζομένων και ορίζονται ως κανόνες από τα ισχύοντα συμβόλαια. Το σύνολο των περιορισμών πλήθους l το αναπαριστούμε ως $C = \{c_1, c_2, ..., c_l\}$, και τα αντίστοιχα βάρη $W = \{w_1, w_2, ..., w_l\}$.

3.1.3 Αποτίμηση προγράμματος εργασίας προσωπικού

Για να γίνει η αποτίμηση του εκάστοτε προγράμματος εργασίας, δηλαδή ο έλεγχος ποιότητας του προγράμματος ως προς τον αριθμό των περιορισμών που παραβιάζει, αρχικά ελέγχουμε αν παραβιάζεται ο απόλυτος περιορισμός. Εάν όντως παραβιάζεται, η λύση απορρίπτεται. Σε περίπτωση που η λύση είναι πραγματοποιήσιμη (feasible), μπορούμε να προχωρήσουμε στον έλεγχο των ελαστικών περιορισμών.

Για κάθε περιορισμό ελέγχουμε τις αναθέσεις από βάρδιες σε κάθε πιθανή λύση και εφόσον παραβιάζεται κάποιος, προσθέτουμε το βάρος του στη βαθμολογία της αντίστοιχης λύσης. Άρα, η συνάρτηση ελέγχου του ελαστικού περιορισμού c_i για μια πιθανή λύση V_j είναι:

$$k(V_j, c_i, w_i) = \begin{cases} w_i, & \text{εάν ο } c_i \text{ παραβιάζεται στο } V_j \\ 0, & \text{αλλιώς} \end{cases} \tag{5}$$

Η καταλληλότητα είναι αντιστρόφως ανάλογη της βαθμολογίας κάθε λύσης, δηλαδή όσο μικρότερη βαθμολογία, τόσο πιο αποδεκτή και η λύση. Η συνολική βαθμολογία μιας πιθανής λύσης V_j, υπολογίζεται αθροίζοντας όσα περιγράφησαν παραπάνω:

$$TotalPenalty_{V_j} = k(V_j, c_1, w_1) + k(V_j, c_2, w_2) + ... + k(V_j, c_i, w_i) \tag{6}$$

3.1.4 Συνάρτηση καταλληλότητας

Η συνάρτηση καταλληλότητας στοχεύει στην βαθμολόγηση κάθε προγράμματος/ χρωμοσώματος, έτσι ώστε να επιχειρήσουμε την ελαχιστοποίηση της. Δεδομένου του πληθυσμού από προγράμματα εργασίας $\Pi = \{V_1, V_2, ..., V_{pop}\}$, ψάχνουμε το πρόγραμμα V_{min}, που είναι αυτό με τη μικρότερη βαθμολογία $TotalPenalty_{V_{min}}$, δηλαδή ζητούμενο είναι το:

$$\min_{V_j \in \Pi} TotalPenalty_{V_j} \tag{7}$$

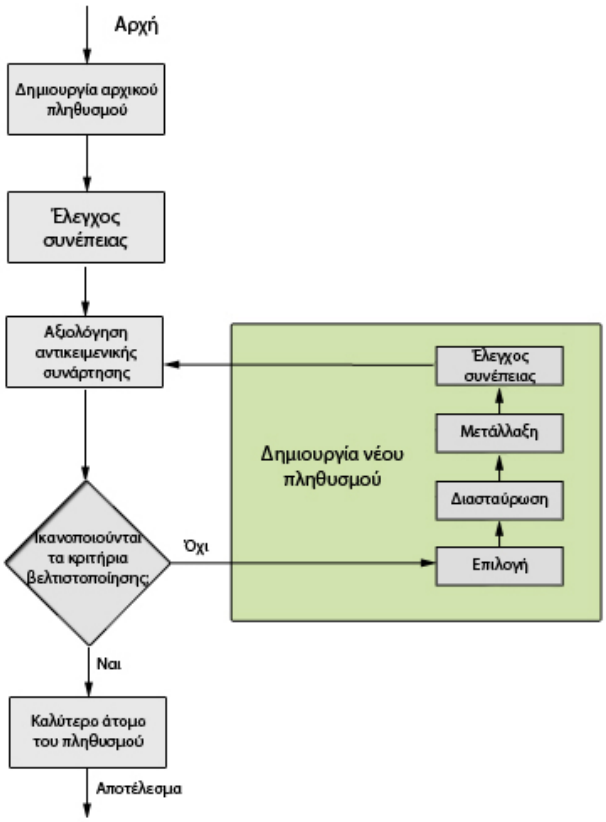
4 Υλοποίηση ενός Γενετικού Αλγορίθμου για το Πρόβλημα του Χρονοπρογραμματισμού του Προσωπικού

