ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# ΜΕΛΕΤΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΤΗΝ ΕΞΑΓΩΓΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΔΙΔΑΣΚΑΛΙΑΣ ΤΟΥ ΤΜΗΜΑΤΟΣ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΤΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

**ΕΛΕΝΗ ΛΕΒΕΝΤΗ**

**ΕΠΙΒΛΕΠΩΝ:**

**Δρ. ΜΗΝΑΣ ΔΑΣΥΓΕΝΗΣ**

**ΜΕΛΗ ΕΞΕΤΑΣΤΙΚΗΣ ΕΠΙΤΡΟΠΗΣ:**

**Δρ. ΜΗΝΑΣ ΔΑΣΥΓΕΝΗΣ**

**Δρ. ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΤΕΡΓΙΟΥ**

**ΚΟΖΑΝΗ**

**ΙΟΥΝΙΟΣ, 2012**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# ΜΕΛΕΤΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΤΗΝ ΕΞΑΓΩΓΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΔΙΔΑΣΚΑΛΙΑΣ ΤΟΥ ΤΜΗΜΑΤΟΣ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΤΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΕΛΕΝΗ ΛΕΒΕΝΤΗ

**ΕΠΙΒΛΕΠΩΝ:**

Δρ. ΜΗΝΑΣ ΔΑΣΥΓΕΝΗΣ

**ΜΕΛΗ ΕΞΕΤΑΣΤΙΚΗΣ ΕΠΙΤΡΟΠΗΣ:**

Δρ. ΜΗΝΑΣ ΔΑΣΥΓΕΝΗΣ

Δρ. ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΤΕΡΓΙΟΥ

**ΚΟΖΑΝΗ**

**ΙΟΥΝΙΟΣ, 2012**

*"A general algorithm is like a size 48 cloth. It will cover everybody but it doesn't fit any-one very well"*

*Kathryn A. Dowsland*

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Μηνά Δασυγένη, για την επίβλεψη και καθοδήγησή του στην εκπόνηση της παρούσας διπλωματικής εργασίας. Η μεθοδικότητα του και το ενδιαφέρον του συντέλεσαν καταλυτικά στη διαμόρφωση του τελικού αποτελέσματος.

# *ΠΕΡΙΛΗΨΗ*

Η υλοποίηση ωρολογίου προγράμματος για εκπαιδευτικά ιδρύματα αποτελεί ένα αρκετά σύνθετο πρόβλημα. Η χειρονακτική υλοποίηση του απαιτεί πολύ κόπο και χρόνο, καθώς θα πρέπει να ικανοποιεί αρκετούς περιορισμούς. Στην παρούσα διπλωματική, αναπτύχθηκε η δημιουργία μιας εφαρμογής, για υλοποίηση του ωρολογίου προγράμματος του Πανεπιστημίου Δυτικής Μακεδονίας, καθώς και για άλλα ιδρύματα που έχουν συμβατά κριτήρια με αυτό. Σκοπός της εφαρμογής είναι η ανάθεση των διαλέξεων, αλλά και των εργαστηρίων όλων των μαθημάτων στο χρόνο και στις αίθουσες. Χρησιμοποιήθηκαν για την ανάπτυξη του διάφοροι ευρετικοί αλγόριθμοι αναζήτησης, των οποίων τα αποτελέσματα εμφανίζονται στην οθόνη, ενώ ταυτόχρονα αποθηκεύονται σε κατάλληλα αρχεία. Για την ανάπτυξη των αλγορίθμων, λήφθηκε υπόψη ένα μεγάλο πλήθος κριτηρίων, τα οποία είναι απαραίτητα για την ορθή σχεδίαση του ωρολογίου προγράμματος. Κάθε αλγόριθμος, διαλέγει τη σειρά με την οποία θα καταχωρηθούν τα μαθήματα στο πρόγραμμα, ανάλογα με τα κριτήρια που έχουν οριστει για τον καθένα τους.

Ο σχεδιασμός του μοντέλου πραγματοποιήθηκε με χρήση γλώσσας προγραμματισμού C++ μέσω του Microsoft Visual Studio 2010. Η εφαρμογή υλοποιήθηκε σε 6000 γραμμές κώδικα, με χρήση 29 συναρτήσεων και οι απαιτήσεις μνήμης, κατά τη διάρκεια εκτέλεσης του προγράμματος, είναι αμελητέες καθώς το μέγεθος των πινάκων δεν ξεπερνά τα 50KB.

*Design and Construction of an Informative System for the elaboration of a teaching programme for the Information Technology and Telecommunications Engineering Department of the University of Western Macedonia.*

# *ABSTRACT*

The materialization of the timetable of institutes is a quite complex problem. Its manual materialization requires a lot of effort and time, as it should satisfy considerable restrictions. In the present thesis, the creation of an application for the materialization of the timetable of the University of Western Macedonia was developed, as well as for others institutions holding compatible criteria to the University of Western Macedonia. The purpose of the application is to allocate the lectures, as well as the workshops of the courses according to the teaching time and the classrooms available. Various heuristic algorithms were used for the development of the application and their results appear on the screen while at the same time, they are saved into the appropriate files. For the development of the algorithms, a great deal of criteria was taken into account , which are essential for the proper design of the timetable. Each algorithm picks the order the courses will be integrated in the timetable, according to the criteria set for each of them.

The design of the model was accomplished with the use of the programming language C++ through the Microsoft Visual Studio 2010. The application was materialized in 6000 code lines together with the use of 29 functions and the demands of saving space, during program execution, are negligible, as the size of the tables does not exceed that of 50 KB.

# *ΠΕΡΙΕΧΟΜΕΝΑ*

**ΕΙΣΑΓΩΓΗ ..... 10**

1.1 Ιστορικη αναδρομη ..... 12
1.2 Αλγοριθμος GRASP ..... 12
1.3 Δομη εργασιας ..... 14

**ΧΡΟΝΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ..... 15**

2.1 Πολυπλοκοτητα χρονοπρογραμματισμου ..... 17
2.2 Προβληματα Ικανοποιησης Περιορισμων με πεπερασμενα πεδια ..... 18
2.3 Προβληματα Ικανοποιησης Περιορισμων με μη πεπερασμενα πεδια ..... 22
2.4 Ειδη περιορισμων ..... 23
2.5 Αλγοριθμοι αναζητησης ..... 25

**ΚΑΤΑΡΤΙΣΗ ΩΡΟΛΟΓΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ..... 31**

3.1 Γενικευμενο προβλημα αναθεσης τιμων ..... 32
3.2 Τα δεδομενα του προβληματος ..... 33
3.3 Αυστηροι περιορισμοι προγραμματος ..... 34

**ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ..... 39**

4.1 Δεδομενα και περιορισμοι προβληματος ..... 39
4.2 Μεταβλητες προγραμματος ..... 41
4.3 Περιγραφη εφαρμογης ..... 43

**ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ ..... 54**

5.1 Ευκολο Προβλημα ..... 56
5.2 Μετριο προβλημα ..... 58
5.3 Δυσκολο προβλημα ..... 60

**ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΟΠΤΙΚΕΣ ..... 63**

**ΒΙΒΛΙΟΓΡΑΦΙΑ ..... 66**

**ΠΑΡΑΡΤΗΜΑΤΑ ..... 68**

## ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχημα 1: Το προβλημα του χρωματιστου χαρτη ..... 19
Σχημα 2: Αναπαρασταση περιορισμων με γραφο ..... 20
Σχημα 3: Το προβλημα των 4ων βασιλισσων ..... 21
Σχημα 4: Το προβλημα του κρυπταριθμητικου αινιγματος (α) και αναπαρασταση του με γραφο (β) ..... 24
Σχημα 5: Γραφημα αλγοριθμου οπισθοδρομησης ..... 27
Σχημα 6: Διαγραμμα ροης της εφαρμογης ..... 43
Σχημα 7: Διαγραμμα ροης συναρτησης programA() ..... 50
Σχημα 8: Διαγραμμα ροης συναρτησης programB() ..... 51
Σχημα 9: Διαγραμμα ροης συναρτησης class_lessons() ..... 52

## ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικονα 1: Γραφικη παρασταση αλγοριθμου τοπικης αναζητησης ..... 28
Εικονα 2: Εικονα περιορισμου 2 ..... 35
Εικονα 3: Εικονα περιορισμου 2 ..... 35
Εικονα 4: Παραθυρο εκτελεσης της εφαρμογης ..... 44
Εικονα 5: Μορφη αρχειου .txt για εισοδο δεδομενων ..... 45

## ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΙΝΑΚΑΣ 1: ΠΙΝΑΚΑΣ ΠΕΡΙΟΡΙΣΜΟΥ 1 ..... 35
ΠΙΝΑΚΑΣ 2: ΠΑΡΑΔΕΙΓΜΑ ΠΕΡΙΟΡΙΣΜΟΥ 5 ..... 37
ΠΙΝΑΚΑΣ 3: ΑΝΤΙΣΤΟΙΧΗΣΗ ΧΡΟΝΟΘΥΡΙΔΩΝ ΜΕ ΑΡΙΘΜΟΥΣ ..... 42
ΠΙΝΑΚΑΣ 4: ΔΕΔΟΜΕΝΑ ΕΥΚΟΛΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 56
ΠΙΝΑΚΑΣ 5: ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΥΚΟΛΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 57
ΠΙΝΑΚΑΣ 6: ΔΕΔΟΜΕΝΑ ΜΕΤΡΙΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 58
ΠΙΝΑΚΑΣ 7: ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕΤΡΙΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 59
ΠΙΝΑΚΑΣ 8: ΔΕΔΟΜΕΝΑ ΔΥΣΚΟΛΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 60
ΠΙΝΑΚΑΣ 9: ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΔΥΣΚΟΛΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 61

## ΚΑΤΑΛΟΓΟΣ ΠΑΡΑΡΤΗΜΑΤΩΝ

ΠΑΡΑΡΤΗΜΑ 1: ΑΝΑΘΕΣΕΙΣ (%) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΕΥΚΟΛΟ ΠΡΟΒΛΗΜΑ ..... 69
ΠΑΡΑΡΤΗΜΑ 2: ΧΡΟΝΟΙ ΕΚΤΕΛΕΣΗΣ (MSEC) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΕΥΚΟΛΟ ΠΡΟΒΛΗΜΑ ..... 70
ΠΑΡΑΡΤΗΜΑ 3: ΑΝΑΘΕΣΕΙΣ (%) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΜΕΤΡΙΟ ΠΡΟΒΛΗΜΑ ..... 71
ΠΑΡΑΡΤΗΜΑ 4: ΧΡΟΝΟΙ ΕΚΤΕΛΕΣΗΣ (MSEC) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΜΕΤΡΙΟ ΠΡΟΒΛΗΜΑ ..... 72
ΠΑΡΑΡΤΗΜΑ 5: ΑΝΑΘΕΣΕΙΣ (%) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΔΥΣΚΟΛΟ ΠΡΟΒΛΗΜΑ ..... 74
ΠΑΡΑΡΤΗΜΑ 6: ΧΡΟΝΟΙ ΕΚΤΕΛΕΣΗΣ (MSEC) ΠΡΟΣΟΜΟΙΩΣΕΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟ ΔΥΣΚΟΛΟ ΠΡΟΒΛΗΜΑ ..... 76
ΠΑΡΑΡΤΗΜΑ 7: ΕΝΑ ΑΠΟ ΤΑ ΑΡΧΕΙΑ ΕΙΣΟΔΟΥ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΕΥΚΟΛΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... 77

# ΚΕΦΑΛΑΙΟ 1

## ΕΙΣΑΓΩΓΗ

Η αυτόματη κατάρτιση ωρολογίου προγράμματος αποτελεί ένα πολύ δύσκολο πρόβλημα τόσο λόγω του μεγάλου αριθμού δεδομένων, που μπορεί να δέχεται όσο και λόγω του μεγάλου αριθμού ετερογενών περιορισμών, που πρέπει να ικανοποιούνται. Η δημιουργία ενός ωρολογίου προγράμματος μπορεί να οριστεί ως ένα πρόβλημα χρονικού προγραμματισμού. Στο πρόβλημα αυτό υπάρχει ένας συγκεκριμένος αριθμός διαλέξεων, κάθε μία από αυτές μπορεί να ανήκει σε μία ή και περισσότερες ομάδες μαθημάτων, που πρέπει να ανατεθεί στο χρόνο και γενικότερα υπάρχουν αρκετοί περιορισμοί, άλλοι αυστηροί και άλλοι λιγότερο.

Οι “σκληροί” περιορισμοί είναι αυστηροί και αφορούν ζητήματα πολύ σημαντικά για την εύρυθμη λειτουργία του Πανεπιστήμιου . Για παράδειγμα ένας τέτοιος περιορισμός είναι ότι η κάθε διάλεξη απαιτεί κάποιους πόρους όπως οι αίθουσες και οι καθηγητές. Οι πόροι αυτοί είναι περιορισμένοι και για το λόγο αυτό πρέπει να λαμβάνονται υπόψιν στην υλοποίηση του προγράμματος. Όπως επίσης, για παράδειγμα αν δυο μαθήματα διδάσκονται από τον ίδιο καθηγητή δε μπορούν να συμπίπτουν χρονικά. Οι “εύκολοι” περιορισμοί είναι λιγότερο αυστηροί, και μπορεί να

μην ικανοποιούνται απαραίτητα, όπως για παράδειγμα ένα μάθημα, που παρακολουθείται από μεγάλο αριθμό φοιτητών, πρέπει να ανατίθεται σε αίθουσα μεγάλης χωρητικότητας.

Γενικά η κατάρτιση ωρολογίου προγράμματος ανήκει στις τάξεις των NP-Complete προβλημάτων, δηλαδή στα προβλήματα που λύνονται σε πολυωνυμικό χρόνο και ο χρόνος αναζήτησης λύσης αυξάνει εκθετικά σε σχέση με τον αριθμό των δεδομένων. Για αυτό λοιπόν η λύση του προβλήματος αυτού είναι πολύ σημαντικό ζήτημα για την αποδοτικότητα της εφαρμογής, τόσο το μοντέλο υλοποίησης του όσο και ο αλγόριθμος αναζήτησης λύσης που θα χρησιμοποιηθεί. Λύσεις για τη διεκπεραίωση του μπορεί να υπάρξουν πολλές ή και καμία. Στην πρώτη περίπτωση η καλύτερη ή μία πολύ καλή λύση μπορεί να βρεθεί σε σχέση με τις υπόλοιπες εφικτές λύσεις. Από την άλλη, εάν το πρόβλημα έχει καθοριστεί έτσι ώστε να μην υπάρχει εφικτή λύση, τότε θα πρέπει να αναγνωριστεί, πιθανών μέσω μιας εξαντλητικής αναζήτησης του χώρου.

Βασικο είναι να τονιστεί  ότι δεν είναι δυνατή η ύπαρξη μοντέλου για το πρόβλημα κατάρτισης ωρολογίου προγράμματος, που να μπορεί να ανταποκρίνεται στις απαιτήσεις όλων των εκπαιδευτικών ιδρυμάτων και αυτό γιατί οι διαφορές μεταξύ των ιδρυμάτων είναι μεγάλες και κυρίως όσον αφορά τα κριτήρια ποιότητας που λαμβάνονται υπόψη για το πρόγραμμα. Η παρούσα εφαρμογή έχει γίνει με βάση τις απαιτήσεις του Πανεπιστήμιου Δυτικής Μακεδονίας. Για τον λόγο αυτό, η εφαρμογή αυτή μπορεί να θεωρηθεί ανεπαρκής για κάποιο άλλο ίδρυμα ή ακόμα και για το ίδιο το πανεπιστήμιο σε περίπτωση που πραγματοποιηθούν αλλαγές στις ανάγκες του. Παρόλα αυτά, είναι συμβατή με τα περισσότερα τμήματα που υιοθετούν παρόμοια κριτήρια. Τέλος, πρέπει να αναφέρουμε ότι υπάρχουν κάποιες βασικές κατηγορίες κανόνων, οι οποίες παραμένουν κοινές σε όλα τα ιδρύματα, όπως το παράδειγμα του αυστηρού περιορισμού που αναφέραμε παραπάνω.

## *1.1 Ιστορική αναδρομή*

Το πρόβλημα του χρονικού προγραμματισμού έχει εμφανίσει ιδιαίτερο ενδιαφέρον από το 1950 [1]. Πολλοί ερευνητές έχουν ασχοληθεί με το πρόβλημα με διαφορετικούς τρόπους. Το πρόβλημα του χρονοπρογραμματισμού παρουσιάζει ιδιαίτερο ενδιαφέρον μέχρι και σήμερα τόσο λόγω της ευρείας φύσης του, καθώς περιλαμβάνει σημαντικά προβλήματα χρονικού προγραμματισμού, όπως το πρόγραμμα σχολείου, πανεπιστημίου, εξετάσεων, αθλημάτων και εργασίας ωραρίου, όσο και λόγω της μεγάλης πολυπλοκότητάς του.

Ο Bard Adym [2] στην εργασία του το 1995 παρέχει όλες τις σχετικές βιβλιογραφίες του προβλήματος του χρονοπρογραμματισμού που έχουν δημοσιευθεί από το 1960 μέχρι το 1995. Στην εργασία του αυτή φαίνεται μία σημαντική αύξηση των σχετικών με το αντικείμενο ερευνών τη δεκαετία του 60, προς το τέλος της δεκαετίας του ‘70 είναι λιγότερες, ενώ τη δεκαετία του ‘80 ξεκινά πάλι μια ανάπτυξη, η οποία αγγίζει το μέγιστο το 1995, στις 60 δημοσιεύσεις. Ας σημειωθεί ότι την χρονιά αυτή έλαβε χώρα και το 1ο International Conference on the Practice and Theory of Automated Timetabling (PATAT) [3]. Την επόμενη χρονιά ιδρύθηκε το European Association of Operational Research Societies Working Group on Automated Timetabling και μέχρι σήμερα αριθμεί πάνω 300 μέλη, σε πάνω από 60 χώρες.

Ο Welsh και ο Powell [4] παρατήρησαν την σχέση ανάμεσα στο χρωματισμό γράφου και στον χρονοπρογραμματισμό, το 1967. Η σχέση αυτή είναι σημαντική, αφού ασκεί επιρροή ακόμη και στις σύγχρονες έρευνες χρονοπρογραμματισμού. Μία μεγάλη κατηγορία αλγορίθμων έχει βασιστεί πάνω στις μεθοδολογίες αυτές.

## *1.2 Αλγόριθμος GRASP*

Οι άνθρωποι που ασχολούνται με την γραμματειακή υποστήριξη στα εκπαιδευτικά ιδρύματα, καταλήγουν στο συμπέρασμα πόσο δύσκολο είναι να λύσουν το “αιώνιο” πρόβλημα του χρονοπρογραμματισμού. Δηλαδή, τον τρόπο με τον οποίο θα εξασφαλίζεται η απρόσκοπτη και επαρκής διδασκαλία, ενόσω οι καθηγητές θα μπορούν να διαθέτουν το χρόνο τους σε ένα συγκεκριμένο σύνολο αιθουσών με προκαθορισμένο αριθμό ωρών και σπουδαστών. Οι μαθηματικοί έχουν αποδείξει ότι το πρόβλημα του χρονικού προγραμματισμού, ανήκει στην κατηγορία των NP-Complete προβλημάτων. Αν και φαίνεται να μην υπάρχει μία δεδομένη ή από τα πριν εύκολη προκαθορισμένη

λύση αυτού του είδους των προβλημάτων, το να βρεθεί μία απλή απάντηση είναι πιθανόν αδύνατον όταν τα δεδομένα εισόδου είναι τεράστια.

Το 2009, μια νέα προσέγγιση για την επίλυση του προβλήματος του εκπαιδευτικού χρονοδιαγράμματος αναπτύχθηκε από τους ερευνητές Arnaldo Vieira Moura και Rafael Augusto Scaraficci [8] στο ινστιτούτο Πληροφορικής, στο πανεπιστήμιο Campinas, στη Βραζιλία. Η μέθοδος αυτή ονομάζεται GRASP (Greedy Randomized Adaptive Search Procedure - Άπληστη Τυχαία Διαδικασία Αναζήτησης) και είναι μία ευρετική μέθοδος. Η προσέγγιση αυτή, όπως και όλες οι άλλες, δεν εγγυάται την εύρεση της καταλληλότερης απάντησης του προβλήματος με ένα γρήγορο και αποδοτικό τρόπο.

Τα εκπαιδευτικά προβλήματα χρονοδιαγράμματος περιλαμβάνουν τον σχεδιασμό ενός αριθμού “συναντήσεων” διαφορετικών πόρων χωρίς να τους επικαλύπτει, έτσι ώστε ο κατάλληλος εκπαιδευτικός να είναι διαθέσιμος για το συγκεκριμένο μάθημα και την συγκεκριμένη ώρα. Η διαδικασία αυτή, χειροκίνητα, απαιτεί εξαιρετική προσοχή και μπορεί να είναι εξαιρετικά χρονοβόρο για μεγάλα εκπαιδευτικά ιδρύματα. Επιπλέον λόγω της πολυπλοκότητας, οι σχεδιαστές δε μπορούν να επιτύχουν πάντα την βέλτιστη λύση, οδηγούμενοι συχνά σε λύσεις, οι οποίες χαρακτηρίζονται ασυνεπείς σχετικά με τις επιθυμίες των καθηγητών.

Ένα εργαλείο βελτιστοποίησης μπορεί να βοηθήσει τους σχεδιαστές, μειώνοντας τον χρόνο που απαιτείται για να κατασκευαστεί και βελτιώνοντας το αποτέλεσμα της λύσης. Το πρόβλημα του χρονοπρογραμματισμού ορίζεται από ένα σύνολο M ανεξάρτητων αιθουσών, από N καθηγητές, S μαθήματα και P χρονικές περιόδους. Το εργαλείο βελτιστοποίησης θα πρέπει να βρει χρονονοθυρίδα για κάθε καθηγητή, βεβαιώνοντας ταυτόχρονα ότι θα δύνεται η δυνατότητα σε όλους τους σπουδαστές να παρακολουθούν τα εβδομαδιαία μαθήματα, ενώ θα πρέπει ταυτόχρονα να ικανοποιείται ένα σύνθετο σύνολο περιορισμών, όπως οι επιθυμίες των καθηγητών και η διαθεσιμότητά τους και η επαρκής ισορροπία στο ημερήσιο πρόγραμμα.

Το πρόβλημα προκύπτει επειδή ορισμένοι καθηγητές συχνά διδάσκουν πολλά μαθήματα σε διαφορετικές αίθουσες, οι μαθητές γενικά δεν πρέπει να λαμβάνουν περισσότερες από 3-4 διαλέξεις του ίδιου μαθήματος την ίδια μέρα και θα πρέπει να υπάρχουν διαλείμματα τόσο για τους σπουδαστές όσο και για τους καθηγητές. Σε αυτά και σε κάποια άλλα κριτήρια θα πρέπει να ανατεθεί ένας βαθμός σημαντικότητας, έτσι ώστε με την βοήθεια τους να εφαρμοστεί ο αλγόριθμος.

Ο αλγόριθμος GRASP επαναλαμβάνει ένα βασικό κύκλο τριών βημάτων. Πρώτον επιλέγει τυχαία ένα μάθημα και ορίζει τους πόρους του, στην συνέχεια προσθέτει το δεύτερο μάθημα και το ταξινομεί με σε σχέση με το πρώτο, μετά προσθέτει το τρίτο μάθημα κ.ο.κ. Η αναπτυσσόμενη λίστα μαθημάτων ταξινομεί τα μαθήματα που έχουν μεγαλύτερο “βάρος” με την έννοια ότι διαφορετικά κριτήρια λαμβάνονται υπόψη

κάθε φορά. Αυτό το στάδιο ονομάζεται "greedy" (άπληστο). Η πρόσθεση κάθε επόμενου μαθήματος θα πρέπει να προσαρμόζεται στη λίστα και να ταξινομείται ανάμεσα στα άλλα μαθήματα. Το δεύτερο στάδιο βελτιώνει την λίστα κατάταξης, χρησιμοποιώντας μια τοπική διαδικασία αναζήτησης, η οποία συγκρίνει γειτονικά μαθήματα και τα ταξινομεί ανάλογα. Αυτό το στάδιο συνεχίζεται μέχρι να μην υπάρχουν άλλες βελτιώσεις. Τέλος, μια στρατηγική χρησιμοποιείται για να καταδείξει τις βέλτιστες λύσεις, οι οποίες μετά χρησιμοποιούνται σαν “οδηγοί” στην τελική λύση. Ο βασικός αυτός κύκλος επαναλαμβάνεται, έως ότου τοποθετηθούν όλα τα μαθήματα στη λίστα.

Η ερευνητική ομάδα δοκίμασε επιτυχώς τη μέθοδο αυτή σε τρία λύκεια της Βραζιλίας. Ο αλγόριθμος αυτός μπορεί εύκολα να προσαρμοστεί σε οποιοδήποτε ίδρυμα.

## *1.3 Δομή εργασίας*

Στο κεφάλαιο 2, αναλύεται η πολυπλοκότητα του προβλήματος χρονοπρογραμματισμού και αναφέρονται επιγραμματικά τα είδη των περιορισμών. Επιπλέον, περιγράφονται συνοπτικά 2 βασικά προβλήματα περιορισμών και 6 βασικοί αλγόριθμοι αναζήτησης. Στο κεφάλαιο 3, περιγράφονται τα δεδομένα και οι γενικοί περιορισμοί που πρέπει να τηρούνται σε κάθε εκπαιδευτικό ίδρυμα, για την ορθή σχεδίαση ενός ωρολογίου προγράμματος. Στο κεφάλαιο 4, αναλύονται διεξοδικά τα δεδομένα και οι μεταβλητές που χρησιμοποιήθηκαν στην συγκεκριμένη εφαρμογή. Επίσης, γίνεται περιγραφή της εφαρμογής με χρήση διαγραμμάτων ροής. Στο κεφάλαιο 5, καταγράφονται τα πειραματικά αποτελέσματα της εφαρμογής και οι σχετικές παρατηρήσεις πάνω σε αυτά. Τέλος, στο κεφάλαιο 6, αναλύονται οι πειραματικές μετρήσεις, σε πλήθος εισόδων με διαφορετικές απαιτήσεις και προτείνονται κάποιες μελλοντικές επεκτάσεις της εφαρμογής.

# *ΚΕΦΑΛΑΙΟ 2*

## *ΧΡΟΝΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ*

Η έννοια του χρονικού προγραμματισμού περιγράφει την ανάγκη για την σχεδίαση ενός ωρολογίου προγράμματος. Το 1996, ο Wren [9] περιγράφει το πρόβλημα του χρονικού προγραμματισμού, ως το πρόβλημα της αξιοποίησης συγκεκριμένων πόρων, λαμβάνοντας υπόψη την ύπαρξη κάποιων περιορισμών, σύμφωνα πάντα με περιορισμένο αριθμό χρονικών διαστημάτων, με σκοπό την ικανοποίηση αντικειμενικών στόχων στον υψηλότερο δυνατό βαθμό. Τα προβλήματα του χρονικού προγραμματισμού εμφανίζουν υψηλή ποικιλομορφία και περικλείουν πολλούς και διαφορετικούς τομείς της ζωής μας. Ο ρόλος που διαδραματίζει ο ορθός σχεδιασμός του ωρολογίου προγράμματος (timetable) συχνά δε γίνεται αντιληπτός, ή τουλάχιστον δεν είναι πάντα κατανοητός ο βαθμός, στον οποίο ένα ωρολόγιο πρόγραμμα καθορίζει την καθημερινότητα μας. Τον υψηλό βαθμό σημαντικότητας των ωρολογίων προγραμμάτων μπορούμε εύκολα να τον συμπεράνουμε, αν αναλογιστούμε την αξία τους και τη

συμβολή στην ομαλή και αποδοτική λειτουργία και οργάνωση τομέων όπως η εκπαίδευση (ωρολόγια προγράμματα σχολείων-πανεπιστημίων), η υγεία (ωρολόγια προγράμματα νοσηλευτών και χειρουργείων), οι μεταφορές (προγράμματα τρένων, λεωφορείων κα.), αλλά και ο αθλητισμός (προγράμματα αγώνων).

Στην πλειονότητά τους τα προβλήματα του χρονικού προγραμματισμού είναι πολύπλοκα και συνδυαστικά. Η πολυπλοκότητα αυτή καθιστά πολύ δύσκολη, έως και αδύνατη, την αποδοτική και βέλτιστη επίλυσή τους. Το γεγονός αυτό ανάγει την επίλυση τέτοιων προβλημάτων σε μία πολύ δύσκολη, χρονοβόρα, επίπονη και συχνά ακριβή διαδικασία. Για το λόγο αυτό, επιχειρούνται διαφορετικές διατυπώσεις του προβλήματος, ώστε να γίνει εφικτή η επίλυσή του. Σε κάποιες περιπτώσεις, η λύση ενός προβλήματος χρονικού προγραμματισμού ουσιαστικά επικεντρώνεται, απλώς και μόνο στην εύρεση ενός ωρολογίου προγράμματος που να ικανοποιεί όλους τους περιορισμούς. Σε αυτές τις περιπτώσεις το πρόβλημα διατυπώνεται ως πρόβλημα αναζήτησης (search problem). Σε κάποιες άλλες όμως περιπτώσεις το πρόβλημα μας διατυπώνεται ως πρόβλημα βελτιστοποίησης (optimization problem). Η απαίτηση που διατυπώνεται στο πρόβλημα βελτιστοποίησης είναι η εύρεση ενός ωρολογίου προγράμματος, που να ικανοποιεί όλους τους "σκληρούς" περιορισμούς (hard constraints), και να ελαχιστοποιεί (ή να μεγιστοποιεί) μία αντικειμενική συνάρτηση που εμπεριέχει τους "μαλακούς" περιορισμούς (soft constraints). Η ανάγκη να τηρηθούν οι "σκληροί" περιορισμοί είναι μεγαλύτερη, ενώ οι "μαλακοί" είναι πιο ελαστικοί και δεύτερης προτεραιότητας.

Επιπλέον, έχει εκφραστεί και μία εναλλακτική προσέγγιση, η οποία σύμφωνα με τους υποστηρικτές της, θεωρείται ότι η διατύπωση ενός προβλήματος βελτιστοποίησης στην πραγματικότητα, δεν είναι τίποτα άλλο, παρά η εφαρμογή τεχνικών βελτιστοποίησης σε ένα πρόβλημα αναζήτησης. Επίσης, υποστηρίζουν ότι η αντικειμενική συνάρτηση στοχεύει στη μείωση του αριθμού των μη εφικτών λύσεων στο πρόβλημα, οπότε πρακτικά, αναπαριστά την απόσταση μίας λύσης από την εφικτή λύση του πραγματικού προβλήματος. Και στις δύο περιπτώσεις (πρόβλημα αναζήτησης ή βελτιστοποίησης) ορίζουμε το θεμελιώδες πρόβλημα που δεν είναι άλλο από το πρόβλημα της επιλογής της λύσης. Στα προβλήματα αναζήτησης η επιλογή γίνεται με βάση το εάν υπάρχει μία λύση, ενώ στα προβλήματα βελτιστοποίησης το πρόβλημα της επιλογής έγκειται στο εάν υπάρχει μία λύση που να αποδίδει δεδομένη αξία-τιμή στην αντικειμενική συνάρτηση.

# *2.1 Πολυπλοκότητα χρονοπρογραμματισμού*

Ήδη έχουμε αναφερθεί στην πολυπλοκότητα των προβλημάτων χρονικού προγραμματισμού. Στην ουσία όμως, αναφερόμαστε στην πολυπλοκότητα του θεμελιώδους προβλήματος της επιλογής. Το θεμελιώδες πρόβλημα ανήκει και αυτό στη κατηγορία των NP-Complete προβλημάτων σχεδόν για όλες τις παραλλαγές του προβλήματος. Για το λόγο αυτό, μία ακριβής λύση επιτυγχάνεται μόνο σε μικρού μεγέθους προβλήματα (π.χ. λιγότερα από 10 μαθήματα, για ένα ωρολόγιο πρόγραμμα πανεπιστημίου), αν και στην πραγματικότητα συνήθως εμπλέκονται μερικές εκατοντάδες μαθημάτων. Έτσι, είναι επόμενο πως μόνο ευρετικές μέθοδοι (heuristic methods) είναι εφικτές και ενδείκνυται να εφαρμοστούν στη διαδικασία επίλυσης του προβλήματος, αν και στην πράξη, ούτε οι ευρετικές μέθοδοι εγγυώνται την εξεύρεση της βέλτιστης λύσης.

Όσον αφορά την κατασκευή ενός ωρολογίου προγράμματος για την εκπαίδευση, προσπαθούμε ένα δεδομένο σύνολο από συναντήσεις μεταξύ μαθητών και καθηγητών να το τοποθετήσουμε μέσα στο χρόνο, έτσι ώστε το πρόγραμμα να είναι εφικτό και αποδεκτό από όλους τους εμπλεκόμενους.

Στην κοινότητα της τεχνητής νοημοσύνης, η ικανοποίηση των προβλημάτων με περιορισμούς για πεπερασμένα και μη πεπερασμένα πεδία έχουν μελετηθεί με τον όρο “Προβλήματα ικανοποίησης περιορισμών”.

**Προβλήματα ικανοποίησης περιορισμών**

Ένα πρόβλημα ικανοποίησης περιορισμών [10], αποτελείται από ένα σύνολο περιορισμών C, το οποίο εφαρμόζεται πάνω σε ένα σύνολο μεταβλητών $x_1,x_2,\ldots,x_n$, και από ένα σύνολο πεδίων D, το οποίο απεικονίζει κάθε μεταβλητή $x_i$ σε ένα σύνολο τιμών $D(x_i)$, τις οποίες επιτρέπεται να παίρνει.

Κάθε κατάσταση του προβλήματος καθορίζεται από την μερική ή ολική ανάθεση τιμών στις μεταβλητές. Κάθε περιορισμός του συνόλου C εμπλέκει ένα υποσύνολο μεταβλητών και καθορίζει τους επιτρεπόμενους συνδυασμούς τιμών για το υποσύνολο αυτό.

Μία λύση του προβλήματος αποτελείται από την πλήρη ανάθεση τιμών στις μεταβλητές όταν ικανοποιούνται όλοι οι περιορισμοί, δηλαδή είναι κατανοητό ότι το πρόβλημα ικανοποίησης αναπαριστά τον περιορισμό:

$$C \wedge x_1 \wedge D(x_1) \wedge \ldots \wedge x_n \wedge D(x_n)$$

## *2.2 Προβλήματα Ικανοποίησης Περιορισμών με πεπερασμένα πεδία*

Το απλούστερο είδος προβλημάτων ικανοποίησης περιορισμών είναι εκείνα που περιέχουν διακριτές μεταβλητές με πεπερασμένα πεδία τιμών. Χαρακτηριστικά παραδείγματα, το παράδειγμα του χρωματισμού χάρτη και εκείνο των Ν βασιλισσών [11]. Στην ομάδα των προβλημάτων ικανοποίησης περιορισμών με πεπερασμένα πεδία ανήκουν και τα προβλήματα εκείνα των οποίων οι μεταβλητές μπορούν να πάρουν λογικές τιμές true ή false. Όπως είπαμε κάθε λύση του προβλήματος αποτελεί πλήρη ανάθεση τιμών σε όλες τις μεταβλητές. Εάν τώρα ο αριθμός των μεταβλητών είναι ίσως με n τότε κάθε λύση θα εμφανίζεται σε βάθος n του δέντρου αναζήτησης, γεγονός, το οποίο κάνει τους αλγόριθμους αναζήτησης κατά βάθος πολύ δημοφιλείς. Στην περίπτωση όπου το μέγιστο μέγεθος πεδίου οποιασδήποτε μεταβλητής του προβλήματος είναι d, τότε ο αριθμός όλων των πιθανών πλήρως αναθέσεων τιμών είναι O(dn), το οποίο είναι εκθετικό ως προς τον αριθμό των μεταβλητών. Αυτό έχει ως αποτέλεσμα όσο αυξάνεται ο αριθμός των μεταβλητών του προβλήματος τόσο να αυξάνεται εκθετικά και ο χώρος αναζήτησης. Στην χειρότερη περίπτωση μάλιστα δε μπορούμε να περιμένουμε ότι ο χρόνος επίλυσης μπορεί να είναι μικρότερος του εκθετικού.

**Πρόβλημα χρωματισμού χάρτη**

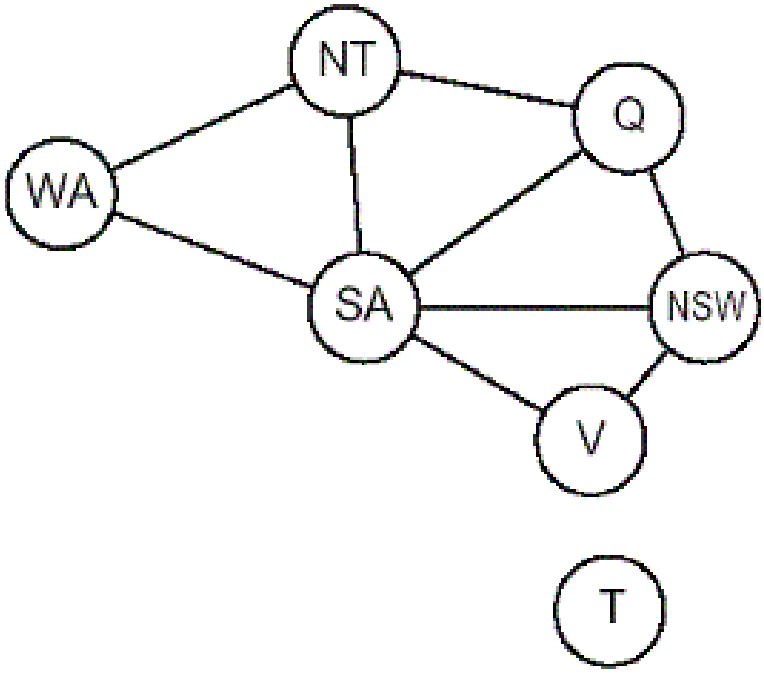
Το πρόβλημα του χρωματισμού χάρτη (map coloring problem) αποτελεί ένα από τα αρχέτυπα προβλήματα ικανοποίησης περιορισμών. Το πρόβλημα αποτελείται από τον χρωματισμό διαφορετικών περιοχών ενός προκαθορισμένου χάρτη, με περιορισμένο αριθμό χρωμάτων και υπόκειται στην περιοριστική συνθήκη ότι δύο παρακείμενες περιοχές δεν επιτρέπεται να έχουν το ίδιο χρώμα. Για παράδειγμα, ας θεωρήσουμε τον χάρτη της Αυστραλίας του σχήματος 1 και ότι έχουμε τα χρώματα κόκκινο, κίτρινο και μπλε.


**Σχήμα 1:** Το πρόβλημα του χρωματιστού χάρτη

Για κάθε περιοχή ορίζεται μία μεταβλητή WA,NT,SA,Q,NSW,V και T, στην οποία πρέπει να γίνει ανάθεση χρώματος. Κάθε μεταβλητή έχει ως πεδίο το σύνολο τιμών {κόκκινο, κίτρινο, μπλε}. Ο παρακάτω περιορισμός εγγυάται ότι κάθε ζεύγος παρακείμενων περιοχών δε μπορεί να έχει το ίδιο χρώμα:

$$WA \neq N \wedge WA \neq SA \wedge NT \neq SA \wedge NT \neq Q \wedge SA \neq Q \wedge SA \neq NSW \wedge SA \neq V \wedge Q \neq NSW \wedge NSW \neq V$$

Μία από τις μεθόδους που έχουν αναπτυχθεί για την αναπαράσταση των περιορισμών του προβλήματος, είναι εκείνη με γράφο. Κάθε κόμβος του γράφου αντιστοιχεί σε μία μεταβλητή του προβλήματος, ενώ κάθε ακμή αντιστοιχεί σε περιορισμό. Στο Σχήμα 2 αναπαριστάται το πρόβλημα του χρωματισμού χάρτη με γράφο. Κάθε περιοχή του χάρτη αντικαθίσταται από κάθε κορυφή της γραφικής παράστασης και δύο κορυφές συνδέονται εάν και μόνο αν οι δύο περιοχές μοιράζονται ένα τμήμα συνόρων. Στο πρόβλημα χρωματιστού χάρτη (Σχήμα 2), η Δυτική Αυστραλία (Western Australia – WA) μοιράζεται ένα τμήμα των συνόρων της με την Νότια Αυτσραλία (South Australia – SA), οπότε και συνδέονται. Αντίθετα, η Τασμανία (Tasmania – T) δε συνδέεται με καμία κορυφή του γράφου.




**Σχήμα 2:** Αναπαράσταση περιορισμών με γράφο

### Πρόβλημα Ν βασιλισσών

Ένα ακόμη γνωστό πρόβλημα ικανοποίησης περιορισμών είναι το πρόβλημα των Ν βασιλισσών (N-queens). Στο πρόβλημα αυτό πρέπει να τοποθετηθούν Ν βασίλισσες πάνω σε μία σκακιέρα μεγέθους Ν×Ν, έτσι ώστε όλες οι βασίλισσες να είναι σε διαφορετικές γραμμές, στήλες και διαγώνιους. Ας θεωρήσουμε το πρόβλημα των τεσσάρων βασιλισσών.