Ο γενετικός αλγόριθμος ακολουθεί παρόμοια διαδικασία με αυτή της φυσικής επιλογής, η οποία διέπει τον έμβιο κόσμο. 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. Με αυτόν τον τρόπο, τα προγράμματα εργασίας της επόμενης γενιάς κληρονομούν τα γονίδια αυτά και το ίδιο εφαρμόζεται πάλι για τις επόμενες γενιές. Στον γενετικό αλγόριθμο, αρχικά ορίζεται ο πληθυσμός που αποτελείται από τυχαίες πιθανές λύσεις από τον χώρο αναζήτησης. Η κάθε μία πιθανή λύση αποτελεί ένα χρωμόσωμα. Επιπλέον, το κάθε χρωμόσωμα αποτελείται από τα γονίδια, τα οποία απεικονίζουν με μαθηματικό τρόπο τα χαρακτηριστικά του χρωμοσώματος. Εισάγωντας κάθε χρωμόσωμα στην συνάρτηση καταλλλότητας, αυτή επιστρέφει μια βαθμολογία που αντιπροσωπεύει την συνολική καταλληλότητα των γονιδίων του χρωμοσώματος. Έπειτα,

σύμφωνα με την βαθμολογία αυτή, επιλέγονται τα καλύτερα για να διασταυρωθούν. Στη διαδικασία της *διασταύρωσης*, δημιουργείται μια λύση "απόγονος", συνθέτοντας τα γονίδια των επιλεγμένων "γονέων".

Για να αποφευχθεί η πλήρης ομογενοποίηση που μπορεί να προκύψει από την παραπάνω υπολογιστική διαδικασία, εισάγεται άλλη μια έννοια, που επιτρέπει την δημιουργία νέων γονιδίων στα χρωμοσώματα, η *μετάλλαξη*. Στις αλληλουχίες των βάσεων του DNA των έμβιων οργανισμών συχνά παρουσιάζονται αλλαγές, τις οποίες τις προκαλούν διάφοροι παράγοντες, εξωτερικοί και εσωτερικοί. Η έννοια της μετάλλαξης εισάγεται στον αλγόριθμο με τον εξής τρόπο: με μια μικρή πιθανότητα αλλάζει η τιμή σε ένα τυχαίο γονίδιο, μετατρέποντας έτσι το χρωμόσωμα σε ένα διαφορετικό, το οποίο επίσης όμως θα πρέπει να ανήκει στον χώρο αναζήτησης (feasible). Συνεπώς, εισάγοντας τη μετάλλαξη δίνεται η δυνατότητα σχηματισμού λύσεων με γονίδια διαφορετικά από αυτά του αρχικού πληθυσμού, αυξάνοντας έτσι το εύρος αναζήτησης.

Οι διαδικασίες της *επιλογής*, της *διασταύρωσης* και της *μετάλλαξης* επαναλαμβάνονται, δημιουργώντας γενιές από χρωμοσώματα, η ποιότητα των οποίων αναμένεται να βελτιωθεί με την πάροδο του χρόνου. Ο συνολικός χρόνος εκτέλεσης ενός γενετικού αλγορίθμου, εξαρτάται κατά κύριο λόγο από τον σχεδιαστή. Μπορεί να θέσει διάφορα κριτήρια για τον τερματισμό του, καθώς δεν υπάρχει ένα γενικό που να λειτουργεί σε όλες τις περιπτώσεις. Το ίδιο ισχύει, όχι μόνο για τον τερματισμό, αλλά και για την επιλογή, τη διασταύρωση και τη μετάλλαξη. Υπάρχουν διάφορες προτάσεις υλοποίησης και ο σχεδιαστής καλείται να επιλέξει κάθε φορά τον συνδυασμό αυτών που αποδίδουν καλύτερα. Καταλήγοντας, ο αλγόριθμος που ζητείται (Αλγ. 2.1), θα πρέπει να ακολουθεί το διάγραμμα ροής της Εικ. 2.