Μπορούμε να τυποποιήσουμε το πρόβλημα ως πρόβλημα ικανοποίησης περιορισμών θεωρώντας ότι κάθε βασίλισσα i έχει δύο μεταβλητές, την μεταβλητή Ri και Ci, οι οποίες αντιστοιχούν στην ανάλογη γραμμή και στήλη που είναι τοποθετημένη η βασίλισσα πάνω στην σκακιέρα. Το πεδίο τιμών της κάθε μεταβλητής είναι {1,2,3,4}. Ο περιορισμός:

$$R_1 \neq R_2 \wedge R_1 \neq R_3 \wedge R_1 \neq R_4 \wedge R_2 \neq R_3 \wedge R_2 \neq R_4 \wedge R_3 \neq R_4,$$

εξασφαλίζει ότι δύο βασίλισσες δε μπορούν να βρίσκονται στην ίδια γραμμή, ενώ ο περιορισμός

$$C_1 \neq C_2 \wedge C_1 \neq C_3 \wedge C_1 \neq C_4 \wedge C_2 \neq C_3 \wedge C_2 \neq C_4 \wedge C_3 \neq C_4,$$

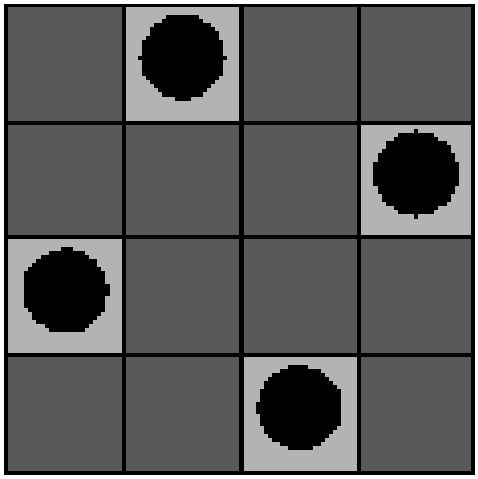
εξασφαλίζει ότι δύο βασίλισσες δε μπορούν να βρίσκονται στην ίδια στήλη, και τέλος οι περιορισμοί

$$C_1 - R_1 \neq C_2 - R_2 \wedge C_1 - R_1 \neq C_3 - R_3 \wedge C_1 - R_1 \neq C_4 - R_4 \wedge C_2 - R_2 \neq C_3 - R_3 \wedge C_2 - R_2 \neq C_4 - R_4 \wedge C_3 - R_3 \neq C_4 - R_4$$

και

$$C_1 + R_1 \neq C_2 + R_2 \wedge C_1 + R_1 \neq C_3 + R_3 \wedge C_1 + R_1 \neq C_4 + R_4 \wedge C_2 \neq C_3 + R_3 \wedge C_2 + R_2 \neq C_4 + R_4 \wedge C_3 + R_3 \neq C_4 + R_4$$

εξασφαλίζουν ότι δύο βασίλισσες δε μπορούν να βρίσκονται στην ίδια διαγώνιο. Μία λύση στο παραπάνω πρόβλημα είναι αυτή του Σχήματος 3.



**Σχήμα 3:** Το πρόβλημα των 4ων βασιλισσών

## *2.3 Προβλήματα Ικανοποίησης Περιορισμών με μη πεπερασμένα πεδία*

Οι διακριτές μεταβλητές μπορούν επίσης, να έχουν μη πεπερασμένα πεδία, για παράδειγμα το σύνολο των ακέραιων αριθμών ή το σύνολο των αλφαριθμητικών. Ας θεωρήσουμε ότι έχουμε δύο εργασίες J1 και J2 [11] τις οποίες πρέπει να αναθέσουμε στο χρόνο. Εάν τώρα θεωρήσουμε ότι κάθε εργασία Ji έχει μία μεταβλητή StartJi, η οποία έχει ως πεδίο τιμών το σύνολο των ακεραίων, κάθε ένας από τους οποίους αντιστοιχεί και σε μία ημέρα, τότε για τις μεταβλητές αυτές με μη πεπερασμένα πεδία είναι αδύνατων να εκφράσουμε περιορισμούς, που να απαριθμούν όλους τους επιτρεπόμενους συνδυασμούς τιμών. Στην περίπτωση αυτή απαιτείται μία γλώσσα, που να μπορεί να εκφράσει περιορισμούς. Εάν για παράδειγμα η εργασία J1 απαιτεί 5 ημέρες για να διεκπεραιωθεί και ισχύει ότι πρέπει να προηγείται της εργασίας J2, τότε χρειαζόμαστε μία γλώσσα περιορισμών με αλγεβρικές ανισότητες όπως StartJ1 + 5 ≤ StartJ2. Είναι πολύ απίθανο να λυθούν τέτοιου είδους περιορισμοί απαριθμώντας όλες τις πιθανές αναθέσεις τιμών, καθώς υπάρχει άπειρος συνδυασμός από αυτές. Ειδικοί αλγόριθμοι αναζήτησης λύσης έχουν αναπτυχθεί για γραμμικούς περιορισμούς πάνω σε μεταβλητές ακεραίων πεδίων.

Οι γραμμικοί περιορισμοί είναι περιορισμοί, στους οποίους οι μεταβλητές εμφανίζονται μόνο σε γραμμική μορφή, για παράδειγμα X + Y = 1. Μπορεί να αποδειχθεί ότι δεν υπάρχει αλγόριθμος, ο οποίος μπορεί να λύσει μη γραμμικούς περιορισμούς για μεταβλητές ακέραιων πεδίων. Σε κάποιες περιπτώσεις, μπορούμε να μετατρέψουμε ένα πρόβλημα ικανοποίησης περιορισμών με ακέραια πεδία σε πρόβλημα πεπερασμένων πεδίων οριοθετώντας τις τιμές όλων των μεταβλητών. Για παράδειγμα, σε ένα πρόβλημα χρονικού προγραμματισμού διεργασιών μπορούμε να θέσουμε ένα άνω όριο, το οποίο θα ισούται με το άθροισμα της διάρκειας όλων των διεργασιών.

## *2.4 Είδη περιορισμών*

Σε ένα πρόβλημα ικανοποίησης περιορισμών, εξετάζοντας τον τύπο των μεταβλητών που μπορούν να υπάρξουν, είναι χρήσιμο να εξετάσουμε και το είδος των περιορισμών

*Μοναδιαίος Περιορισμός*

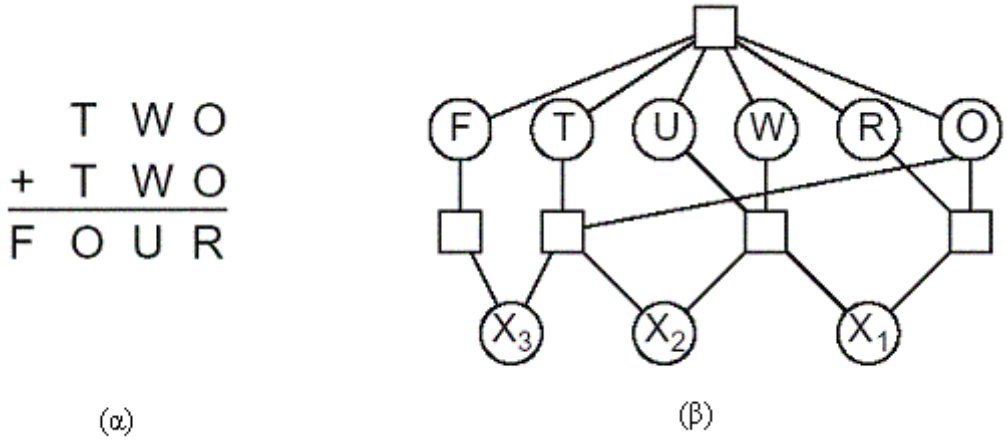
Ο απλούστερος τύπος περιορισμού που υπάρχει είναι ο μοναδιαίος περιορισμός, ο οποίος περιορίζει την τιμή μίας μοναδικής μεταβλητής. Για παράδειγμα, στο παράδειγμα του χρωματισμού γράφου μπορεί η μεταβλητή WA να μην επιτρέπεται να έχει το χρώμα κόκκινο. Κάθε μοναδιαίος περιορισμός μπορεί να εξαλειφθεί απλά ελέγχοντας το πεδίο τιμών την αντίστοιχης μεταβλητής και αφαιρώντας όποια τιμή παραβιάζει τον περιορισμό.

*Δυαδικός Περιορισμός*

Ένας δυαδικός περιορισμός συσχετίζει τις τιμές δύο μεταβλητών, για παράδειγμα $X \neq Y$ αποτελεί ένα δυαδικό περιορισμό στον οποίο οι μεταβλητές X και Y δε μπορούν να έχουν τις ίδιες τιμές. Οι αναπαράσταση δυαδικών περιορισμών μπορεί να γίνει με γράφο.

*Υψηλού βαθμού Περιορισμός*

Οι Υψηλού βαθμού περιορισμοί συσχετίζουν τις τιμές τριών και περισσοτέρων μεταβλητών. Ένα παράδειγμα είναι αυτό του κρυπταριθμητικού αινίγματος του σχήματος 4.




**Σχήμα 4:** Το πρόβλημα του κρυπταριθμητικού αινίγματος (α) και αναπαράστασή του με γράφο (β)

Στο πρόβλημα αυτό κάθε γράμμα αντιστοιχεί και σε ένα διαφορετικό ψηφίο. Το παραπάνω παράδειγμα μπορεί να παρασταθεί με έξη μεταβλητές οι οποίες περιορίζονται από τον περιορισμό alldiff(F,T,U,W,R,O), ο οποίος διασφαλίζει ότι κάθε γράμμα θα αντιστοιχεί και σε ένα διαφορετικό ψηφίο

*Αυστηροί και Εύκαμπτοι* περιορισμοί

Οι περιορισμοί που έχουμε περιγράψει μέχρι τώρα αποτελούν *αυστηρούς* [12] περιορισμούς η παραβίαση των οποίων αποκλείει κάθε εν δυνάμει λύση. Υπάρχουν, όμως και περιπτώσεις πραγματικών εφαρμογών με προβλήματα *ικανοποίησης περιορισμών*, οι οποίες μπορεί να περιέχουν περιορισμούς βάση των οποίων μία λύση μπορεί να θεωρηθεί καταλληλότερη σε σχέση με μία άλλη. Οι περιορισμοί αυτοί ονομάζονται *εύκαμπτοι* περιορισμοί και χρησιμοποιούνται κυρίως σε προβλήματα βελτιστοποίησης ή και τοπικής αναζήτησης.

Οι εύκαμπτοι περιορισμοί μπορούν να εκφραστούν χρησιμοποιώντας κάποια βάρη σε σχέση με τους αντίστοιχους περιορισμούς. Κάθε φορά που κάποιος περιορισμός παραβιάζεται, η αντίστοιχη τιμή βάρους προστίθεται στη συνάρτηση κόστους, βάση της οποίας επιλέγεται ή απορρίπτεται αντίστοιχα μία λύση. Για παράδειγμα, ας θεωρήσουμε το πρόβλημα κατάρτισης ωρολογίου προγράμματος, στο οποίο μία διάλεξη παρακολουθείται από μεγάλο αριθμών φοιτητών και επιθυμούμε να την αναθέσουμε σε μία αίθουσα. Είναι λογικό ότι η αίθουσα την οποία θα επιλέξουμε, θα πρέπει να είναι μεγάλη παρά μικρή σε χωρητικότητα, αν και η ανάθεση της διάλεξης σε μικρή αίθουσα αποτελεί και αυτή λύση, αλλά όχι την βέλτιστη. Έτσι, θα μπορούσαμε να πούμε ότι η ανάθεση της διάλεξης σε μεγάλη αίθουσα θα είχε κόστος 0, ενώ η

ανάθεση της διάλεξης σε μικρή αίθουσα θα είχε κόστος 1, προσδιορίζοντας με αυτόν τον τρόπο την βέλτιστη λύση.

## *2.5 Αλγόριθμοι αναζήτησης*

Κάθε αλγόριθμος αναζήτησης για την εύρεση ενός αντικειμένου ακολουθεί κάποιες συγκεκριμένες ιδιότητες και μεθοδολογίες. Σ' αυτή την παράγραφο περιγράφονται συνοπτικά 6 βασικοί αλγόριθμοι αναζήτησης.

### **1.** Αλγόριθμος "Παρήγαγε κι εξέτασε" (generate and test)

Ο αλγόριθμος αναζήτησης "Παρήγαγε κι εξέτασε" [13] [14] είναι η πιο γενική μέθοδος, δίνει από μία τιμή σε όλες τις μεταβλητές και στην συνέχεια ελέγχει αν είναι η λύση. Βασικό της μειονέκτημα είναι η ύπαρξη τυφλής γεννήτριας αναθέσεις τιμών και η ανακάλυψη ασυνεπειών είναι αργή. Παρ' όλα αυτά μπορεί να βελτιωθεί προσθέτοντας μία έξυπνη γεννήτρια, η οποία κάνει τοπική αναζήτηση, ελέγχοντας έτσι τους περιορισμούς κατά την διάρκεια της ανάθεσης.

Ας δούμε ένα παράδειγμα, υποθέτουμε ότι ένας εργαζόμενος πρέπει να πραγματοποιήσει μία σειρά από δραστηριότητες α, β, γ, δ και ε, οι οποίες μπορούν να συμβούν, οποιαδήποτε χρονική στιγμή από τις 1, 2, 3 ή 4. Έστω ότι Α είναι η χρονική στιγμή, που η δραστηριότητα α θα συμβεί, Β η χρονική στιγμή που η δραστηριότητα β θα συμβεί και αντίστοιχα Γ, Δ και Ε οι χρονικές στιγμές, που θα συμβούν οι γ, δ και ε. Οι πιθανές τιμές, δηλαδή το πεδίο ορισμού, των μεταβλητών Α, Β, Γ, Δ και Ε είναι:

Α={1,2,3,4}, Β={1,2,3,4}, Γ={1,2,3,4}, Δ={1,2,3,4}, Ε={1,2,3,4}

Έστω, ότι πρέπει να ικανοποιούνται οι εξής περιορισμοί:

{(Β≠3), (Γ≠2), (Α≠Β), ( Β≠Γ), (Γ<Δ), (Α=Δ),

(Ε<Α), (Ε<Β), (Ε<Γ), (Ε<Δ), (Β≠Δ)}

Στόχος είναι να βρούμε τις τιμές των μεταβλητών Α, Β, Γ, Δ και Ε ώστε να να πληρούνται όλοι οι περιορισμοί. Οι πιθανοί συνδυασμοί (χωρίς τους περιορισμούς) είναι 45 = 1024:

Q={ {Α=1,Β=1,Γ=1,Δ=1,Ε=1},

{Α=1,Β=1,Γ=1,Δ=1,Ε=2},...,

{Α=4,Β=4,Γ=4,Δ=4,Ε=4} }

Ο αλγόριθμος "Παρήγαγε κι εξέτασε" εξετάζει τον κάθε συνδυασμό ξεχωριστά, ελέγχοντας αν πληροί όλους τους περιορισμούς και ανάλογα τον δέχεται ή τον απορρίπτει. Επομένως, ο συνολικός αριθμός συγκρίσεων είναι 45 * 11=11.264.

Αν οι δραστηριότητες είναι n και οι περιορισμοί e, τότε ο αριθμός συγκρίσεων, δηλαδή η πολυπλοκότητα του αλγορίθμου, είναι O(edn). Αν το n γίνει πολύ μεγάλο, το πρόβλημα γίνεται πολύ γρήγορα ακανθώδες και πρέπει να βρούμε εναλλακτικές μεθόδους λύσης.

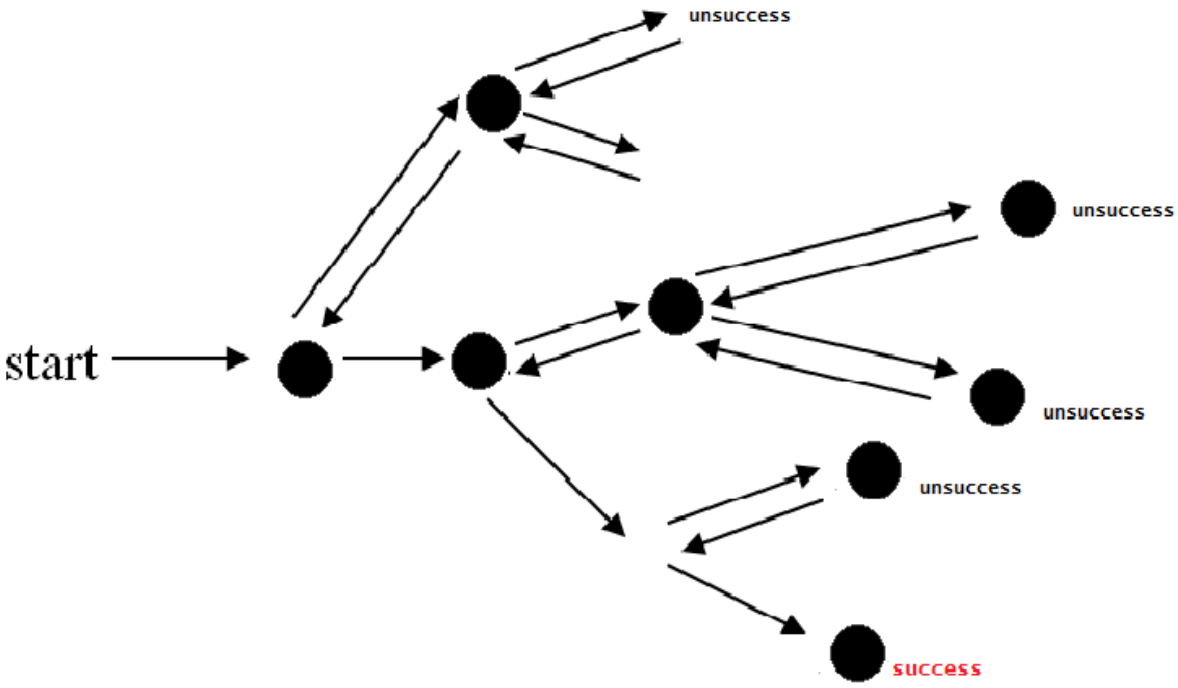
## **2.** Αλγόριθμος οπισθοδρόμησης (backtracking search)

Βασικό χαρακτηριστικό αυτής της μεθόδου [10] είναι η αναίρεση των αποτελεσμάτων κάποιων υπολογιστικών βημάτων και η οπισθοδρόμηση σε κάποιο προηγούμενο βήμα, όπου λαμβάνονται κάποιες διαφορετικές επιλογές. Η οπισθοδρόμηση δεν πρέπει να εμποδίζει τον αλγόριθμο από το να τερματίζει και δεν πρέπει να ξοδεύει χρόνο σε περιττές επαναλήψεις.

Διάφορα προβλήματα μπορούν να θεωρηθούν ως γράφοι των οποίων οι κόμβοι είναι καταστάσεις του προβλήματος και η ύπαρξη ακμής μεταξύ δύο κόμβων δηλώνει τη δυνατότητα κίνησης μεταξύ των καταστάσεων που απεικονίζουν. Η λύση του προβλήματος μπορεί να μεταφραστεί ως την εύρεση κάποιου κόμβου, ή μονοπατιού μέσα στο γράφο, άρα μπορεί να επιτευχθεί με μια κατά βάθος διερεύνηση του γράφου:

α. Αρχικά ξεκινάμε, χωρίς καμιά γνώση για τη λύση του προβλήματος, από τη 'ρίζα' του γράφου, όπως φαίνεται και στο Σχήμα 5. Κατά την αναζήτηση του γράφου χτίζουμε τη λύση ως εξής: επισκεπτόμενοι κάποιο κόμβο μελετούμε τις πληροφορίες που περιέχει σχετικά με τη λύση του προβλήματος. Αν αυτές είναι νόμιμες με τη λύση τότε τις συλλέγουμε και συνεχίζουμε τη διερεύνηση. Διαφορετικά, τις αγνοούμε και οπισθοδρομούμε στον προηγούμενο κόμβο από όπου συνεχίζουμε την κατά-βάθος διερεύνηση ακολουθώντας κάποια διαφορετική επιλογή.

β. Κτίσιμο του γράφου μπορεί να γίνει δυναμικά, κατά την αναζήτηση της λύσης του προβλήματος.




**Σχήμα 5:** Γράφημα αλγορίθμου οπισθοδρόμησης

## 3. Αλγόριθμος τοπικής αναζήτησης (*local search)*

Οι αλγόριθμοι τοπικής αναζήτησης χρησιμοποιούν την τεχνική της επαναληπτικής βελτίωσης και λειτουργούν ως εξής:

α) Διαλέγουμε μία λύση από το χώρο αναζήτησης και την

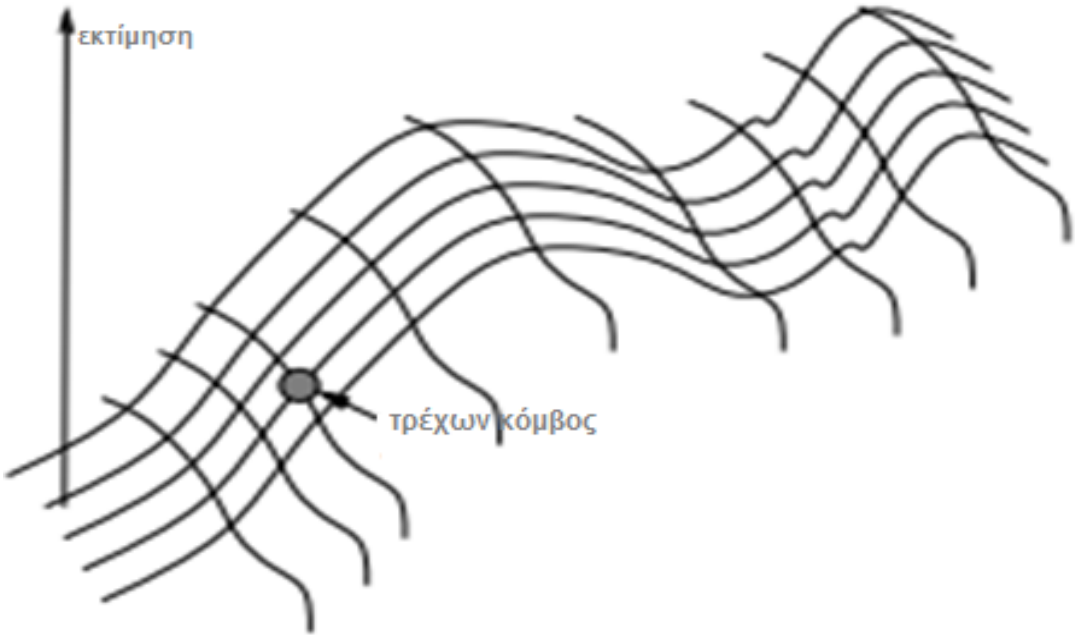
αποτιμούμε. Ονομάζουμε αυτή τη λύση τρέχουσα.

β) Εφαρμόζουμε ένα μετασχηματισμό στην τρέχουσα λύση, για να παράγουμε μία νέα λύση και την αξιολογούμε.

γ) Αν η νέα λύση είναι καλύτερη από την τρέχουσα, τότε ανταλάσσουμε με την τρέχουσα λύση, διαφορετικά απορρίπτουμε την νέα λύση.

δ) Επαναλαμβάνουμε τα παραπάνω βήματα μέχρι κανένας μετασχηματισμός να μην βελτιώνει άλλο την τρέχουσα λύση.

Βασική ιδέα του αλγορίθμου τοπικής αναζήτησης είναι ότι ξεκινάμε με μία λύση και κάνουμε μετασχηματισμούς μέχρι να βρούμε μία λύση, όπως φαίνεται και γραφικά στην Εικόνα 1.




**Εικόνα 1:** Γραφική παράσταση αλγορίθμου τοπικής αναζήτησης

## 4. Αλγόριθμος διάδοσης περιορισμών (constraint propagation)

Κάποιες φορές ο χώρος αναζήτησης μπορεί να μειωθεί δραστικά, εξετάζοντας τις συνέπειες που επιφέρει η μερική ανάθεση τιμών στις μεταβλητές όπου δεν τους έχει γίνει ανάθεση. Μπορούμε να μειώσουμε το πεδίο τιμών αυτών των μεταβλητών, μειώνοντας έτσι τον παράγοντα διακλάδωσης, διαγράφοντας τιμές, οι οποίες δεν είναι συνεπείς με τις τιμές των μεταβλητών που τους έχει γίνει ανάθεση. Ο γενικός όρος για αυτήν την διαδικασία ονομάζεται διάδοση περιορισμού [15]. Με έναν αλγόριθμο διάδοσης περιορισμών, είναι εφικτό να μοντελοποιηθεί και το Sudoku ως εξής:

Όπως σε όλα τα προβλήματα περιορισμών, ορίζουμε πρώτα το σύνολο τιμών και τους περιορισμούς κάθε μεταβλητής.

*Σύνολο μεταβλητών:* Το σύνολο των μεταβλητών αποτελείται από το σύνολο των κελιών του αρχικού Sudoku, που δεν έχουν πάρει τιμή ακόμη.

*Σύνολο τιμών:* Κάθε μεταβλητή μπορεί να πάρει τιμές από 1 έως και size*size(όπου size το μέγεθος του Sudoku).

*Περιορισμός:* Σε κάθε γραμμή, σε κάθε στήλη και σε κάθε υποτετράγωνο (όπως αυτό ορίζεται από τον ορισμό του προβλήματος του Sudoku) δεν επιτρέπεται δύο ή παραπάνω μεταβλητές να έχουν τις ίδιες τιμές.

*Λύση:* Το Sudoku λύνεται όταν όλες οι μεταβλητές έχουν πάρει τιμή και ικανοποιείται ο παραπάνω περιορισμός.

## **5.** Γενετικοί αλγόριθμοι (genetic algorithms)

Ο γενετικός αλγόριθμος είναι αλγόριθμος αναζήτησης λύσεων και επίλυσης προβλημάτων. Αρχικά εξελίσσει ένα σύνολο πιθανών λύσεων μέχρι να βρεθεί εκείνη που ικανοποιεί το ζητούμενο. Η βασική ιδέα είναι ότι μια συγκεκριμένη λύση θα μεταδώσει το περιεχόμενο της, το οποίο είναι μια πληροφορία, στους απογόνους της με την προϋπόθεση ότι αυτή είναι μια καλή λύση. Όσο πιο καλή είναι μια λύση, τόσο μεγαλώνει η πιθανότητα να μεταδώσει τη πληροφορία της, δηλαδή με άλλα λόγια επικρατεί η ισχυρότερη.

Η παρουσίαση των γενετικών αλγορίθμων έγινε από τον Friedberg το 1958, ο οποίος χρησιμοποίησε και την Fortran στην προσπάθειά του να επιλύσει κάποια προβλήματα. Το 1975 ο Holland [16] ενίσχυσε αυτή τη μέθοδο χρησιμοποιώντας σειρές bits για να αναπαραστήσει λειτουργίες, με τρόπο ώστε κάθε συνδυασμός bits να είναι μια έγκυρη λειτουργία. Στο προγραμματισμό παραγωγής η μέθοδος γενετικών αλγορίθμων εφαρμόστηκε από τον Davis μόλις το 1985, ο οποίος κατάφερε να κατασκευάσει μια προτιμημένη διαταγή των διαδικασιών για κάθε μηχανή. Νέα ώθηση στο χώρο έδωσαν οι Falkenauer και Bouffouix [17] το 1991, οι οποίοι οδήγησαν στην εισαγωγή ενός μοντέλου κωδικοποίησης των διαδικασιών που πραγματοποιούνται σε μια μηχανή κατά την λειτουργία της. Ακόμα πιο πρόσφατα, μόλις το 1995, ο Kobayashi [18] εισήγαγε μια νέα τεχνική, σύμφωνα με την οποία ένα χρωμόσωμα είναι μια σειρά από σύμβολα μήκους n και κάθε σύμβολο αναγνωρίζει μια διαδικασία που πρόκειται να ανατεθεί σε μια μηχανή. Τέλος, πιο πρόσφατα, ο Shi [19] το 1997, εφαρμόζει μια τεχνική διασταυρώσεων, που διαιρεί τυχαία έναν αυθαίρετα επιλεγμένο σύντροφο σε δύο υποσύνολα, από τα οποία παράγεται ο απόγονος.

## **6.** Ευρετικοί αλγόριθμοι (heuristics algorithms)

Οι ευρετικοί αλγόριθμοι [20] επικεντρώνονται και αυτοί στην επίλυση προβλημάτων παραγωγής. Οι συνηθισμένοι αλγόριθμοι προχωρούν στη διαδικασία επίλυσης, χωρίς να γνωρίζουν εκ των προτέρων ποια από τα μονοπάτια που ακολουθούν τους οδηγούν σε αδιέξοδο. Σε πραγματικά προβλήματα όμως, ο αριθμός των συνδυασμών που καταφθάνουν σε τερματικές καταστάσεις είναι αρκετά μεγάλος, φαινόμενο που μεταμορφώνει την επίλυση τους σε μία χρονοβόρα διαδικασία. Για αυτό ακριβώς το λόγο χρησιμοποιούνται οι αλγόριθμοι ευρετικής αναζήτησης, οι οποίοι

στοχεύουν τόσο στη μείωση του χρόνου επίλυσης όσο και στη μείωση του αριθμού καταστάσεων που εξετάζει ένας αλγόριθμος. Ο ευρετικός μηχανισμός (heuristic) είναι η τεχνική που στηρίζεται στην πληροφορία εκείνη η οποία αξιολογεί ποιο μονοπάτι δεν οδηγεί σε θεμιτό αποτέλεσμα. Με τη χρησιμοποίηση του συγκεκριμένου μηχανισμού, ο χρόνος που ξοδεύεται μπορεί να μειωθεί σημαντικά, εφόσον δίνεται το πλεονέκτημα της δυνατότητας πρόβλεψης των καταστάσεων που δεν οδηγούν πουθενά και για αυτό τον λόγο μπορούν να κλαδευτούν. Λαμβάνοντας υπόψη την αποδοτικότητα των μηχανών, τα στοιχεία πρέπει να φορτωθούν πρώτα στη μηχανή που μπορεί να τελειώσει πιο γρήγορα. Είναι επιθυμητό να αυξηθεί η χρησιμότητα της μηχανής ή να ελαχιστοποιηθεί ο συνολικός χρόνο ροής.

Η διαδικασία Shifting Bottleneck Procedure (SBP), που δημιούργησε ο Adams (21) το 1988 εμπνεύστηκε από τη θεωρία των περιορισμών, είναι η επικρατέστερη τεχνική, στην οποία βασίζονται οι ευρετικοί αλγόριθμοι. Η βασική ιδέα είναι η αναγωγή ενός προβλήματος m μηχανών σε επιμέρους m προβλήματα μιας μηχανής. Το κάθε υποπρόβλημα λύνεται μεμονωμένα και στη συνέχεια, ανάλογα με τη λύση, οι αντίστοιχες μηχανές ιεραρχούνται σύμφωνα με την επίδοσή τους. Η μηχανή με την καλύτερη δυνατή λύση χαρακτηρίζεται ως “bottleneck” και έχει μεγαλύτερη προτεραιότητα από τις άλλες. Στη συνέχεια η μηχανή αυτή θεωρείται ως μηχανή αναφοράς και επομένως όλες οι προηγούμενες εργασίες επαναπροσδιορίζονται, με τα νέα όμως δεδομένα. Η ίδια διαδικασία επαναλαμβάνεται και για τις υπόλοιπες μηχανές. Αξίζει να αναφερθεί ότι η I/O “bottleneck” διαδικασία στα παράλληλα συστήματα υπολογιστών έχει πρόσφατα ξεκινήσει να δέχεται αυξανόμενο ενδιαφέρον και το μεγαλύτερο ποσοστό της προσοχής επικεντρώνεται στη βελτίωση της απόδοσης των I/O συσκευών, χρησιμοποιώντας χαμηλού επιπέδου παραλληλισμό.

Μέχρι τώρα, αναλύσαμε το γενικευμένο πρόβλημα του χρονικού προγραμματισμού και την πολυπλοκότητά του. Στο επόμενο κεφάλαιο, γίνεται περιγραφή των δεδομένων και των περιορισμών, που πρέπει να τηρούνται για κάθε χρονικό προγραμματισμό εκπαιδευτικών ιδρυμάτων.

# *ΚΕΦΑΛΑΙΟ 3*

## *ΚΑΤΑΡΤΙΣΗ ΩΡΟΛΟΓΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ*

Για να προσεγγίσουμε το πρόβλημα της κατάρτισης ωρολογίου προγράμματος στη γενική εκδοχή είναι απαραίτητο να συγκεντρώσουμε τις προδιαγραφές, που έχουν τεθεί για αυτό και έπειτα να προχωρήσουμε σε μία υλοποίηση, που θα τις καλύπτει. Ανάμεσα σε αυτά τα δύο βήματα, είναι απαραίτητο να δούμε πιο συστηματικά το πρόβλημα, ορίζοντας το μαθηματικά.

# *3.1 Γενικευμένο πρόβλημα ανάθεσης τιμών*

Η κατηγορία προβλημάτων χρονοπρογραμματισμού αφορά προβλήματα αναζήτησης που απαντώνται ακόμα και στην καθημερινότητά μας. Για παράδειγμα, ένας αθλητής σχεδιάζει για όλη τη χρονιά το πότε θα κάνει προπονήσεις, την ημερήσια διάρκεια τους, σε ποιούς τομείς θα δώσει βάρος κάθε φορά και σε ποιες διοργανώσεις θα συμμετάσχει. Γενικότερα, κάθε πρόβλημα που έχει σαν στόχο την αντιστοίχιση ενός συνόλου από χρονικές περιόδους (με χαρακτηριστικά την ώρα έναρξης ή τέλους και τη διάρκειά τους) σε ένα σύνολο από δραστηριότητες ονομάζεται πρόγραμμα χρονοπρογραμματισμού(scheduling problem).

Μια άλλη κατηγορία είναι τα προβλήματα ανάθεσης που διατυπώνονται ως εξής[23] [24]: Έστω ότι έχουμε n οντότητες, m πόρους και ένα σύνολο από K δευτερεύοντες περιορισμούς. Κάθε πόρος είναι μοναδιαίος, δηλαδή μπορεί να ανατεθεί σε μία αποκλειστικά οντότητα. Το ζητούμενο είναι σε καθεμία οντότητα να εκχωρηθεί ένας ξεχωριστός πόρος, έτσι ώστε να ικανοποιούνται οι K δευτερεύοντες περιορισμοί. Αν υπάρχουν πολλές λύσεις, επιλέγεται αυτή που ελαχιστοποιεί τη συνάρτηση κόστους που θα έχει οριστεί. Το γενικευμένο πρόβλημα ανάθεσης είναι όμοιο με το πρόβλημα ανάθεσης, με τη διαφορά ότι κάθε οντότητα μπορεί να απαιτεί και παραπάνω από έναν πόρους. Π.χ. ένα πρόβλημα ανάθεσης είναι το σερβίρισμα m πιάτων με φαγητό (πόροι) σε n πελάτες (οντότητες) ενός εστιατορίου. Σε κάθε πελάτη-οντότητα μπορεί να εκχωρηθεί μόνο ένας πόρος. Επομένως, αυτοί που επιθυμούν και ένα δεύτερο πιάτο θα δυσαρεστηθούν. Αν το πρόβλημα γίνει γενικευμένο, τότε το εμπόδιο αυτό δεν υφίσταται, εφόσον βέβαια επαρκούν τα πιάτα-πόροι.

Θα ορίσουμε ένα συγκεκριμένο τύπο προβλήματος χρονοπρογραμματισμού, για να απεικονίσουμε ένα γενικευμένο πρόβλημα ανάθεσης. Έστω ότι έχουμε n δραστηριότητες, καθεμία από τις οποίες πρέπει να συνδεθεί με μία χρονική περίοδο pi του συνόλου $P = \{pi \mid 1 \leq i \leq mp\}$, οι χρονικές περίοδοι του P δεν αλληλεπικαλύπτονται. Κάθε δραστηριότητα απαιτεί α πόρους, έναν από κάθε πεπερασμένο σύνολο Qi, 1≤i≤ α. Συνεπώς, το γενικευμένο πρόβλημα που προκύπτει αφορά n δραστηριότητες-οντότητες, που πρέπει να συνδεθούν με α πόρους η καθεμία, από το σύνολο $Q \times P, με\ Q = UiQi$. Μέσα στους δευτερεύοντες περιορισμούς θα υπάρχουν και οι εξής δύο:

- Οι πόροι, οι οποίοι θα εκχωρούνται σε μία δραστηριότητα, θα αφορούν την ίδια χρονική περίοδο. Δηλαδή, αν (q1,p1) και (q2, p2) οι πόροι για την ίδια δραστηριότητα, τότε p1=p2.

- Κάθε δραστηριότητα θα πρέπει να αντιστοιχίζεται με κάποιο στοιχείο κάθε συνόλου Qi, 1≤ i≤ α. Με άλλα λόγια, για μία δεδομένη δραστηριότητα και για 1≤ i≤ α, θα υπάρχει q$\in$Qi, έτσι ώστε ο πόρος (q,p), p$\in$P να εκχωρείται σε αυτήν.

Σε κάθε πρόβλημα ανάθεσης θα πρέπει να ορίζεται μια συνάρτηση κόστους, για να ελαχιστοποιηθεί κατά την επίλυση του προβλήματος. Σε αυτό το σημείο έχουμε τη μεγαλύτερη διαφοροποίηση ανάμεσα στα γενικευμένα προβλήματα ανάθεσης χρονοπρογραμματισμού. Ας δούμε μερικά παραδείγματα: Όταν έχουμε να κάνουμε με βάρδιες, ο στόχος συνήθως είναι η ισοκατανομή στις ημέρες των ωρών εργασίας κάθε εργαζόμενου. Μία άλλη περίπτωση είναι όταν μια κατασκευαστική εταιρία αποσκοπεί στο να περατώσει ένα έργο το συντομότερο δυνατόν, και μια άλλη όταν δίνει περισσότερο βάρος στο χρηματικό κόστος και έτσι επιζητά την μείωση των υπερωριών και την ελάττωση της συνεχόμενης χρήσης των μηχανημάτων.

## *3.2 Τα δεδομένα του προβλήματος*

Πριν μοντελοποιήσουμε το πρόβλημα κατάρτισης ωρολογίου προγράμματος θα προσπαθήσουμε να συγκεντρώσουμε όλες τις απαιτήσεις για την ορθή σχεδίασή του.

Ο καμβάς πάνω στον οποίο σχεδιάζεται το ωρολόγιο πρόγραμμα δεν είναι άλλος από τον χρόνο. Ένα πρόγραμμα αφορά συνήθως την κατανομή των μαθημάτων μέσα σε μία εβδομάδα. Η γενικότερη περίπτωση με την οποία θα ασχοληθούμε περιλαμβάνει 5 ημέρες. Κάθε ημέρα κατανέμεται σε 11 χρονοθυρίδες (slots). Η ακριβής χρονική διάρκεια μιας χρονοθυρίδας, καθώς αφορά κάθε εκπαιδευτικό ίδρυμα είναι 1 ώρα.

Εκτός από τις ημέρες και τις διδακτικές περιόδους θα πρέπει να είναι γνωστά και τα σύνολα των αιθουσών, των εργαστηρίων και των καθηγητών. Για κάποιον καθηγητή μπορεί να υπάρξει η απαίτηση κάποιων ορίων στις ώρες ανά μέρα ή στις ημέρες ανά εβδομάδα που μπορεί να διδάξει.

Πολλές φορές για λόγους απλότητας, στη βιβλιογραφία για τα προβλήματα ωρολόγιων προγραμμάτων, ένα μάθημα αναφορικά με τη διάρκειά του έχει σαν μοναδικό χαρακτηριστικό τον αριθμό των χρονοθυρίδων, που καταλαμβάνει στο πρόγραμμα. Και έτσι, αν για παράδειγμα το μάθημα “Ανάλυση αλγορίθμων” έχει αριθμό χρονοθυρίδων ίσο με 5, οποιαδήποτε κατανομή αυτών είναι νόμιμη. Συνεπώς, την 1η μέρα του προγράμματος θα μπορούσαμε να έχουμε 3 ώρες Ανάλυση αλγορίθμων και την 2η ημέρα 2 ώρες. Όμως, θα υπήρχε και η περίπτωση την 1η ημέρα να γίνουν 4

ώρες Ανάλυση αλγορίθμων και την 2η ημέρα 1 ώρα ή ακόμα να γίνουν και οι 5 ώρες σε 1 ημέρα μόνο. Ωστόσο, στα εκπαιδευτικά ιδρύματα τέτοια ζητήματα σχεδόν ποτέ δεν ρυθμίζονται τυχαία. Για κάθε μάθημα δε γνωστοποιείται απλά ο αριθμός των διδακτικών περιόδων του, αλλά ο αριθμός των διαλέξεων του στο πρόγραμμα και η χρονική διάρκεια καθεμιάς από αυτές. Π.χ. για την Ανάλυση αλγορίθμων θα πρέπει να μπορούμε να δηλώσουμε ότι έχουμε μία διάλεξη των 3 και μία των 2 χρονοθυρίδων.

Για κάθε μάθημα δηλώνονται οι καθηγητές, οι οποίοι θα το διδάξουν και αν η διάλεξη γίνεται σε αίθουσα ή σε εργαστήριο. Είναι σημαντικό επίσης, να παρέχεται η δυνατότητα ομαδοποίησης των μαθημάτων, ανάλογα με τις ανάγκες των φοιτητών και των κανονισμών ενός εκπαιδευτικού ιδρύματος. Π.χ. τα μαθήματα ομαδοποιούνται ανά εξάμηνο, καθώς οι φοιτητές συχνά παρακολουθούν όλα τα μαθήματα του εξαμήνου τους.

Τέλος, είναι απαραίτητη η καταγραφή των χρονοθυρίδων, στις οποίες δεν θα είναι διαθέσιμη μια συγκεκριμένη αίθουσα, ένας καθηγητής ή ένα μάθημα.

## *3.3 Αυστηροί περιορισμοί προγράμματος*

Οι αυστηροί περιορισμοί, οι οποίοι πρέπει να ικανοποιούνται από το σύστημα είναι οι εξής 8:

*Περιορισμός 1 από 8.* Μία διάλεξη πρέπει να διδάσκεται μέσα στα πλαίσια της ημέρας.

Έστω μία διάλεξη l με χρονική στιγμή έναρξης S και διάρκεια d. Εάν τώρα το πρόβλημα αποτελείται από D ημέρες και H ώρες διδασκαλίας για κάθε ημέρα, τότε ο τέλος της πρώτης ημέρας θα είναι την χρονική στιγμή H+1, το τέλος της δεύτερης ημέρας την χρονική στιγμή 2*H+1,...,και το τέλος της j ημέρας την χρονική στιγμή j*H+1 αντίστοιχα. Έτσι, θα πρέπει να ισχύει ότι $S \neq (j * H + 1) - i$ για $\forall$ i,j όπου $1 \leq j \leq D$, $1 \leq i < d$. Για παράδειγμα, ας θεωρήσουμε το πρόβλημα μας αποτελείται από 3 ημέρες και 5 ώρες διδασκαλίας για κάθε ημέρα. Σε αυτή την περίπτωση για την διάλεξη l θα ισχύει ότι η μεταβλητή S θα έχει ως πεδίο τιμών το σύνολο {1..15} και έστω ότι η διάρκεια διδασκαλίας της l είναι οι 2 ώρες. Αν εφαρμόσουμε τώρα την παραπάνω σχέση προκύπτει ότι το νέο πεδίο της S θα είναι {1..4,6..9,11..14}, το οποίο απεικονίζεται και στον Πίνακα 1 όπου οι τιμές με υπογράμμιση είναι οι μη επιτρεπόμενες για την μεταβλητή S.

| | 1η ημέρα | 2η ημέρα | 3η ημέρα |
|---|---|---|---|
| 1η ώρα | 1 | 6 | 11 |
| 2η ώρα | 2 | 7 | 12 |
| 3η ώρα | 3 | 8 | 13 |
| 4η ώρα | 4 | 9 | 14 |
| 5η ώρα | **5** | **10** | **15** |

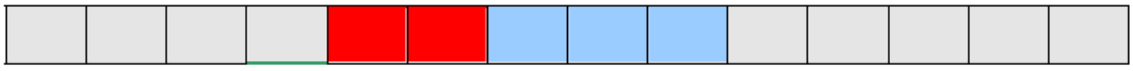
**Πίνακας 1:** Πίνακας περιορισμού 1

*Περιορισμός 2 από 8.* Ένας καθηγητής δε μπορεί να παραδίδει συγχρόνως δύο διαλέξεις.