Εικ. 2: Το διάγραμμα ροής του ζητούμενου γενετικού αλγορίθμου.

4.1 Δημιουργία αρχικού πληθυσμού

Όσον αφορά τη δημιουργία του αρχικού πληθυσμού πάνω στον οποίο θα εφαρμοστεί ο γενετικός αλγόριθμος, προκύπτουν δύο ερωτήματα. Το πρώτο αφορά το μέγεθος του πληθυσμού *pop* και το δεύτερο τον τρόπο με τον οποίο θα δημιουργηθεί.

Η επιλογή του μεγέθους του πληθυσμού έχει προσεγγιστεί από πολλές θεωρητικές απόψεις, παρ' όλο που είναι πάνω-κάτω γνωστό το ότι αυτή έγκειται στην εξισορρόπιση αποδοτικότητας και αποτελεσματικότητας. Γενικά ένας σχετικά μικρός πληθυσμός μπορεί να μην επαρκεί για την πλήρη αναζήτηση του χώρου, ενώ ένας μεγάλος να μην καταφέρνει να συγκλίνει σε αποδεκτή λύση.

Όσον αφορά το πώς δημιουργείται ο αρχικός πληθυσμός, είναι γενικά αποδεκτό ότι πρέπει να γίνεται "τυχαία", το οποίο στην πράξη σημαίνει μέσω ψευδοτυχαίων ακολουθιών. Ως *χρωμόσωμα* στην παρούσα υλοποίηση, θα θεωρείται ένα *πρόγραμμα εργασίας*, διαισθητικά δηλαδή ένας πίνακας με τόσες γραμμές όσοι οι εργαζόμενοι, και τόσες στήλες όσες είναι οι ημέρες σχεδιασμού. Οι τιμές που λαμβάνει η κάθε θέση αποτελούν και τη βάρδια την οποία δουλεύει ο αντίστοιχος εργαζόμενος, την αντίστοιχη ημέρα. Με '0' συμβολίζεται η ελεύθερη ημέρα, με '1' η πρωινή βάρδια, με '2' η απογευματινή και με '3' η βραδυνή. Τα *pop* σε αριθμό διαφορετικά προγράμματα εργασίας/χρωμοσώματα, αποτελούν τον πληθυσμό. Ένα παράδειγμα χρωμοσώματος, φαίνεται στην Εικ. 2. Ο 3ος εργαζόμενος αρχίζει με απογευματινή βάρδια, μετά του έχουν ανατεθεί δύο πρωινές, ακολουθούν 2 ρεπό, μια πρωινή και μια απογευματινή. Σημειώστε οτι σε κάθε στήλη (ημέρα) υπάρχει ακριβώς ένας εργαζόμενος ανά βάρδια.

0	3	0	1	2	3	0
1	2	3	0	0	0	3
2	1	1	0	0	1	2
3	0	2	3	3	0	0
0	0	0	2	1	2	1

Πίνακας 2: Παράδειγμα χρωμοσώματος με 5 εργαζόμενους, 3 βάρδιες, για ορίζοντα 7 ημερών.

4.2 Αξιολόγηση - Συνάρτηση Καταλληλότητας

Η συνάρτηση καταλληλότητας (fitness function) είναι αυτή που διακρίνει τις επιθυμητές λύσεις, από τις μη επιθυμητές. Αυτό επιτυγχάνεται με την ανάθεση μιας βαθμολογίας σε κάθε χρωμόσωμα, η οποία αντιπροσωπεύει το βαθμό που αυτό είναι επιθυμητό. Κάθε χρωμόσωμα ελέγχεται αρχικά για το αν είναι συνεπές (feasible), δηλαδή αν ικανοποιείται ο πίνακας κάλυψης. Σε περίπτωση που δεν ικανοποιείται, αποκλείεται, και δεν προχωρά σε βαθμολόγηση. Στην αντίθετη περίπτωση, συνεχίζουμε στον έλεγχο των ελαστικών περιορισμών, όπου για κάθε ένα που παραβιάζεται, προσθέτουμε το βάρος του στη βαθμολογία του χρωμοσώματος που τον παραβιάζει.

4.3 Επιλογή

Η επιλογή, είναι η διαδικασία κατά την οποία επιλέγονται τα καταλληλότερα άτομα (=χρωμοσώματα) για τη δημιουργία του επόμενου πληθυσμού, και αποτελεί έναν από τους λεγόμενους *γενετικούς τελεστές*. Αυτό που επιτυγχάνεται με την επιλογή, είναι η επικράτηση των καλών γονιδίων και η διάδοσή τους σε επόμενες γενιές. Δίχως τον γενετικό τελεστή της επιλογής, ένας γενετικός αλγόριθμος θα έκανε απλά τυχαία αναζήτηση.

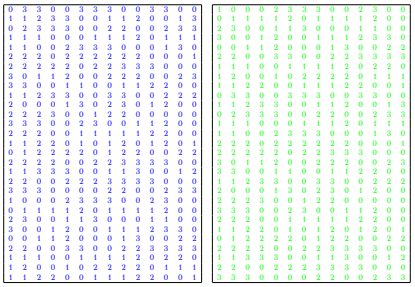
Υπάρχουν διάφοροι τρόποι επιλογής. Ένας είναι με στοχαστική δειγματοληψία, ή επιλογή με χρήση ρουλέτας (roulette wheel selection). Σε αυτόν τον τρόπο, ένας τροχός χωρίζεται σε κομμάτια, μικρότερα ή μεγαλύτερα, ανάλογα με τη βαθμολογία των χρωμοσωμάτων. Περιστρέφοντας τον τροχό, ο δείκτης σταματά πάνω από ένα

συγκεκριμένο κομμάτι. Έτσι επιλέγεται το αντίστοιχο χρωμόσωμα. Είναι φανερό, ότι όσο μεγαλύτερη είναι η βαθμολογία ενός χρωμοσώματος τόσο πιο πιθανό είναι να επιλεχθεί.

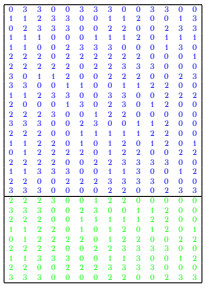
Άλλοι τρόποι είναι η στοχαστική καθολική δειγματοληψία, η τοπική επιλογή (local selection), η επιλογή αποκοπής (truncation selection), η επιλογή με τουρνουά (tournament selection), η επιλογή σταθερής κατάστασης (steady state selection) και η ταξινομημένη επιλογή.

Μία τεχνική για την επιλογή που αποδίδει πολλές φορές, είναι ο ελιτισμός (elitism). Κατά τις διαδικασίες που προαναφέρθηκαν, είναι πιθανό να μην επιλεγεί το ικανότερο χρωμόσωμα του πληθυσμού. Επιλέγοντας λοιπόν κατευθείαν το καλύτερο και περνώντας το στην επόμενη γενιά δίχως αλλαγές, εξασφαλίζουμε τη διάδοση των καλών γονιδίων. Παρ'όλα αυτά,ο ελιτισμός μπορεί να οδηγήσει στην πρόωρη σύγκλιση του αλγόριθμου σε τοπικά ακρότατα.

Πίνακας 3: Χρωμοσώματα γονείς (A, B)