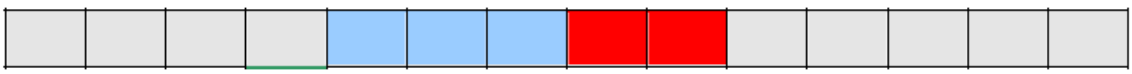
Έστω δύο διαλέξεις l και l' οι οποίες διδάσκονται από τον ίδιο καθηγητή. Οι διαλέξεις l και l' έχουν χρονικές στιγμές έναρξης S και S'και διάρκεια d και d' αντίστοιχα. Οι διαλέξεις δεν πρέπει να συμπίπτουν χρονικά γιατί ο καθηγητής δε μπορεί να βρίσκεται σε δύο διαφορετικές αίθουσες συγχρόνως. Επομένως, θα πρέπει είτε η διάλεξη l να προηγείται της l', είτε η διάλεξη l'να προηγείται της l και άρα θα πρέπει να ισχύει ότι:

$$professor(l) = professor(l') \rightarrow (S + d \leq S') \vee (S' + d' \leq S)$$

Για παράδειγμα, ας θεωρήσουμε ότι οι διαλέξεις l και l' διδάσκονται από τον ίδιο καθηγητή και επιπλέον έχουν διάρκεια 2 και 3 ωρών αντίστοιχα. Θα πρέπει είτε η l να διδάσκεται 2 ώρες νωρίτερα της l' όπως φαίνεται στην Εικόνα 2, είτε η l' να διδάσκεται 3 ώρες νωρίτερα της l, όπως φαίνεται στην Εικόνα 3.



**Εικόνα 2:** Εικόνα περιορισμού 2



**Εικόνα 3:** Εικόνα περιορισμού 2

*Περιορισμός 3 από 8.* Δύο διαλέξεις που ανήκουν σε διαφορετικά μαθήματα και τα μαθήματα τους ανήκουν στην ίδια ομάδα μαθημάτων, δε μπορούν να διδάσκονται συγχρόνως.

Έστω δύο διαλέξεις l και l' οι οποίες ανήκουν σε διαφορετικά μαθήματα. Έστω ότι τα μαθήματα αυτά ανήκουν στην ίδια ομάδα μαθημάτων g. Οι διαλέξεις l και l' έχουν χρονικές στιγμές έναρξης S και S' και διάρκεια d και d' αντίστοιχα. Οι διαλέξεις δεν πρέπει να συμπίπτουν χρονικά εφόσον έχουμε θεωρήσει ότι τα μαθήματα της ίδιας ομάδας δεν επιτρέπεται να διδάσκονται συγχρόνως. Επομένως, θα πρέπει είτε η διάλεξη l να προηγείται της l', είτε η διάλεξη l'να προηγείται της l και άρα θα πρέπει να ισχύει ότι

$$\left(course(l) \neq course\left(l^{'}\right)\right) \wedge (course(l) \in g) \wedge course(l') \in g) \rightarrow \left(S + d \leq S^{'}\right) \vee \left(S^{'} + d^{'} \leq S\right)$$

*Περιορισμός 4 από 8.* Δύο διαλέξεις δε μπορούν να διδάσκονται στην ίδια αίθουσα συγχρόνως.

Έστω δύο διαλέξεις l και l' οι οποίες διδάσκονται στην ίδια αίθουσα. Οι διαλέξεις l και l' έχουν χρονικές στιγμές έναρξης S και S'και διάρκεια d και d' αντίστοιχα. Οι διαλέξεις δεν πρέπει να συμπίπτουν χρονικά γιατί η αίθουσα δε μπορεί να φιλοξενεί δύο διαφορετικές διαλέξεις μαθημάτων συγχρόνως. Επομένως, θα πρέπει είτε η διάλεξη l να προηγείται της l', είτε η διάλεξη l'να προηγείται της l και άρα θα πρέπει να ισχύει ότι

$$\left(course(l) \neq course\left(l^{'}\right)\right) \wedge (course(l) \in g) \wedge course(l') \in g) \rightarrow \left(S + d \leq S^{'}\right) \vee \left(S^{'} + d^{'} \leq S\right)$$

*Περιορισμός 5 από 8.* Μία διάλεξη δε μπορεί να διδαχθεί τις χρονικές στιγμές που ο καθηγητής δεν είναι διαθέσιμος.

Έστω μία διάλεξη l, η οποία διδάσκεται από τον καθηγητή t. Η διάλεξη l έχει χρονική στιγμή έναρξης S και διάρκεια d. Αν το πρόβλημα έχει H ώρες διδασκαλίας για κάθε μέρα και ο καθηγητής δεν είναι διαθέσιμος κάποιες χρονικές στιγμές, τότε θα πρέπει να ισχύει ότι

$$(professor(l) = t) \wedge notavailable(t, p) \wedge \left(\frac{p-i-1}{H} = \frac{p-1}{H} = \frac{p-1}{H}\right) \rightarrow S + i \neq p$$

$$για\ \forall p, \text{i } Όπου\ \text{p} \in \text{P}, 0 \leq \text{ i} < d$$

Η συνθήκη (S+ i -1) /H =(p-1)/H, 0≤ i< d εξασφαλίζει ότι δεν πρόκειται να διαγραφούν από το πεδίο της S τιμές, οι οποίες δεν αντιστοιχούν στην ίδια ημέρα με την χρονική στιγμή p όταν για αυτή την χρονική στιγμή ο καθηγητής δεν είναι διαθέσιμος. Για παράδειγμα, ας θεωρήσουμε το πρόβλημα μας αποτελείται από 3 ημέρες και 5 ώρες διδασκαλίας για κάθε ημέρα. Σε αυτή την περίπτωση για την διάλεξη l θα ισχύει ότι η μεταβλητή S θα έχει ως πεδίο τιμών το σύνολο {1..15} και έστω ότι η διάρκεια διδασκαλίας της l είναι οι 3 ώρες. Έστω τώρα, ότι η διάλεξη l διδάσκεται από τον καθηγητή t, ο οποίος δεν είναι διαθέσιμος τις χρονικές στιγμές {4,11}. Αν εφαρμόσουμε τώρα την παραπάνω σχέση προκύπτει ότι το νέο πεδίο της S θα είναι {1,5..10,12..15}, το οποίο απεικονίζεται και στον Πίνακα 2 όπου οι τιμές με υπογράμμιση είναι οι μη επιτρεπόμενες για την μεταβλητή S.

| | 1η ημέρα | 2η ημέρα | 3η ημέρα |
|---|---|---|---|
| 1η ώρα | 1 | 6 | **11** |
| 2η ώρα | **2** | 7 | 12 |
| 3η ώρα | **3** | 8 | 13 |
| 4η ώρα | **4** | 9 | 14 |
| 5η ώρα | 5 | 10 | 15 |

**Πίνακας 2:** Παράδειγμα περιορισμού 5

Παρατηρούμε ότι, ενώ ο καθηγητής δεν είναι διαθέσιμος τις χρονικές στιγμές 4 και 11, παρόλα αυτά απομακρύνονται από το πεδίο της S οι τιμές 2,3,4 και 11 και αυτό γιατί η διάλεξη έχει διάρκεια 3 ώρες δε μπορεί να ξεκινήσει τις χρονικές στιγμές 2 και 3 γιατί θα απαιτούνται να ήταν ο καθηγητής στο μάθημα την χρονική στιγμή 4. Με την ίδια λογική θα έπρεπε να διαγραφούν λόγω της μη διαθεσιμότητας του την χρονική στιγμή 11 οι τιμές 10 και 9, επειδή όμως ανήκουν σε διαφορετική ημέρα από την 11 δε διαγράφονται.

*Περιορισμός 6 από 8.* Δε μπορεί να γίνει ανάθεση μίας αίθουσας σε ένα μάθημα τις χρονικές στιγμές που αυτή δεν είναι διαθέσιμη.

Έστω μία διάλεξη l, η οποία διδάσκεται σε μία αίθουσα r. Η διάλεξη l έχει χρονική στιγμή έναρξης S και διάρκεια d. Αν το πρόβλημα έχει H ώρες διδασκαλίας για κάθε μέρα και η αίθουσα δεν είναι διαθέσιμη κάποιες χρονικές στιγμές τότε θα πρέπει να ισχύει ότι:

$$(room(l) = r) \wedge notavailable(r, p) \wedge (\frac{p - i - l}{H} = \frac{p - l}{H}) \rightarrow S + i \neq p$$

$για\ \forall p, \mathrm{i}\ όπου\ \mathrm{p} \in \mathrm{P}, 0 \leq \mathrm{i} < d$

Παρόμοιο παράδειγμα μπορούμε να θεωρήσουμε αυτό του Πίνακα 2.

*Περιορισμός 7 από 8.* Μία διάλεξη, η οποία είναι εργαστηριακή, δε μπορεί να διδάσκεται σε μη εργαστηριακή αίθουσα.

*Περιορισμός 8 από 8.* Μία διάλεξη, η οποία δεν είναι εργαστηριακή, δε μπορεί να διδάσκεται σε εργαστηριακή αίθουσα.

Μέχρι τώρα, περιγράψαμε τα δεδομένα και τους περιορισμούς που πρέπει να τηρούνται, σε κάθε εκπαιδευτικό ίδρυμα, για την ορθή σχεδίαση ενός προγράμματος διδασκαλίας. Στο επόμενο κεφάλαιο, περιγράφονται τα δεδομένα και οι μεταβλητές που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία. Επίσης, αναλύονται οι αλγόριθμοι που υλοποιήθηκαν και γίνεται πλήρης περιγραφή της εφαρμογής, με τη βοήθεια διαγραμμάτων ροής.

# *ΚΕΦΑΛΑΙΟ 4*

# *ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ*

## *4.1 Δεδομένα και περιορισμοί προβλήματος*

Παρακάτω περιγράφονται τα δεδομένα που είναι απαραίτητα για την κατασκευή ωρολογίου προγράμματος του Πανεπιστημίου δυτικής Μακεδονίας:

**Αίθουσες και εργαστήρια:** Αφορά τις αίθουσες διδασκαλίας και τα εργαστήρια που είναι διαθέσιμα για τη διεξαγωγή των μαθημάτων. Επιλέξαμε το πλήθος τους να είναι μεταβλητό, καθώς μπορεί στο μέλλον να αλλάξει, ανάλογα με τις διαθέσιμες εγκαταστάσεις του Πανεπιστημίου.

**Ημέρες:** Οι ημέρες διεξαγωγής μαθημάτων θα είναι 5. Το πλήθος των ημερών δεν είναι μεταβλητό, εφόσον πρόκειται για πρόγραμμα πανεπιστημίου και δεν προβλέπεται να αλλάξει στο μέλλον. Οι μέρες που διεξάγονται τα μαθήματα σύνισταν να είναι από Δευτέρα μέχρι και Παρασκευή.

**Διδακτικές Περίοδοι:** Οι διδακτικές περίοδοι αφορούν τις ώρες στις οποίες θα γίνονται τα μαθήματα και είναι μη μεταβλητό μέγεθος, εφόσον τα μαθήματα διεξάγονται από τις 9:00 μέχρι τις 20:00, δηλαδή 11 χρονοθυρίδες την ημέρα και 11*5=55 χρονοθυρίδες την εβδομάδα.

**Μαθήματα:** Τα χαρακτηριστικά του κάθε μαθήματος είναι το όνομα του, σε ποιό εξάμηνο και σε ποιά κατεύθυνση ανήκει, πόσες χρονοθυρίδες απαιτούνται εβδομαδιαία καθώς επίσης και από ποιον καθηγητή διεξάγεται. Τα μαθήματα είναι ένα μεταβλητό μέγεθος, καθώς είναι πιθανόν να αλλάξει είτε ο καθηγητής είτε να αλλάξουν οι ώρες διδασκαλίας, ακόμη και να προστεθεί/αφαιρεθεί ένα μάθημα.

**Καθηγητές:** Πρόκειται για το εκπαιδευτικό προσωπικό, που διδάσκει τα μαθήματα. Τα χαρακτηριστικά των καθηγητών είναι το ονοματεπώνυμο, οι διαθέσιμες ώρες διδασκαλίας και ποια τα μαθήματα που διδάσκουν. Οι διαθέσιμες ώρες διδασκαλίας των καθηγητών, πιθανότατα διαφέρουν κάθε χρόνο, οπότε κρίθηκε απαραίτητο η μεταβλητή καθηγητής να μην είναι σταθερή.

Οι περιορισμοί που υπάρχουν προκειμένου να είναι δυνατή η ορθή κατασκευή του ωρολογίου προγράμματος είναι οι εξής 5:

- Έλεγχος διαθεσιμότητας αίθουσας ή εργαστηρίου. Δεν είναι εφικτό να διδάσκουν περισσότεροι από ένας καθηγητές την ίδια χρονική στιγμή στην ίδια αίθουσα.
- Έλεγχος διαθεσιμότητας κάθε καθηγητή σύμφωνα με τις ώρες που δήλωσαν ότι διαθέτουν, για την διεξαγωγή των μαθημάτων τους.
- Να μην συμπίπτουν μαθήματα του ίδιου εξαμήνου την ίδια χρονική περίοδο, εκτός και αν ανήκουν σε διαφορετικές κατευθύνσεις.
- Έλεγχος για την συνέχεια του μαθήματος, π.χ. αν το μάθημα είναι δίωρο να μην διασπάται σε 2 μονόωρα.
- Ένας καθηγητής δε μπορεί να διδάσκει σε δύο αίθουσες ταυτόχρονα.

Για την υλοποίηση της εφαρμογής και την ικανοποίηση όλων των περιορισμών χρησιμοποιήσαμε ευρετικούς αλγόριθμους. Προφανώς το καλύτερο δυνατό αποτέλεσμα θα είναι όταν γίνει ανάθεση μαθημάτων 100%.

## *4.2 Μεταβλητές προγράμματος*

Παρακάτω καταγράφουμε τις μεταβλητές που θα χρησιμοποιήσουμε για την αποθήκευση δεδομένων στο πρόγραμμα και τις αναλύουμε επιγραμματικά.

**number_of_classes:** μεταβλητή τύπου integer, αποθηκεύει το διαθέσιμο πλήθος αιθουσών διδασκαλίας.

**number_of_labs:** μεταβλητή τύπου integer, αποθηκεύει το διαθέσιμο πλήθος εργαστηρίων.

**number_of_professors:** μεταβλητή τύπου integer, αποθηκεύει το πλήθος των καθηγητών που διδάσκουν στο πανεπιστήμιο.

**names_of_professors[ ]:** μονοδιάστατος πίνακας τύπου char, έχει μέγεθος ίσο με τον αριθμό των καθηγητών και καταχωρούνται τα ονόματα των καθηγητών.

**available_hours_professors[ ][ ] :** δισδιάστατος πίνακας τύπου integer μεγέθους αριθμός_καθηγητών×55. Το 55 προκύπτει από τις συνολικές χρονοθυρίδες εβδομαδιαίως, δηλ 5ημέρες την εβδομάδα επί 11ώρες την ημέρα. Σ' αυτόν τον πίνακα καταχωρούνται οι διαθέσιμες ώρες διδασκαλίας κάθε καθηγητή.

**hours_of_teaching_professors[ ][ ]:** δισδιάστατος πίνακας τύπου integer, στον οποίο καταχωρείται η διάρκεια της κάθε διάλεξης κάθε καθηγητή.

**course_name[ ][ ]:** δισδιάστατος πίνακας τύπου char, στον οποίο καταχωρούνται, κατά αντιστοιχία με τον πίνακα hours_of_teaching_professors, τα ονόματα των μαθημάτων.

**lab[ ][ ]:** δισδιάστατος πίνακας τύπου char, στον οποίο καταχωρείται κατά αντιστοιχία με τον πίνακα hours_of_teaching_professors, η πληροφορία αν το μάθημα διεξάγεται σε αίθουσα ή σε εργαστήριο. Αν διεξάγεται σε αίθουσα καταχωρείται στο αντίστοιχο κελί η τιμή 'n' (no), αν διεξάγεται σε εργαστήριο η τιμή 'y' (yes).

**semester[ ][ ]:** δισδιάστατος πίνακας τύπου integer, στον οποίο καταχωρείται κατά αντιστοιχία με τον πίνακα *hours_of_teaching_professors*, σε ποιο εξάμηνο διδάσκεται το μάθημα (1 για το πρώτο, 2 για το δεύτερο κ.ο.κ.)

**direction[ ][ ]:** δισδιάστατος πίνακας τύπου char, στον οποίο καταχωρείται, κατά αντιστοιχία με τον πίνακα *hours_of_teaching_professors,* σε ποια κατεύθυνση διδάσκεται το μάθημα. 'Α' για την πρώτη, 'Β' για την δεύτερη, 'Γ' για την Τρίτη. Αν το μάθημα δεν ανήκει σε κάποια κατεύθυνση καταχωρείται η τιμή 'Ω'.

**slots[ ] [ ]:** δισδιάστατος πίνακας (Πίνακας 3) τύπου integer και είναι ένας βοηθητικός δισδιάστατος πίνακας 5×11 (5 μέρες την εβδομάδα, 11 ώρες την ημέρα), ο οποίος δείχνει τις μέρες και τις ώρες διεξαγωγής των μαθημάτων. Στο πρόγραμμα

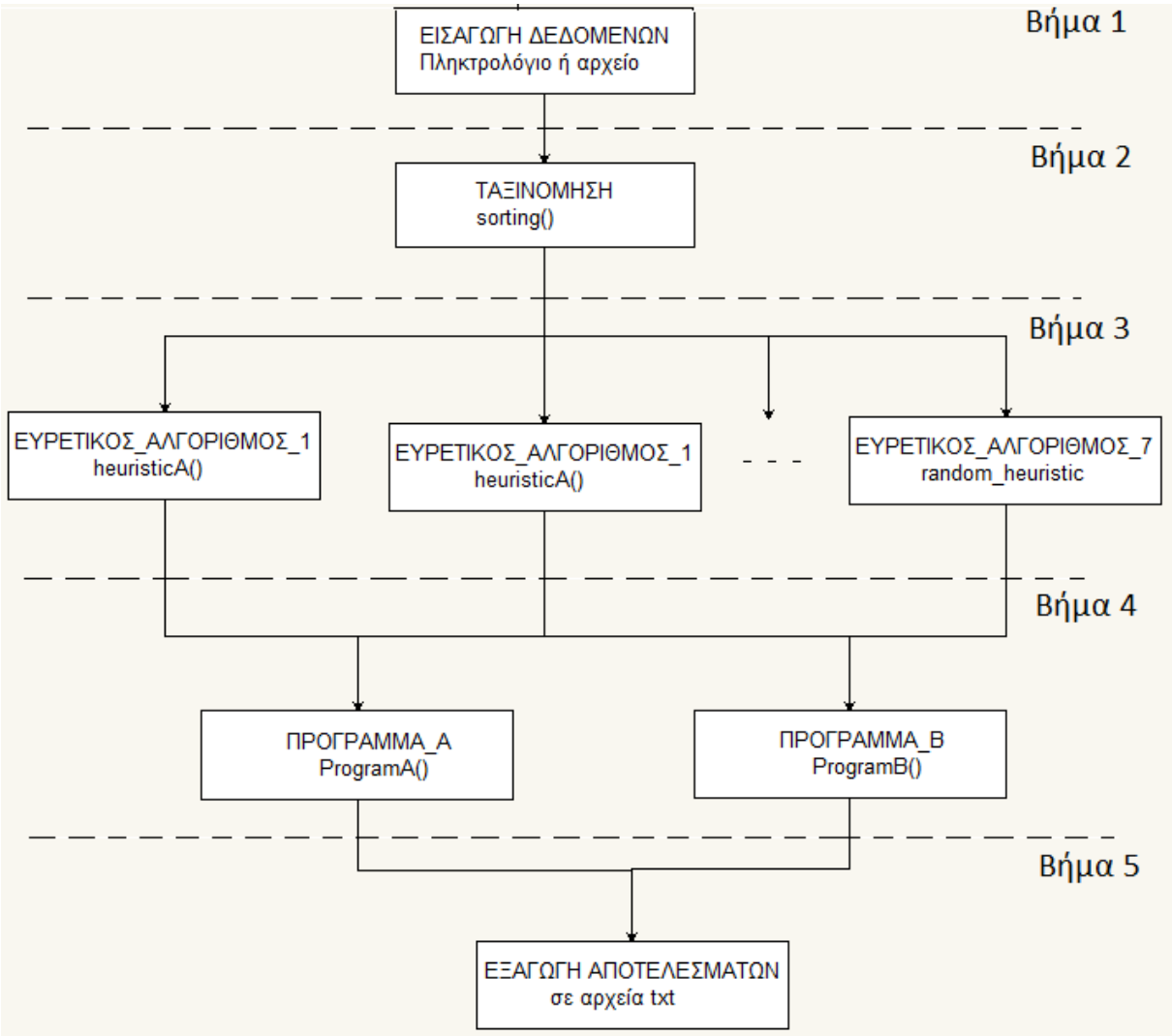
αναφερόμαστε σε συγκεκριμένη χρονοθυρίδα ανάλογα με τον "αριθμό" που της δώσαμε. Για παράδειγμα, για να αναφερθούμε στις 9-10 την Δευτέρα χρησιμοποιούμε τον αριθμό "1", στις 10-11 τον αριθμό "2" κ.ο.κ. Σκόπιμα δεν χρησιμοποιήσαμε τους αριθμούς 12, 24, 36 και 48, για να μπορεί το πρόγραμμα να καταλαβαίνει με αυτόν τον τρόπο πότε αλλάζει ημέρα. Αν για παράδειγμα ένας καθηγητής είχε δηλώσει ότι μπορεί να διδάξει τις ώρες 10, 11, 12 (αντί 13), 13, 14 και το μάθημα του είναι τρίωρο, το πρόγραμμα θα τοποθετούσε το μάθημα (αν φυσικά πληρούσε και τους υπόλοιπους περιορισμούς π.χ. αν έβρισκε κενή αίθουσα) στις ώρες 10, 11 και 12. Επομένως, για την αποφυγή τέτοιων λαθών δεν χρησιμοποιούμε το 12.

| | **ΔΕΥΤΕΡΑ** | **ΤΡΙΤΗ** | **ΤΕΤΑΡΤΗ** | **ΠΕΜΠΤΗ** | **ΠΑΡΑΣΚΕΥΗ** |
|---|---|---|---|---|---|
| **9-10** | 1 | 13 | 25 | 37 | 49 |
| **10-11** | 2 | 14 | 26 | 38 | 50 |
| **11-12** | 3 | 15 | 27 | 39 | 51 |
| **12-13** | 4 | 16 | 28 | 40 | 52 |
| **13-14** | 5 | 17 | 29 | 41 | 53 |
| **14-15** | 6 | 18 | 30 | 42 | 54 |
| **15-16** | 7 | 19 | 31 | 43 | 55 |
| **16-17** | 8 | 20 | 32 | 44 | 56 |
| **17-18** | 9 | 21 | 33 | 45 | 57 |
| **18-19** | 10 | 22 | 34 | 46 | 58 |
| **19-20** | 11 | 23 | 35 | 47 | 59 |

**Πίνακας 3:** Αντιστοίχηση χρονοθυρίδων με αριθμούς

## 4.3 Περιγραφή εφαρμογής

Για καλύτερη δυνατή κατανόηση της εφαρμογής, τα 5 κύρια μέρη του προγράμματος αναπαρίστανται σε διάγραμμα ροής στο Σχήμα 7 και στην συνέχεια αναλύονται διεξοδικά.




**Σχήμα 6:** Διάγραμμα ροής της εφαρμογής

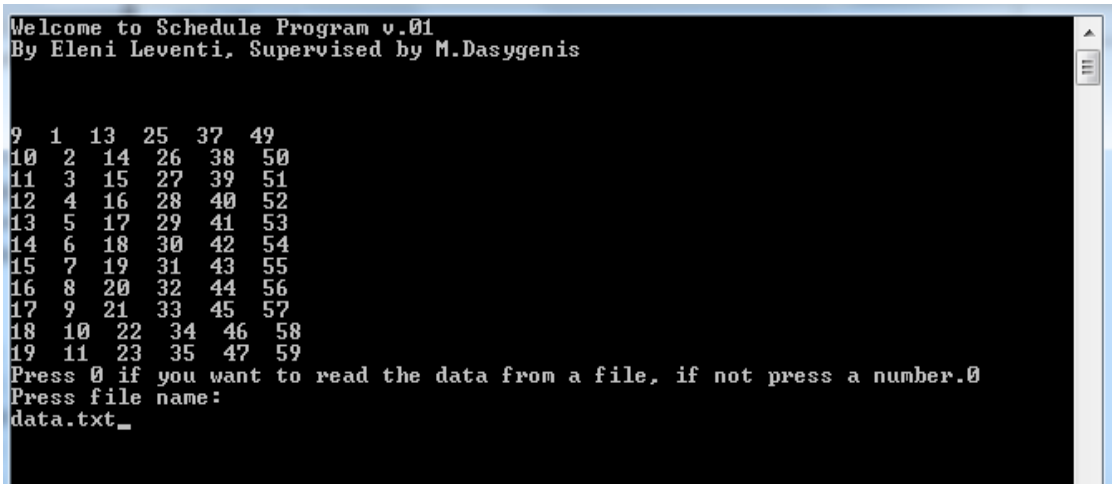
***Βήμα 1 από 5. Εισαγωγή δεδομένων***

Αρχικά στο πρόγραμμα, όπως είναι αναμενόμενο, εισάγονται τα δεδομένα του προβλήματος τα οποία αναφερθήκαν στην παράγραφο 4.2. Ο χρήστης επιλέγει αν επιθυμεί να εισάγει τα δεδομένα από το πληκτρολόγιο ή να διαβαστούν από αρχείο τύπου .txt, καθώς και το όνομα του αρχείου, όπως φαίνεται και στην Εικόνα 4. Η εισαγωγή δεδομένων από αρχείο είναι προτιμότερη επιλογή, για λόγους χρηστικότητας και αποθήκευσης. Αν ο χρήστης εισάγει κάθε δεδομένο ξεχωριστά από το πληκτρολόγιο κατά την διάρκεια της εκτέλεσης, δεν έχει την δυνατότητα να αλλάξει κάποια λάθος πληκτρολόγηση, αλλά θα πρέπει να εκτελέσει από την αρχή το πρόγραμμα και χωρίς μάλιστα να έχουν αποθηκευτεί τα δεδομένα που είχε ήδη εισάγει. Επιπλέον, με την χρήση αρχείου, δίνεται η δυνατότητα στο πρόγραμμα να αυτοματοποιηθεί και να συνδεθεί με άλλες εφαρμογές, όπως π.χ. εφαρμογές web.

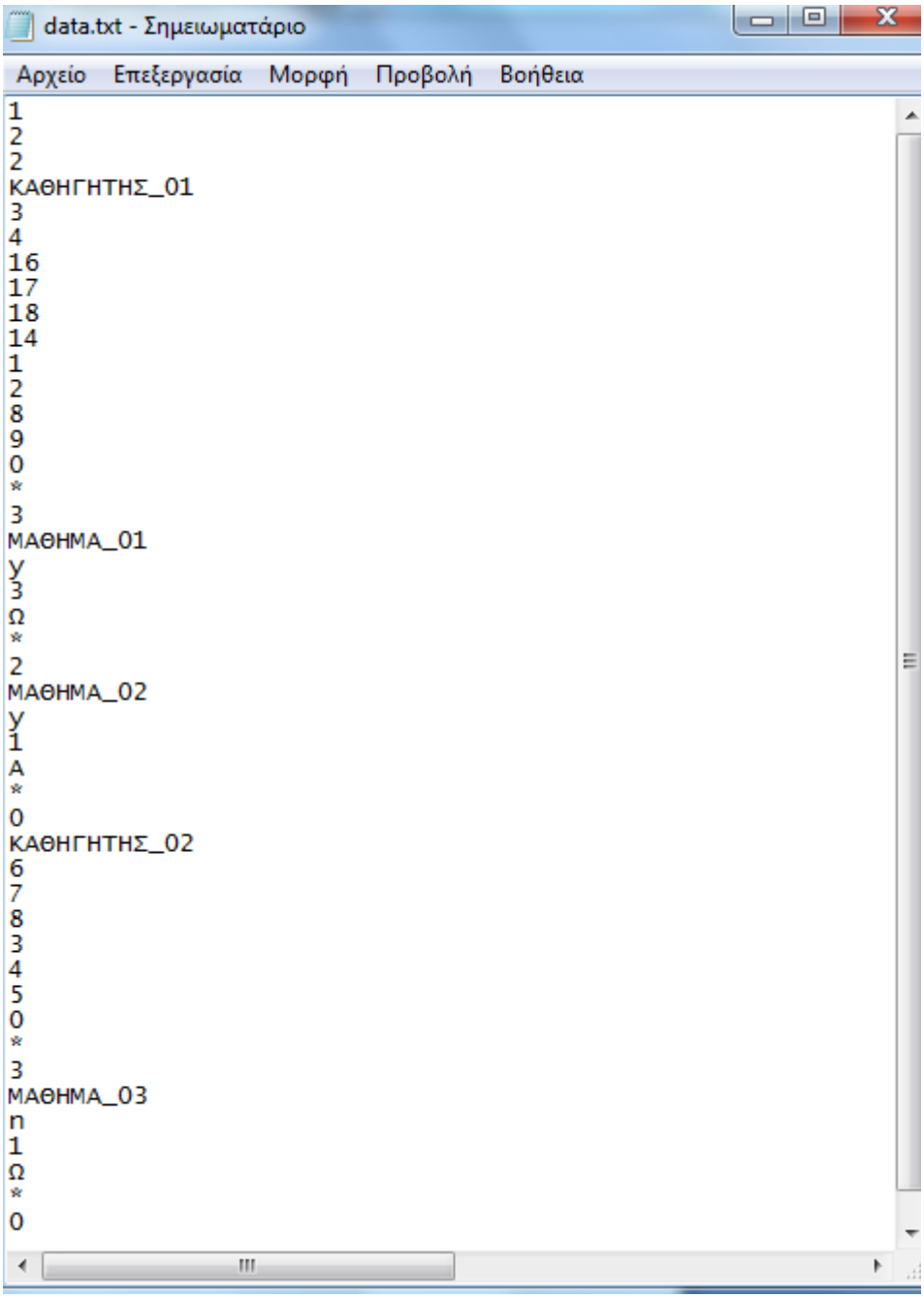
Εντολές στο πρόγραμμα:

```
cout<<"Press 0 if you want to read the data from a file, if not press
a number.";
cin>>select;


filename=(char*)malloc(30*sizeof(char));
cout<<"Press file name:\n";
cin>>filename;
```




**Εικόνα 4:** Παράθυρο εκτέλεσης της εφαρμογής




**Εικόνα 5:** Μορφή αρχείου .txt για είσοδο δεδομένων

Στο αρχείο, που εισάγονται τα δεδομένα, είναι απαραίτητο να έχει την μορφή που φαίνεται στη Εικόνα 5. Ακολουθεί αναλυτική περιγραφή του περιεχομένου του ενδεικτικού αρχείου, με την σειρά που εμφανίζονται τα στοιχεία.

**1:** πλήθος αιθουσών, το οποίο καταχωρείται στη μεταβλητή number_of_classes του προγράμματος.

**2:** πλήθος εργαστηρίων, καταχωρείται στη μεταβλητή number_of_labs του προγράμματος.

**2:** πλήθος καθηγητών, καταχωρείται στη μεταβλητή number_of_professors του προγράμματος.

**ΚΑΘΗΓΗΤΗΣ_1:** όνομα καθηγητή, καταχωρείται στην θέση 0 του πίνακα names_of_professors[].

**3,4,16,17,18,14,1,2,8,9** και **0:** διαθέσιμες χρονοθυρίδες του ΚΑΘΗΓΗΤΗ_1 και καταχωρούνται στον πίνακα available_hours_professors[][], στα κελιά [0][0], [0][1], [0][2],... και [0][10] αντίστοιχα. Ο αριθμός 0 δηλώνει ότι ο καθηγητής δεν έχει άλλες διαθέσιμες χρονοθυρίδες.

***:** λειτουργεί ως διαχωριστικό των διαθέσιμων ωρών και των μαθημάτων που ακολουθούν, ώστε να γίνεται κατανοητό από το πρόγραμμα, για την σωστή ανάθεση των δεδομένων στους κατάλληλους πίνακες κάθε φορά.

**3:** διάρκεια του μαθήματος, καταχωρείται στον πίνακα hours_of_teaching_professors[][], στο κελί [0][0].

**ΜΑΘΗΜΑ_1:** όνομα μαθήματος, καταχωρείται στον πίνακα course_name[][], στο κελί [0][0].

**y** (yes): το μάθημα διεξάγεται σε εργαστηριακή αίθουσα, καταχωρείται στον πίνακα lab[][], στο κελί [0][0].

**3:** εξάμηνο, στο οποίο διεξάγεται το μάθημα, καταχωρείται στον πίνακα semester[][] στο κελί [0][0].

**Ω:** κατεύθυνση, στην οποία διεξάγεται το μάθημα, το γράμμα Ω δηλώνει ότι είναι υποχρεωτικό για όλες τις κατευθύνσεις. Καταχωρείται στον πίνακα direction[ ][ ] στο κελί [0][0].

***:** λειτουργεί ως διαχωριστικό για κάθε μάθημα.

**2:** διάρκεια επόμενου μαθήματος, καταχωρείται στον πίνακα hours_of_teaching_professors[][], στο κελί [0][1].

**ΜΑΘΗΜΑ_02:** όνομα μαθήματος, καταχωρείται στον πίνακα course_name[][], στο κελί [0][1].

**y:** το μάθημα διεξάγεται σε εργαστηριακή αίθουσα καταχωρείται στον πίνακα lab[][], στο κελί [0][1].

**1:** εξάμηνο, στο οποίο διεξάγεται το μάθημα, καταχωρείται στον πίνακα semester[][] στο κελί [0][1].

**A:** κατεύθυνση, στην οποία διεξάγεται το μάθημα, καταχωρείται στον πίνακα direction[ ][ ] στο κελί [0][1].

***:** λειτουργεί ως διαχωριστικό για κάθε μάθημα.

**0:** ο καθηγητής δεν έχει άλλες ώρες διδασκαλίας-μαθήματα.

**ΚΑΘΗΓΗΤΗΣ_02:** όνομα επόμενου καθηγητή, καταχωρείται στην θέση 1 του πίνακα names_of_professors[].

**6,7,8,3,4,5** και **0**: διαθέσιμες χρονοθυρίδες του ΚΑΘΗΓΗΤΗ_2 και καταχωρούνται στον πίνακα available_hours_professors[][], στα κελιά [1][0], [1][1], [1][2],... και [1][10] αντίστοιχα. Ο αριθμός 0 δηλώνει ότι ο καθηγητής δεν έχει άλλες διαθέσιμες χρονοθυρίδες.

***:** λειτουργεί ως διαχωριστικό των διαθέσιμων ωρών του ΚΑΘΗΓΗΤΗΣ_02 και των μαθημάτων που ακολουθούν.

**3:** διάρκεια του μαθήματος, καταχωρείται στον πίνακα hours_of_teaching_professors[][], στο κελί [1][0].

**ΜΑΘΗΜΑ_03:** όνομα μαθήματος, καταχωρείται στον πίνακα course_name[][], στο κελί [1][0].

**n:** το μάθημα διεξάγεται σε μη εργαστηριακή αίθουσα, καταχωρείται στον πίνακα lab[][], στο κελί [1][0].

**1:** εξάμηνο, στο οποίο διεξάγεται το μάθημα, καταχωρείται στον πίνακα semester[][] στο κελί [1][0].

**Ω:** κατεύθυνση, στην οποία διεξάγεται το μάθημα, καταχωρείται στον πίνακα direction[ ][ ], στο κελί [1][0].

***:** λειτουργεί ως διαχωριστικό για κάθε μάθημα.

**0:** ο καθηγητής δεν έχει άλλες ώρες διδασκαλίας-μαθήματα.

***Βήμα 2 από 5. Ταξινόμηση – sorting()***

Η συνάρτηση sorting(), εφόσον υπολογίσει τις συνολικές διαθέσιμες χρονοθυρίδες διδασκαλίας και τις συνολικές ώρες διδασκαλίας κάθε καθηγητή, εκτελεί τις εξής ταξινομήσεις:

- Αύξουσα ταξινόμηση *διαθέσιμων* χρονοθυρίδων κάθε καθηγητή (*available_hours_professors*),π.χ. όταν εισάγονται από το πληκτρολόγιο οι χρονοθυρίδες 15,5,4 θα ταξινομούνται ως εξής 4,5,15. Η συγκεκριμένη ταξινόμηση

εκτελείται προς διευκόλυνση του χρήστη, ώστε να μην χρειάζεται να τις ταξινομεί ο ίδιος κάθε φορά που θέλει να προσθέσει κάποιες διαθέσιμες ώρες (χρονοθυρίδες) επιπλέον.

- Φθίνουσα ταξινόμηση χρονοθυρίδων *διδασκαλίας* κάθε καθηγητή (*hours_of_teaching_professors*), π.χ. αν ένας καθηγητής έχει ένα 2ωρο μάθημα και ένα 3ωρο προτεραιότητα καταχώρησης στο πρόγραμμα θα έχει το 3ωρο. Η συγκεκριμένη ταξινόμηση εκτελείται λόγω του ότι είναι πιο εύκολο να βρεθούν 2 διαθέσιμες συνεχόμενες χρονοθυρίδες απ' ότι 3.

Τέλος, γίνεται έλεγχος αν η ταξινόμηση είναι η αναμενόμενη, εξάγοντας τα αποτελέσματα σε ένα αρχείο με όνομα "professors_data_classified.txt".

Οι ταξινομήσεις που αναφέρθηκαν επηρεάζουν άμεσα όλους τους πίνακες, στους οποίους είναι καταχωρημένα τα δεδομένα. Επομένως, πρέπει να δοθεί ιδιαίτερη προσοχή ώστε να μην αλλοιωθούν τα δεδομένα. Για παράδειγμα, αν αλλάξει θέση στον πίνακα names_of_professors ένας καθηγητής λόγω λιγότερων διαθέσιμων χρονοθυρίδων είναι αναγκαίο να αλλάξουν θέση και τα υπόλοιπα δεδομένα που αφορούν τον συγκεκριμένο καθηγητή. Δηλαδή, να γίνουν αλλαγές σε αντιστοιχία με τον πίνακα names_of_professors και στους πίνακες course_name, lab, hours_of_teaching_professors, semester και direction.

***Βήμα 3 από 5.Επιλογή πρώτου ευρετικού αλγορίθμου*** **(heuristic)**

Ο χρήστης καλείται να επιλέξει τον ευρετικό αλγόριθμο που επιθυμεί. Επιπλέον, έχει την δυνατότητα να διαλέξει την εκτέλεση όλων των ευρετικών αλγορίθμων και στο τέλος να διαλέξει ποιο πρόγραμμα καλύπτει τις ανάγκες του ανάλογα με τα αποτελέσματα του κάθε αλγορίθμου.

Οι ευρετικοί αλγόριθμοι του προγράμματος είναι οι εξής:

*heuristicA:* Αύξουσα ταξινόμηση συνολικών διαθέσιμων χρονοθυρίδων διδασκαλίας κάθε καθηγητή (total_available_hours_professors). Δηλαδή, ο καθηγητής με τις λιγότερες διαθέσιμες χρονοθυρίδες έχει προτεραιότητα καταχώρησης στο πρόγραμμα.

*heuristicB*: Αύξουσα ταξινόμηση συνολικών ωρών διδασκαλίας κάθε καθηγητή (total_hours_of_teaching_of_professor). Σ' αυτόν τον αλγόριθμο προτεραιότητα καταχώρησης έχει ο καθηγητής με τις περισσότερες ώρες διδασκαλίας.

*heuristicC:* Συνδυασμός heuristicA και heuristicB. Δηλαδή, αύξουσα ταξινόμηση σύμφωνα με τον αριθμό που προκύπτει αν αφαιρέσουμε τις συνολικές ώρες διδασκαλίας από τις συνολικές διαθέσιμες κάθε καθηγητή.

*heuristicD:* αύξουσα ταξινόμηση εξαμήνου, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 1ου εξαμήνου, στην συνέχεια του 2ου κ.ο.κ ανά καθηγητή.

*heuristicE:* Φθίνουσα ταξινόμηση εξαμήνων, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 10ου εξαμήνου, στην συνέχεια του 9ου κ.ο.κ για όλα τα μαθήματα.

*heuristicF:* Αύξουσα ταξινόμηση εξαμήνων, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 1ου εξαμήνου, στην συνέχεια του 2ου κ.ο.κ για όλα τα μαθήματα.