Πίνακας 4: Πιθανός απόγονος από A και B

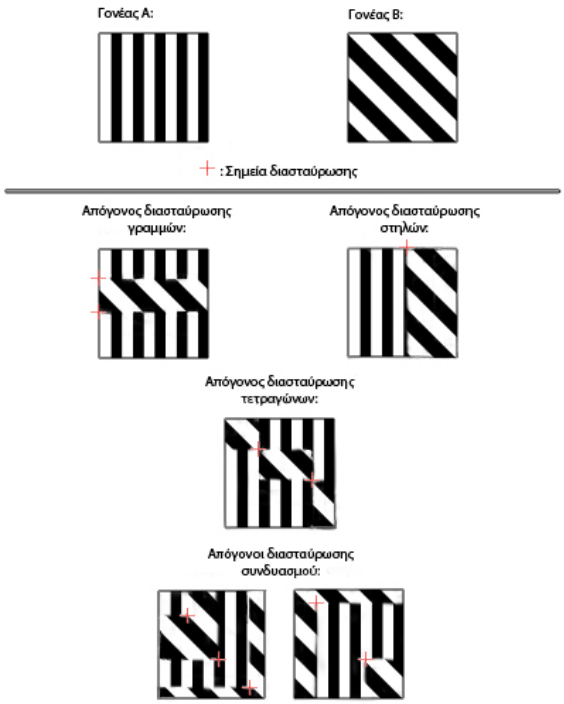


4.4 Διασταύρωση

Ένας άλλος γενετικός τελεστής, είναι η διασταύρωση (crossover). Στο στάδιο της διασταύρωσης, κάθε ένα από τα επιλεγμένα χρωμοσώματα υπάρχει πιθανότητα να διασταυρωθεί με ένα άλλο, αλλιώς παραμένει ως έχει. Υπάρχουν πολλοί τρόποι διασταύρωσης, οι οποίοι έχουν πάντα τον ίδιο σκοπό: Την κληροδότηση γονιδίων από δύο γονικά χρωμοσώματα, σε ένα χρωμόσωμα απόγονο.

Στη διασταύρωση ενός σημείου (one-point crossover), επιλέγεται ένα σημείο μέσα στα χρωμόσωματα, το οποίο καθορίζει ποια γονίδια θα κληρονομηθούν από τον ένα γονέα, και ποια από τον άλλον. Πιο συγκεκριμένα, από την αρχή του χρωμοσώματος έως το σημείο που επιλέγεται, τα γονίδια του απογόνου είναι του ενός γονέα, και από το επιλεγμένο σημείο έως το τέλος του χρωμοσώματος, είναι του άλλου. Όταν θέλουμε περισσότερες

εναλλαγές μεταξύ των γονιδίων των γονέων, επιλέγουμε πολυσημειακή διασταύρωση (multi-point crossover). Διαφορετικοί τρόποι διασταύρωσης φαίνονται στην Εικ. 3.[3]



Εικ. 3: Μέθοδοι διασταύρωσης.

4.5 Μετάλλαξη

Ο τελεστής της μετάλλαξης καλείται να εμποδίσει την ομογενοποίηση του πληθυσμού των χρωμοσωμάτων που ίσως προκληθεί σε επόμενες γενιές από την διασταύρωση, εισάγοντας αυθαίρετη διαφοροποίηση. Συνήθως, εφαρμόζεται ανά γονίδιο, με μια μικρή πιθανότητα. Όταν επιτυγχάνεται η πιθανότητα αυτή, τότε η τιμή του γονιδίου αλλάζει σε μια διαφορετική, από το πεδιο ορισμού.

Σκοπός της παραπάνω διαδικασίας είναι η διατήρηση της διαφορετικότητας. Η μετάλλαξη πρέπει να επιτρέπει στον αλγόριθμο να ξεφεύγει από τοπικά ακρότατα, εμποδίζοντας τα χρωμοσώματα από το να γίνουν όμοια μεταξύ τους. Το τελευτάιο μπορεί να καθυστερήσει, ή ακόμα και να σταματήσει τελείως την εξέλιξη.

Διαφορετικά είδη μετάλλαξης φαίνονται στην Εικ. 4

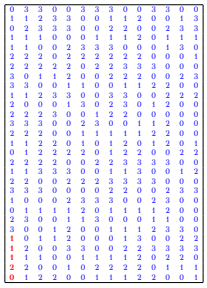
4.6 Επανάληψη - Κριτήρια τερματισμού

Η όλη λειτουργία του γενετικού αλγόριθμου βασίζεται στις γενιές χρωμοσωμάτων. Ξεκινώντας με τον αρχικό πληθυσμό, εφαρμόζονται οι γενετικοί τελεστές και δημιουργείται ένας καινούργιος, ο οποίος αποτελεί την επόμενη γενιά. Σε αυτήν εφαρμόζονται εκ νέου οι γενετικοί τελεστές, καταλήγοντας σε μία νέα γενιά. Το πότε θα τερματίζει ο αλγόριθμος εξαρτάται από τον σχεδιαστή. Αυτό μπορεί να γίνεται μετά την πάροδο είτε συγκεκριμένου αριθμού γενεών, ή συγκεκριμένης χρονικής διάρκειας. Ένα άλλο κριτήριο τερματισμού είναι και η διαφορά στη βαθμολογία. Αν οι γενιές περνούν δίχως να βελτιώνεται η βαθμολογία πάνω από ένα ποσοστό,

[3]Χρησιμοποιώντας διασταύρωση κατά γραμμές μπορεί να χαλάσουμε το feasibility, ενώ με διασταύρωση κατά στήλες όχι.

Εικ. 4: Διάφορα είδη μετάλλαξης

Πίνακας 5: Μετάλλαξη στο χρωμόσωμα A (inversion)



μπορεί να θεωρηθεί ότι η εξέλιξη έχει σταματήσει, συνεπώς ο αλγόριθμος τερματίζει. Τελικά, η λειτουργία ενός απλού γενετικού αλγορίθμου περιγράφεται στον παρακάτω αλγόριθμο:

Algorithm 4.1: ΑπλοσΓενετικοσΑλγοριθμος()

```
Δημιουργία αρχικού πληθυσμού
Έλεγχος συνέπειας
Υπολογισμός τιμής συνάρτησης καταλληλότητας για κάθε χρωμόσωμα
while not Ολοκληρωμένος
do { comment: Δημιουργία νέας γενιάς
     for i ← 1 to Μέγεθος πληθυσμού/2
     do { Επιλογή δύο καλών χρωμοσωμάτων για διασταύρωση
          do { Διασταύρωση και δημιουργία νέων χρωμοσωμάτων
               Εφαρμογή μετάλλαξης στα νέα χρωμοσώματα
               Έλεγχος συνέπειας
               Εισαγωγή των χρωμοσωμάτων στη νέα γενιά
     if Ικανοποίηση κριτηρίου τερματισμού
       then Ολοκληρωμένος ← true
```

5 Το Πρόβλημα WHPP

Στο πρόβλημα των Weil, G., K. Heus, P. Francois, και M. Poujade[4] έχουμε τριάντα εργαζόμενους και διάρκεια δεκατεσσάρων ημερών. Υπάρχουν περιορισμοί συμβολαίου, και λαμβάνονται υπ'όψιν και οι ώρες που διαρκεί η κάθε βάρδια, θέτωντας άνω όριο τις εβδομήντα για κάθε εργαζόμενο. Η πρωινή και η απογευματινή βάρδια διαρκούν οκτώ ώρες και η βραδινή δέκα. Έχουμε τον απόλυτο περιορισμό του πίνακα κάλυψης, δηλαδή για κάθε μέρα θα πρέπει να υπάρχουν ακριβώς τόσοι εργαζόμενοι, όσοι ζητούνται σε κάθε βάρδια. Οι διαστάσεις και οι περιορισμοί του προβλήματος φαίνονται στα παρακάτω:

Διαστάσεις	
Διάρκεια	14
Εργαζόμενοι	30

Περιορισμοί (Soft Constraints)	
Περιορισμός	*Βάρος*
Max 70 ώρες εργασίας	1000
Max 7 συνεχόμενες ημέρες εργασίας	1000
Max 4 συνεχόμενες νυκτερινές βάρδιες	1000
Να αποφεύγεται ανάθεση βάρδιας νύκτα και την επόμενη ημέρα πρωί	1000
Να αποφεύγεται ανάθεση βάρδιας απόγευμα και την επόμενη ημέρα πρωί	800
Να αποφεύγεται ανάθεση βάρδιας νύκτα και την επόμενη ημέρα απόγευμα	800
Τουλάχιστον 2 ημέρες άδειας μετά από 4 νυκτερινές	100
Τουλάχιστον 2 ημέρες άδειας μετά από 7 ημέρες εργασίας	100
Να αποφεύγεται ανάθεση βάρδιας εργασία-άδεια-εργασία	1
Να αποφεύγεται ανάθεση βάρδιας άδεια-εργασία-άδεια	1
Εργασια το πολύ ένα Σαββ/κο (από τα δύο του χρονικού ορίζοντα)	1