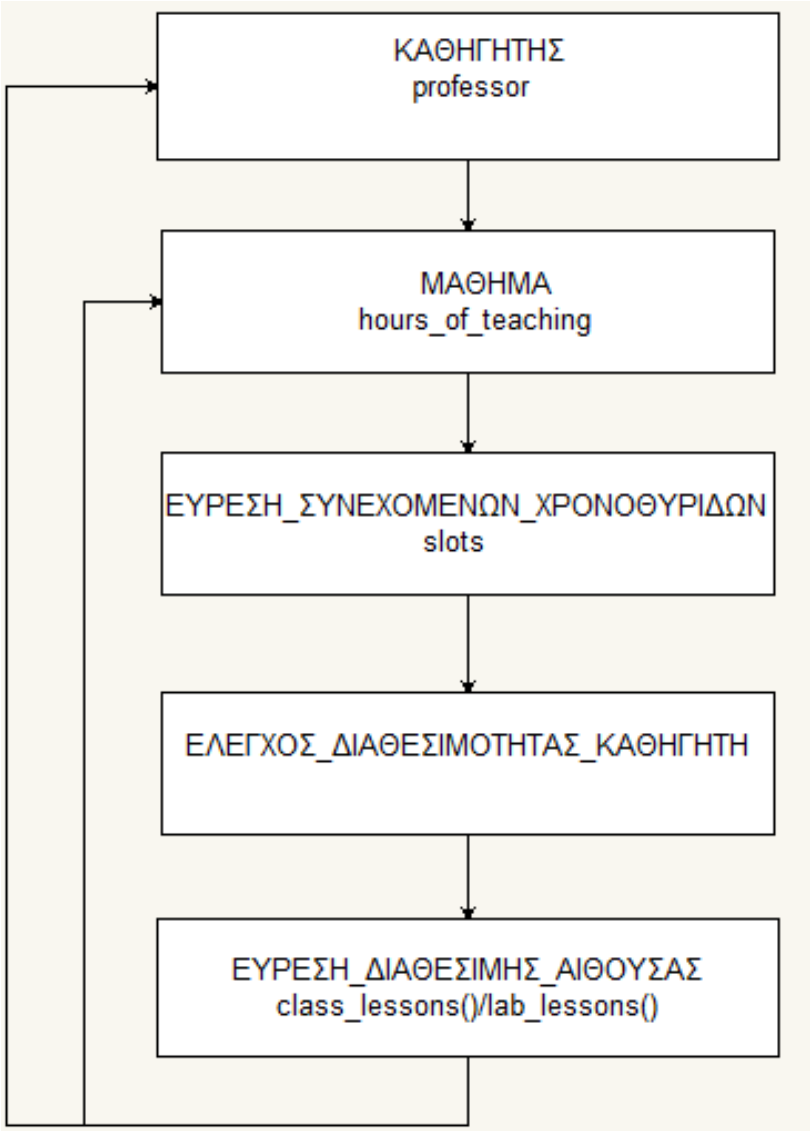
*random_heuristic:* Επιλέγει τυχαία την σειρά που θα καταχωρηθούν οι καθηγητές στο πρόγραμμα.

***Βήμα 4 από 5. Επιλογή δεύτερου ευρετικού αλγορίθμου*** (program)

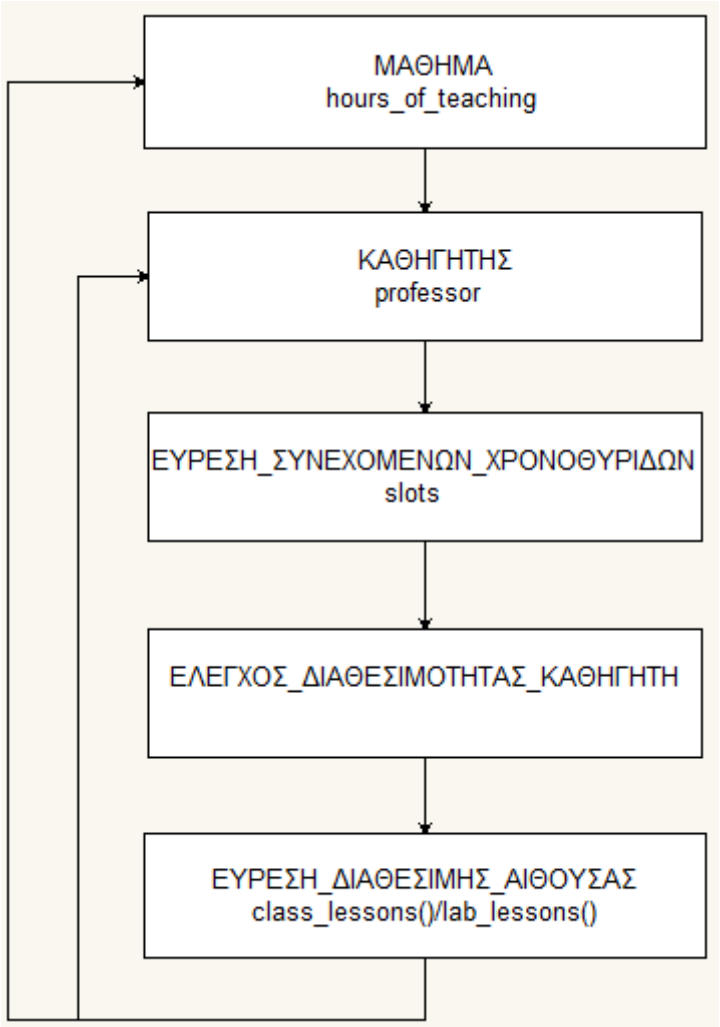
Σ' αυτό το βήμα ο χρήστης καλείται να επιλέξει ακόμη έναν αλγόριθμο από τους επόμενους 2:

programA: Καλώντας την συνάρτηση programA(), η εφαρμογή επιχειρεί να καταχωρήσει στο ωρολόγιο πρόγραμμα πρώτα όλα τα μαθήματα του πρώτου καθηγητή, στην συνέχεια του δεύτερου κ.ο.κ., όπως φαίνεται και στο διάγραμμα ροής (Σχήμα 8) στη συνάρτηση εκτελούνται και οι συναρτήσεις class_lessons() και lab_lessons(). Η class_lessons() στην περίπτωση που το μάθημα γίνεται σε αίθουσα και η lab_lessons() στην περίπτωση που το μάθημα γίνεται σε εργαστήριο.

programB: Καλώντας την συνάρτηση programB(), η εφαρμογή επιχειρεί να καταχωρήσει στο ωρολόγιο πρόγραμμα, το πρώτο μάθημα του πρώτου καθηγητή, στην συνέχεια το πρώτο μάθημα του δεύτερου καθηγητή κ.ο.κ., όπως φαίνεται και στο διάγραμμα ροής (Σχήμα 9) στη συνάρτηση εκτελούνται και οι συναρτήσεις class_lessons() και lab_lessons(). Η class_lessons() στην περίπτωση που το μάθημα γίνεται σε αίθουσα και η lab_lessons() στην περίπτωση που το μάθημα γίνεται σε εργαστήριο.




**Σχήμα 7:** Διάγραμμα ροής συνάρτησης programA()

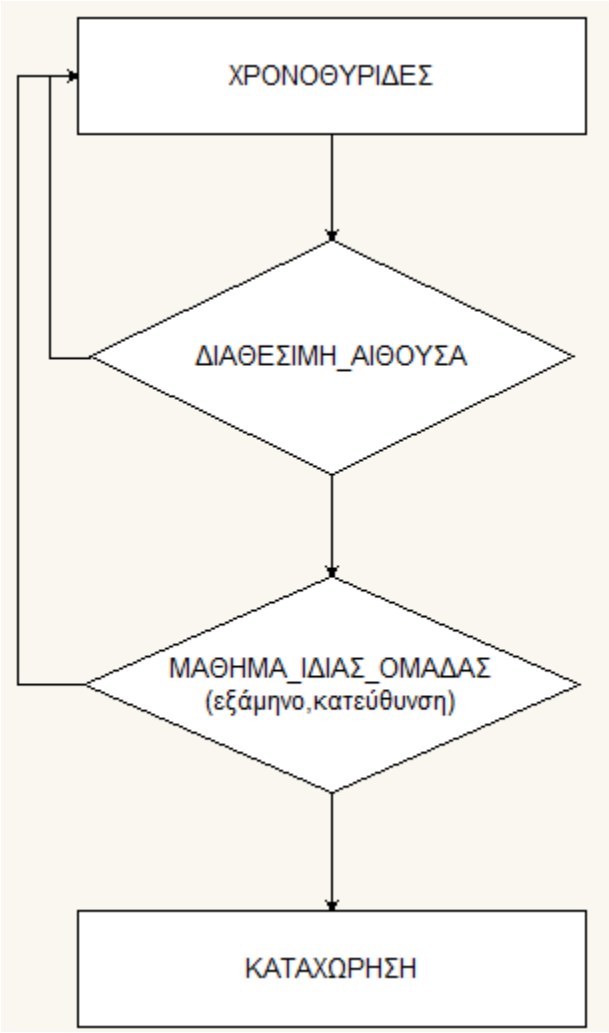



**Σχήμα 8:** Διάγραμμα ροής συνάρτησης programB()

class_lessons()/lab_lessons() (Σχήμα 9): Όπως ήδη αναφέραμε οι συναρτήσεις class_lessons() και lab_lessons() καλούνται από τις συναρτήσεις programA και programB. Ας δούμε τώρα και τον λόγο για τον οποίο εκτελούνται. Σ' αυτό το κομμάτι του προγράμματος ελέγχονται οι παρακάτω περιορισμοί που είναι απαραίτητοι για την δημιουργία του ωρολογίου προγράμματος :

- Έλεγχος διαθεσιμότητας αίθουσας για τις διαθέσιμες χρονοθυρίδες.
- Έλεγχος αν πραγματοποιείται μάθημα του ίδιου εξαμήνου.
- Έλεγχος αν το μάθημα είναι κατεύθυνσης.

Τέλος, πραγματοποιείται καταχώρηση αποτελεσμάτων στους πίνακες, program_professors_classes, program_classes, program_semester, program_course_classes και program_direction.




**Σχήμα 9:** Διάγραμμα ροής συνάρτησης class_lessons()

***Βήμα 5 από 5.Εμφάνιση ωρολογίου προγράμματος σε αρχεία .txt***

Στην συνάρτηση main τα αποτελέσματα που προκύπτουν από την εφαρμογή, δηλαδή το εβδομαδιαίο πρόγραμμα, εμφανίζονται στα εξής αρχεία:

*ΑΙΘΟΥΣΑ_1.txt*: Αποθηκεύονται τα μαθήματα που καταχωρήθηκαν στην ΑΙΘΟΥΣΑ_1. Επιπλέον, δημιουργούνται αντίστοιχα αρχεία και για τις υπόλοιπες αίθουσες.

*ΕΡΓΑΣΤΗΡΙΟ_1.txt:* Αποθηκεύονται τα μαθήματα που καταχωρήθηκαν στο ΕΡΓΑΣΤΗΡΙΟ_1. Επιπλέον, δημιουργούνται αντίστοιχα αρχεία και για τα υπόλοιπα εργαστήρια.

*ΕΞΑΜΗΝΟ_1.txt:* Αποθηκεύονται τα μαθήματα του πρώτου εξαμήνου. Αντίστοιχα αρχεία δημιουργούνται και για τα υπόλοιπα εξάμηνα.

*program_for_each_professor.txt:* Αποθηκεύονται τα μαθήματα που καταχωρήθηκαν για κάθε καθηγητή.

*missing.txt:* Δημιουργείται αν δεν έχει γίνει ανάθεση όλων των ωρών διδασκαλίας. Εμφανίζεται π.χ. το μήνυμα “Δεν καταχωρήθηκαν 3 ώρες του ΚΑΘΗΓΗΤΗ_1”.

Στην περίπτωση που ο χρήστης θελήσει να εκτελεστούν όλοι οι αλγόριθμοι τα αποτελέσματα θα εμφανιστούν στον φάκελο all_heuristics, ο οποίος θα περιέχει 13 υποφακέλους, όσοι δηλαδή είναι και οι ευρετικοί αλγόριθμοι. Για κάθε ευρετικό αλγόριθμο, ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά καθηγητή, ανά αίθουσα, ανά εργαστήριο, ανά εξάμηνο καθώς και σε πόσες ώρες καθηγητών δεν έχει γίνει ανάθεση. Επιπλέον, στο φάκελο all_heuristics δημιουργείται και ένα αρχείο, το οποίο εμφανίζει το ποσοστό ανάθεσης και τον χρόνο εκτέλεσης κάθε αλγορίθμου.

Στο επόμενο κεφάλαιο, γίνεται καταγραφή των αποτελεσμάτων για τους αλγόριθμους, που αναλύσαμε στην παρούσα διπλωματική, ύστερα από προσομοιώσεις που πραγματοποιήθηκαν.

# *ΚΕΦΑΛΑΙΟ 5*

## *ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ*

Σ' αυτό το κεφάλαιο θα αξιολογηθούν οι αλγόριθμοι που χρησιμοποιήθηκαν για την επίλυση του προβλήματος, υπολογίζοντας τους χρόνους εκτέλεσης και το ποσοστό επιτυχίας.

Οι χρόνοι εκτέλεσης υπολογίστηκαν με χρήση τις συνάρτησης clock(), στην συνάρτηση της εφαρμογής με όνομα all_heuristics. Ακριβώς, πριν την εκτέλεσης του κάθε ευρετικού αλγορίθμου τοποθετήσαμε την μεταβλητή time_start=clock() και αμέσως μετά την εκτέλεσή του την μεταβλητή time_end=clock(). Επομένως, για την εύρεση του χρόνου εκτέλεσης κάθε ευρετικού αλγορίθμου αφαιρέσαμε αυτές τις 2 μεταβλητές, δηλαδή:

```
heurictics_time[i]=time_end-time_start
```

Οι γραμμές κώδικα για τον πρώτο ευρετικό αλγόριθμο είναι οι εξής:

```
time_start=clock();
heuristicA_ProgramB();
time_end=clock();
heurictics_time[1]=time_end-time_start;
table_delete();
```

Το ποσοστό ανάθεσης υπολογίστηκε στην συνάρτηση all_heuristics. Για να το υπολογίσουμε χρειάστηκε να μετρήσουμε προηγουμένως, τις συνολικές ώρες διδασκαλίας (total_hours_of_teaching) του πανεπιστημίου και τις ώρες, οι οποίες δεν καταχωρήθηκαν στο ωρολόγιο πρόγραμμα, για κάθε ευρετικό αλγόριθμο (total_missing[i]). Τα ποσοστά ανάθεσης καταχωρήθηκαν στον πίνακα:

**in[i]**=(total_hours_of_teaching-total_missing[i])*100/total_hours_of_teaching

Οι χρόνοι εκτέλεσης και τα ποσοστά ανάθεσης διεξάγονται στο αρχείο time_and_missing.txt, το οποίο βρίσκεται στον φάκελο all_heuristics. Παρακάτω φαίνονται οι αντίστοιχες γραμμές κώδικα:

```
outfile<<setw(width)<<"Χρόνος εκτέλεσης (msec)";
for(i=0;i<13;i++)   outfile<<setw(width)<<heurictics_time[i]<<"|";;

outfile<<"\n";
outfile<<setw(width)<<"Ανάθεση (%)";

for(i=0;i<13;i++)
{
in[i]= (total_hours_of_teaching-
total_missing[i])*100/total_hours_of_teaching;
outfile<<setw(width)<<setiosflags(ios::fixed)<<setprecision(2)<<in[i]<<"|";
}
```

Οι συνολικές χρονοθυρίδες για κάθε πρόβλημα δίνονται από τον τύπο: 55*(πλήθος_αιθουσων+πλήθος_εργαστηρίων). Οι χρονοθυρίδες που αντιστοιχούν στα μαθήματα θεωρίας και στα μαθήματα εργαστηρίου είναι αντίστοιχα 55*πλήθος_αιθουσων και 55*πλήθος_εργαστηρίων. Κάθε φορά θα υπολογίζουμε τις χρονοθυρίδες ανάλογα με τα δεδομένα του προβλήματος, για να γνωρίζουμε εξ αρχής αν το πρόβλημα μπορεί να είναι επιλύσιμο. Όσες περισσότερες χρονοθυρίδες έχουμε στη διάθεση μας, συγκριτικά με τις ώρες διδασκαλίας, αυξάνεται ανάλογα και ο βαθμός επίλυσης του προβλήματος. Για παράδειγμα αν έχουμε 1 αίθουσα και 60 ώρες

διδασκαλίας, οι χρονοθυρίδες είναι 55 άρα είναι αδύνατο να γίνει ανάθεση όλων των μαθημάτων, εφόσον χρονοθυρίδες< ώρες_διδασκαλίας.

Ο βαθμός επίλυσης του προβλήματος εξαρτάται και από τις διαθέσιμες ώρες διδασκαλίας των καθηγητών. Αν οι συνολικές διαθέσιμες ώρες είναι ίσες με τις συνολικές ώρες διδασκαλίας το πρόβλημα είναι οριακά επιλύσιμο, ενώ αν είναι λιγότερες το πρόβλημα είναι αδύνατο να λυθεί. Επομένως, μπορούμε να αυξήσουμε τον βαθμό επίλυσης με δύο τρόπους, είτε αυξάνοντας τους διαθέσιμους πόρους είτε εισάγοντας επιπλέον διαθέσιμες ώρες διδασκαλίας.

Η εφαρμογή εκτελέστηκε σε υπολογιστή με επεξεργαστή Intel® Pentium® Dual CPU T3200 2GHz, εγκατεστημένη μνήμη (RAM) στα 3GB, 64-bit λειτουργικό σύστημα και έκδοση Windows 7 Professional 2009.

Για κάθε ένα από τα 3 είδη προβλημάτων (εύκολο, μέτριο, δύσκολο) που ακολουθούν, έγιναν 15 προσομοιώσεις για κάθε αλγόριθμο, με 15 διαφορετικές τιμές εισόδου κάθε φορά και υπολογίσθηκαν οι μέσοι όροι. Κάθε προσομοίωση εκτελέστηκε 2 φορές, για επιβεβαίωση αποτελεσμάτων. Στα πειραματικά δεδομένα, δεν υπάρχουν κατευθύνσεις, λόγω του ότι δεν υπάρχουν κατευθύνσεις στο τμήμα μας.

## *5.1 Εύκολο Πρόβλημα*

Το πρώτο πρόβλημα είναι το πιο εύκολο για να λυθεί. Στον Πίνακα 4 φαίνονται τα δεδομένα του προβλήματος.

| | |
|---|---|
| Αίθουσες | 2 |
| Εργαστήρια | 1 |
| Καθηγητές | 10 |
| Ώρες διδασκαλίας | 42 |
| Διαθέσιμες ώρες καθηγητών | 42*2=84 |

**Πίνακας 4:** Δεδομένα εύκολου προβλήματος

Το πλήθος των αιθουσών είναι 2, των εργαστηρίων 1 και ο αριθμός των καθηγητών 10. Οι ώρες διδασκαλίας ορίσαμε να είναι 42 και οι διαθέσιμες ώρες καθηγητών να είναι διπλάσιες από τις συνολικές ώρες διδασκαλίας (σε αίθουσα και εργαστήριο).

Οι διαθέσιμες χρονοθυρίδες διδασκαλίας είναι 55*(2+1)=165.

Άρα, η ανάθεση όλων των μαθημάτων στο πρόγραμμα είναι πιθανή.

Στον Παράρτημα 1 και 2 φαίνονται τα αποτελέσματα των προσομοιώσεων και στον Πίνακα 5 ο μέσος όρος αυτών. Στο Παράρτημα 7, φαίνονται ενδεικτικά τα δεδομένα καταχώρησης, μιας απο τις προσομοιώσεις.

Ο μέσος χρόνος εκτέλεσης της συνάρτησης sorting(), που περιγράψαμε στην παράγραφο 4.3, είναι 2,66 msec.

| | Χρόνος εκτέλεσης(msec) | Ανάθεση (%) |
|---|---|---|
| heuristicA_ProgramA | 104.13 | 95.94 |
| heuristicA_ProgramB | 82.8 | 98.21 |
| heuristicB_ProgramA | 73.27 | 95.12 |
| heuristicB_ProgramB | 65.93 | 98.54 |
| heuristicC_ProgramA | 30.13 | 94.96 |
| heuristicC_ProgramB | 24.73 | 98.21 |
| heuristicD_ProgramA | 25.53 | 94.47 |
| heuristicD_ProgramB | 24.20 | 97.07 |
| heuristicE_ProgramA | 26.47 | 93.98 |
| heuristicE_ProgramB | 30.80 | 96.91 |
| heuristicF_ProgramA | 26.40 | 95.45 |
| heuristicF_ProgramB | 26.80 | 97.07 |
| random_heuristic | 1505,73 | 98.05 |

**Πίνακας 5:** Μέσος όρος αποτελεσμάτων εύκολου προβλήματος

**Παρατηρήσεις αποτελεσμάτων**

- Ο χρόνος εκτέλεσης του ευρετικού αλγόριθμου random_heuristic παρατηρούμε ότι είναι αρκετά μεγαλύτερος (1505,73msec), συγκριτικά με τους υπόλοιπους. Αυτό συμβαίνει γιατί η συνάρτηση random_heuristic() καλείται 300 φορές ή μέχρι να καταχωρηθούν όλα τα μαθήματα στο ωρολόγιο πρόγραμμα.

- Οι ευρετικοί αλγόριθμοι heuristicA_ProgramA, heuristicA_ProgramB, heuristicB_ProgramA και heuristicB_ProgramB εκτελούνται σε λίγο μεγαλύτερο χρόνο από τους υπόλοιπους. Αυτό οφείλεται στο γεγονός ότι γίνονται περισσότερες μεταθέσεις σε αυτές τις συναρτήσεις.

- Ελάχιστα μεγαλύτερη επιλυσιμότητα έχουν οι αλγόριθμοι heuristicA_ProgramB και heuristicC_ProgramB. Δε μπορούμε να βγάλουμε το συμπέρασμα ότι είναι και αποδοτικότεροι από τους υπόλοιπους λόγω τις μικρής τους διαφοράς στο ποσοστό ανάθεσης. Αν δούμε τα αποτελέσματα του Παραρτήματος 1, παρατηρούμε ότι κάθε είσοδος δεδομένων, μπορεί να επιλυθεί 100% με διαφορετικό αλγόριθμο. Επομένως, κάθε περίπτωση είναι ξεχωριστή και εξαρτάται κάθε φορά από τα δεδομένα του προβλήματος.