Πίνακας 6: Απόλυτος περιορισμός

Πίνακας κάλυψης (Hard Constraint)							
	ΔΕΥ	ΤΡΙ	ΤΕΤ	ΠΕΜ	ΠΑΡ	ΣΑΒ	ΚΥΡ
Πρωινή	10	10	5	5	5	5	5
Απογευμ.	10	10	10	5	10	5	5
Βραδινή	5	5	5	5	5	5	5

6 Ζητούμενα

1. Υλοποιείστε έναν σκελετό γενετικού αλγορίθμου, όπως αυτόν που περιγράφεται από τον Αλγ. 2.1. και την Εικ. 2.
2. Σκεφτείτε και υλοποιείστε μια συνάρτηση αρχικοποίησης του πληθυσμού. Είναι συνεπή (feasible) τα χρωμοσώματα που παράγει;
3. Υλοποιείστε μια συνάρτηση ελέγχου συνέπειας των χρωμοσωμάτων (έλεγχος απόλυτου περιορισμού), βάσει των απαιτήσεων του WHPP.
4. Υλοποιείστε μια μέθοδο υπολογισμού βαθμολόγησης (συνάρτηση καταλληλότητας) βάσει των απαιτήσεων του WHPP.
5. Υλοποιείστε μια μέθοδο επιλογής.
6. Σκεφθείτε και υλοποιείστε 2 διαφορετικές μεθόδους διασταύρωσης.
7. Σκεφθείτε και υλοποιείστε 2 διαφορετικές μεθόδους μετάλλαξης.
8. Υλοποιείστε και αναφέρετε ένα ή περισσότερα κριτήρια τερματισμού.
9. Πειραματιστείτε με διάφορες τιμές στον αρχικό πληθυσμό pop, στον μέγιστο αριθμό επαναλήψεων $iter_{max}$, και στις πιθανότητες επιλογής p_{sel}, διασταύρωσης p_{cross} και μετάλλαξης p_{mut}.**Επιλέξτε μετά από πειραματισμούς δύο τέτοια σετ τιμών**.
10. Εκτελέστε τουλάχιστον 5 φορές για κάθε συνδυασμό επιλογών τιμών **(από τα παραπάνω δυο επιλεγμένα σετ τιμών)** και μεθόδων διασταύρωσης και μετάλλαξης, και παρουσιάστε τους μέσους όρους των βαθμολογιών των καλύτερων χρωμοσωμάτων που προέκυψαν (άρα για κάθε έναν απο $2 \times 2 \times 2 = 8$ συνδυασμούς).
11. Παρουσιάστε σε γραφήματα το μέσο όρο βαθμολογίας των χρωμοσωμάτων ανά γενιά, καθώς και τη βαθμολογία του καλύτερου χρωμοσώματος ανά γενιά.
12. **BONUS (2% βαθμού μαθήματος)**: Υλοποιείστε μια μέθοδο τοπικής αναζήτησης και ενσωματώστε τη στον αλγόριθμο, μετά το στάδιο της μετάλλαξης, με πιθανότητα ενεργοποίησης p_{loc}. Αναφέρετε τα αποτελέσματα.

References

[1] *Peter Brucker*, *Rong Qu*, *Edmund Burke*, *Gerhard Post* , **A decomposition, construction and post-processing aproach for nurse rostering**.

[2] *B. Cheang*, *H. Li*, *A. Lim*, *B. Rodrigues*. **Nurse rostering problems - a bibliographic survey.**

[3] *Edmund K. Burke*, *Patrick de Causmaecker*, *Greet vanden Berghe*, *Hendrik van Landeghem*. **The state of the art of nurse rostering.**

[4] *Weil*, *G.*, *K. Heus*, *P. Francois*, *and M. Poujade*, **Constraint programming for Nurse scheduling**, in IEEE Engineering in Medicine and Biology, 1995, σελ. 417–422.

[5] *Colin R. Reeves*, *Jonathan E. Rowe*, **Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory**, Operations Research/Computer Science Interfaces Series Springer, 2002