## *5.2 Μέτριο πρόβλημα*

Το δεύτερο πρόβλημα είναι μέτριας δυσκολίας. Στον Πίνακα 6 φαίνονται τα δεδομένα του προβλήματος.

| | |
|---|---|
| Αίθουσες | 4 |
| Εργαστήρια | 2 |
| Καθηγητές | 25 |
| Ώρες διδασκαλίας | 100 |
| Διαθέσιμες ώρες καθηγητών | 100*2=200, 220, 270 |

**Πίνακας 6:** Δεδομένα μέτριου προβλήματος

Το πλήθος των αιθουσών το αυξήσαμε σε 4, των εργαστηρίων σε 2 και των καθηγητών σε 25. Οι ώρες διδασκαλίας ορίσαμε να είναι 100. Οι διαθέσιμες ώρες καθηγητών, μέχρι και την 5η προσομοίωση (Παράρτημα 3) διαλέξαμε, όπως και στο εύκολο πρόβλημα να είναι διπλάσιες από τις συνολικές ώρες διδασκαλίας, δηλαδή 200. Μέχρι και την 10η προσομοίωση προσθέσαμε άλλες 20 διαθέσιμες ώρες, δηλαδή έχουμε συνολικά 220 διαθέσιμες ώρες καθηγητών. Τέλος, για τις υπόλοιπες 5, εισάγαμε συνολικά 270 διαθέσιμες ώρες. Καταλήξαμε σ' αυτήν την απόφαση λόγω του ότι δεν υπήρχε σε κανέναν από τους 13 αλγόριθμους 100% ανάθεση μαθημάτων.

Οι διαθέσιμες χρονοθυρίδες για τα μαθήματα θεωρίας είναι 55*(4+2)=330.

Άρα, η ανάθεση όλων των μαθημάτων στο πρόγραμμα είναι πιθανή.

Στον Παράρτημα 3 και 4 φαίνονται τα αποτελέσματα των προσομοιώσεων και στον Πίνακα 7 ο μέσος όρος αυτών.

Ο μέσος χρόνος εκτέλεσης της συνάρτησης sorting(), είναι 4.4 msec.

| | Χρόνος εκτέλεσης (msec) | Ανάθεση (%) |
|---|---|---|
| heuristicA_ProgramA | 249.4 | 93.31 |
| heuristicA_ProgramB | 67.3 | 95.57 |
| heuristicB_ProgramA | 47.3 | 93.29 |
| heuristicB_ProgramB | 52 | 94.24 |
| heuristicC_ProgramA | 45.6 | 92.70 |
| heuristicC_ProgramB | 49 | 94.57 |
| heuristicD_ProgramA | 44.6 | 93.10 |
| heuristicD_ProgramB | 44.1 | 94.63 |
| heuristicE_ProgramA | 44.4 | 92.97 |
| heuristicE_ProgramB | 46 | 94.97 |
| heuristicF_ProgramA | 44.5 | 92.49 |
| heuristicF_ProgramB | 50 | 94 |
| random_heuristic | 14393 | 88.40 |

**Πίνακας 7:** Μέσος όρος αποτελεσμάτων μέτριου προβλήματος

**Παρατηρήσεις αποτελεσμάτων**

- Οι χρόνοι εκτέλεσης των ευρετικών αλγορίθμων αυξήθηκαν, σε σχέση με το εύκολο πρόβλημα, όπως ήταν αναμενόμενο λόγω του μεγαλύτερου πλήθους δεδομένων.
- Στο μέτριο πρόβλημα ελάχιστα μεγαλύτερο ποσοστό καταχώρησης μαθημάτων έχει ο ευρετικός αλγόριθμος heuristicA_ProgramB.

## *5.3 Δύσκολο πρόβλημα*

Το τρίτο και τελευταίο πρόβλημα είναι δύσκολης επιλυσιμότητας. Στον πίνακα φαίνονται τα δεδομένα του προβλήματος.

| | |
|---|---|
| Αίθουσες | 6 |
| Εργαστήρια | 2 |
| Καθηγητές | 40 |
| Ώρες διδασκαλίας | 160 |
| Διαθέσιμες ώρες καθηγητών | 160*2=320, 360, 400, 480, 580, 640 |

**Πίνακας 8:** Δεδομένα δύσκολου προβλήματος

Λόγω του μεγάλου πλήθος δεδομένων αυτού του προβλήματος δημιουργήθηκε πρόβλημα στην συνάρτηση rand(). Επομένως, δεν εκτελέσαμε τον ευρετικό αλγόριθμο random_heuristic.

Το πλήθος των αιθουσών το αυξήσαμε σε 6, των εργαστηρίων το αφήσαμε 2, όπως και στο μέτριο πρόβλημα, και των καθηγητών το αυξήσαμε σε 40. Οι ώρες διδασκαλίας που εισάγαμε είναι 160. Οι διαθέσιμες ώρες καθηγητών, μέχρι και την 5η προσομοίωση (Παράρτημα 5) διαλέξαμε, όπως και στα προηγούμενο πρόβλημα να είναι διπλάσιες από τις συνολικές ώρες διδασκαλίας, δηλαδή 320. Στις προσομοιώσεις 6 και 7 προσθέσαμε άλλες 40 διαθέσιμες ώρες, δηλαδή έχουμε συνολικά 360 διαθέσιμες ώρες καθηγητών. Στις 8 μέχρι και 11 έχουμε 400 διαθέσιμες ώρες. Τέλος, για τις υπόλοιπες 4 εισάγαμε τριπλάσιο αριθμό από τις ώρες διδασκαλίας, δηλαδή 480. Λόγω του αυξημένου πλήθους δεδομένων, είναι δύσκολο να δημιουργηθεί ωρολόγιο πρόγραμμα με 100% ανάθεση μαθημάτων, γι' αυτό κρίθηκε απαραίτητο να αυξηθούν οι διαθέσιμες ώρες των καθηγητών.

Οι διαθέσιμες χρονοθυρίδες είναι 55*(6+2)=440.

Άρα, η ανάθεση όλων των μαθημάτων στο πρόγραμμα είναι πιθανή.

Στον Παράρτημα 5 και 6 φαίνονται τα αποτελέσματα των προσομοιώσεων και στον Πίνακα 9 ο μέσος όρος αυτών.

Ο μέσος χρόνος εκτέλεσης της συνάρτησης sorting(), είναι 7.6 msec

| | Χρόνος εκτέλεσης (msec) | Ανάθεση (%) |
|---|---|---|
| heuristicA_ProgramA | 293 | 88.99 |
| heuristicA_ProgramB | 92.4 | 88.89 |
| heuristicB_ProgramA | 84.93 | 87.99 |
| heuristicB_ProgramB | 87.6 | 89.42 |
| heuristicC_ProgramA | 92 | 91.80 |
| heuristicC_ProgramB | 92.13 | 89.98 |
| heuristicD_ProgramA | 90.73 | 92 |
| heuristicD_ProgramB | 90.67 | 90.33 |
| heuristicE_ProgramA | 86 | 91 |
| heuristicE_ProgramB | 93.33 | 90.28 |
| heuristicF_ProgramA | 93.47 | 89.83 |
| heuristicF_ProgramB | 99 | 89.76 |

**Πίνακας 9:** Μέσος όρος αποτελεσμάτων δύσκολου προβλήματος

**Παρατηρήσεις αποτελεσμάτων**

- Παρατηρούμε ότι οι χρόνοι εκτέλεσης αυξήθηκαν, εφόσον έχουμε ακόμη μεγαλύτερο πλήθος δεδομένων.

- Το ποσοστό ανάθεσης μειώθηκε αισθητά. Οι ώρες διδασκαλίας είναι 160, όπως ήδη αναφέραμε, επομένως είναι δύσκολο να δημιουργηθεί ωρολόγιο πρόγραμμα, στο οποίο η ανάθεση να φτάσει το 100 %.

  Στο Παράρτημα 5, στο οποίο φαίνονται αναλυτικά τα αποτελέσματα και των 15 προσομοιώσεων, κανένας ευρετικός αλγόριθμος δεν κατάφερε να λύσει το πρόβλημα στο 100%. Για το λόγο αυτό θα εκτελέσαμε 5 επιπλέον προσομοιώσεις, αυξάνοντας το πλήθος των διαθέσιμων ωρών.

- Προσομοίωση 16: 480+40=520 διαθέσιμες ώρες, δεν έχουμε αγγίξει το 100%.

- Προσομοίωση_17: 480+100=580 διαθέσιμες ώρες, καμία σημαντική βελτίωση.

- Προσομοίωση_18: Ορίσαμε συνολικά 640 (4*160) διαθέσιμες ώρες, δηλαδή τετραπλάσιες από τις ώρες διδασκαλίας. Όπως φαίνεται και στο Παράρτημα 5, παρατηρούμε σημαντική βελτίωση στα αποτελέσματα όλων των ευρετικών αλγορίθμων, η ανάθεση αγγίζει μέχρι και το 97.99%.

- Προσομοίωση_19: Με τα ίδια δεδομένα, που εκτελέσαμε στην προσομοίωση_18, εκτελούμε και την 19, με την διαφορά ότι αλλάξαμε τις τιμές εισόδου. Διαπιστώσαμε ότι πετύχαμε καταχώρηση όλων των μαθημάτων στο ωρολόγιο πρόγραμμα, κατά την εκτέλεση της συνάρτησης heuristicA_ProgramA.

Στο επόμενο και τελευταίο κεφάλαιο, καταλήγουμε σε κάποια συμπεράσματα, στηριζόμενοι στα αποτελέσματα των προσομοιώσεων, και προτείνουμε ιδέες για βελτίωση και επέκταση της παρούσας εφαρμογής.

# *ΚΕΦΑΛΑΙΟ 6*

## *ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΟΠΤΙΚΕΣ*

Στόχος της παρούσας μελέτης ήταν η υλοποίηση ενός ωρολογίου προγράμματος για εφαρμογή από το πανεπιστήμιο δυτικής Μακεδονίας και πιο συγκεκριμένα στο τμήμα μηχανικών πληροφορικής και τηλεπικοινωνιών. Το πρόγραμμα υλοποιήθηκε σε γλωσσά προγραμματισμού C++ μέσω του προγράμματος Microsoft Visual Studio 2010. Ύστερα από μετρήσεις του προγράμματος καταλήξαμε ότι το πρόγραμμα είναι σε θέση να καλύψει πλήρως τις απαιτήσεις μεσαίας κλίμακας πλήθους καθηγητών. Σε μετρήσεις που έγιναν για μεγάλο αριθμό καθηγητών παρατηρήθηκε ότι για την πλήρη καταχώρηση των μαθημάτων με βάση τις δηλωθείσες ώρες των καθηγητών, απαιτείται οι ώρες αυτές να είναι τουλάχιστον οι τετραπλάσιες σε σύγκριση με τις ώρες διδασκαλίας. Αυτή η δυσκολία οφείλεται στο πλήθος των περιορισμών για την σχεδίαση του προγράμματος καθώς και στην επιβάρυνση του ωρολογίου προγράμματος με τη

κάλυψη των αρχικών ωρών. Συνεπώς η προσθήκη και καταχώρηση επιπλέον διαθέσιμων ωρών καθίσταται απαραίτητη.

Η αναπτυχθείσα εφαρμογή συνίσταται για την διεξαγωγή ενός εβδομαδιαίου προγράμματος όταν δεν υπάρχουν ιδιαίτερες προτιμήσεις ωρών από τους καθηγητές ή έστω όταν διατίθεται μια πληθώρα εναλλακτικών διαθέσιμων ωρών.

Μια πρόταση για μελλοντική επέκταση αποτελεί η ανάπτυξη μιας συνάρτησης στο πρόγραμμα που θα βασίζεται στις ΜΗ διαθέσιμες ώρες των καθηγητών. Οι καθηγητές θα καλούνται να δηλώσουν τις ώρες για τις οποίες ΔΕ μπορούν να κάνουν μάθημα, δίνοντας την δυνατότητα στο πρόγραμμα να επιλέξει ανάμεσα σε μεγαλύτερο πλήθος ωρών. Ωστόσο, αυτό δεν θα έχει ουσία, αν οι μη διαθέσιμες ώρες των καθηγητών υπερβαίνουν τις διαθέσιμες.

Άλλη μια πρόταση είναι η προσθήκη μιας συνάρτηση για την κάλυψη αναπληρωματικών ωρών σε περιπτώσεις έκτακτων περιστατικών όπως αναβολή μαθημάτων για οποιουσδήποτε λόγους. Η εφαρμογή θα καλείται κάθε φορά που απαιτείται αναπλήρωση μαθήματος, ο χρήστης θα εισάγει τις ώρες αναπλήρωσης, το εξάμηνο στο όποιο ανήκει το μάθημα καθώς και το όνομα του καθηγητή. Θα υπάρχει μία συνάρτηση η οποία θα διαβάζει το υπάρχον ωρολόγιο πρόγραμμα από το ήδη καταχωρημένο αρχείο, το όποιο θα έχει διεξήχθη από την παρούσα εφαρμογή και θα το αποθηκεύει σε πίνακες. Κατόπιν, θα βρίσκει κενές ώρες στο πρόγραμμα, τηρώντας τους αυστηρούς περιορισμούς, όπως το να μην πραγματοποιείται μάθημα του ίδιου εξαμήνου. Το ενδεχόμενο να τηρηθούν και οι εύκολοι περιορισμοί, όπως π.χ. ο καθηγητής να εισάγει και ποιες ώρες έχει στην διάθεση του, καλύτερο θα ήταν να αποφευχθεί, διότι έτσι θα αυξηθεί σημαντικά η δυσκολία εύρεσης ωρών, με κίνδυνο να μην βρεθεί λύση.

Καλή ιδέα θα ήταν επίσης, η δημιουργία ενός αλγορίθμου, που θα στηρίζεται σε διαφορετικούς περιορισμούς, όπως η δημιουργία ενός ωρολογίου προγράμματος που να μην υπερβαίνει τέσσερις συνεχόμενες ώρες μαθήματος ημερησίως, αλλά να παρέχει την δυνατότητα ενδιάμεσων κενών ωρών τόσο για τους φοιτητές, όσο και για τους καθηγητές.

Κάποια μαθήματα, για εκπαιδευτικούς λόγους, ίσως να πρέπει να προηγούνται από κάποια άλλα ώστε να λειτουργούν με εισαγωγικό τρόπο. Για παράδειγμα, το μάθημα “Συστήματα επικοινωνιών Θεωρία”, καλό θα ήταν να γίνεται πριν το μάθημα “Συστήματα επικοινωνιών Εργαστήριο”.

Εύστοχο θα ήταν η εφαρμογή να εκτελεί και έναν αλγόριθμο, ο οποίος θα διεξάγει το πρόγραμμα της εξεταστικής. Ο αλγόριθμος θα διαβάζει τα μαθήματα που εξετάζονται και θα συγκρίνει ανάλογα με τους εκάστοτε περιορισμούς π.χ. μαθήματα του ίδιου

εξαμήνου να εξετάζονται τουλάχιστον ανά δυο μέρες κτλ και θα καταχωρεί την ώρα εξέτασης.

Στην παρούσα διπλωματική, δε γίνεται διάκριση ανάμεσα σε μόνιμους και συμβασιούχους καθηγητές. Θα μπορούσαμε, στο μέλλον, να καταχωρούμε στην εφαρμογή και την βαθμίδα στην οποία ανήκει ο κάθε εκπαιδευτικός και να δίνεται προτεραιότητα καταχώρησης, στον εκπαιδευτικό με την μεγαλύτερη βαθμίδα. Για παράδειγμα, τα μαθήματα των καθηγητών να καταχωρούνται πρώτα, να ακολουθούν των αναπληρωτών κ.ο.κ. Αυτό μπορεί να συμβεί με μία συνάρτηση που θα υλοποιεί την ανάλογη ταξινόμηση.

Τέλος, άλλη μια καλή πρόταση για πιο προσιτή και οικεία εφαρμογή ως προς τον χρήστη είναι η υλοποίηση της εφαρμογής σε γραφικό περιβάλλον. Αυτό μπορεί να επιτευχθεί με την σύνδεση του κώδικα σε φόρμες μέσω του προγράμματος Microsoft Visual Studio Forms.

# *ΒΙΒΛΙΟΓΡΑΦΙΑ*

[1] APPLICATIONS TO TIMETABLING, http://www.cs.nott.ac.uk, Ιούνιος 2012
[2] Bardadym, V.A. Computer Aided School and University Timetabling: The New Wave, in E. Burke and P. Ross (Ends), The Practice Theory of Automated Timetambling I, 1996.
[3] E. Burke and P. Ross, The Practice and Theory of Automated Timetabling I, 1996.
[4] Powell, D.J.A. Welsh and M. B. An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems. 1967.
[5] D. de. Werra, Graphs, Hyper-graphs and Timetabling, Methods of Operations Research, 1985.
[6] E. K. Burke, K. S. Jackson and R. F. Weare. Automated Timetabling,The State of the Art, 1997.
[7] M. W. Laporte, G. Carter, Recent Developments in Practical Course Timetabling, in E. Burke and M. Carter (Ends), Practice and Theory of Automated Timetabling II (PATAT, 1997, Toronto), 1998.
[8] Arnaldo Vieira Moura, Rafael Augusto Scaraficci, A GRASP strategy for a more constrained School Timetabling Problem, 2010.
[9] A. Wren, Scheduling, timetabling and rostering — A special relationship?, 1996.
[10] Stuart Russell, Peter Norvig, Artificial Intelligence A Modern Approach, Second edition.
[11] Kim Marriott and Peter J. Stuckey, Programming with Constraints, An Introduction, 1998.
[12] H. M. Deitel, P. J. Deitel, C++ Προγραμματισμός, Τέταρτη έκδοση
[13] Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2010.
[14] Κουμπαράκης, Μ. Τεχνητή νοημοσύνη, ΥΣ02.
[15] Τεχνιτή Νοημοσύνη, http://cgi.di.uoa.gr, Ιούνιος 2012.
[16] Una-May O' Reilly, Tina Yu,Rick Riolo,Bill Worzel, Genetic Programming, Theory and Practice II, 2005.
[17] E. Falkenauer and Bouffouix, "A genetic algorithm for the job-shop",Proceedings of the IEEE International Conference on Robotics and Automation, 1991.
[18] S. Kobayashi, M Yamamura. An efficient genetic algorithm for jobshop scheduling problems", Proceedings of the Sixth International Con-ference onGenetic Algorithms, Morgan Kaufmann, San Francisco, 1995.
[19] Shi, G. "A genetic algorithm applied to a classic job-shop scheduling problem",International Journal of Systems Science. 1997.

[20] Πληροφορημένη αναζήτηση και εξερεύνηση, http://aima.uom.gr, Ιούνιος 2012
[21] Anant Singh Jain, Sheik Mee. A state-of-the-art review of job-shop scheduling techniques, 1998.
[22] N. Raman, F.Brian Talbot. "The job shop tardiness problem:A decomposition approach". North-Holland : European Journal of Operational Research, 1993.
[23] P. Stamatopoulos, E. Viglas, S. Karaboyas. "Nearly optimum timetable construction through CLP and intelligent search", Int. Journal on Artificial Intelligence Tools, 1998.
[24] Ferland, J. "Generalized assignment-type problems: A powerful modeling scheme". 2nd Int. Conf. on the Practice and Theory of Autamated Timetabling (PATAT), 1997.
[25] Edmund Burke, Peter Ross. Practice and Theory of Automated Tametabling, 1995.
[26] Horton, Ivor. Beginning Visual C++. 2010.
[27] Goltz, Hans-Joachim. Combined Automatic and Interactive Timetabling Using Constrain Logic Programming. 2000.
[28] Samuel Lukas, Arnold Aribowo,Milyandreana Muchri. “Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study: Universitas Pelita Harapan Timetable”.
[29] P. Causmaecker, Practice and Theory of Automated Timetabling IV. 2002.

# *ΠΑΡΑΡΤΗΜΑΤΑ*

| **ΑΝΑΘΕΣΗ** (%) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDPa | hDpB | hEpA | hEpB | hFpA | hFpB | HR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 100 | 100 | 92.68 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_2 | 82.93 | 92.68 | 92.68 | 100 | 82.93 | 92.68 | 82.93 | 92.68 | 82.93 | 92.68 | 82.93 | 92.68 | 100 |
| Προσομοίωση_3 | 90.24 | 95.12 | 100 | 95.12 | 90.24 | 95.12 | 90.24 | 95.12 | 82.93 | 92.68 | 92.68 | 87.80 | 100 |
| Προσομοίωση_4 | 100 | 100 | 92.68 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_5 | 100 | 100 | 90.24 | 90.24 | 100 | 100 | 82,93 | 92.68 | 82.93 | 92.68 | 100 | 100 | 82.93 |
| Προσομοίωση_6 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_7 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_8 | 82.93 | 92.68 | 92.68 | 100 | 82.93 | 92.68 | 82.93 | 92.68 | 82.93 | 92.68 | 82.93 | 92.68 | 100 |
| Προσομοίωση_9 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_10 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 95.12 | 100 | 95.12 | 95.12 | 95.12 | 100 |
| Προσομοίωση_11 | 100 | 100 | 85.37 | 92.68 | 92.68 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_12 | 100 | 100 | 92.68 | 100 | 92.68 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_13 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_14 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_15 | 82.93 | 92.68 | 87.80 | 100 | 82.93 | 92.68 | 78.05 | 87.80 | 78.05 | 87.80 | 78.05 | 87.80 | 87.80 |

**Παράρτημα 1:** Αναθέσεις (%) προσομοιώσεων αλγορίθμων για το εύκολο πρόβλημα

| **ΧΡΟΝΟΣ** (msec) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDpA | hDpB | hEpA | hEpB | hFpA | hFpB | HR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 185 | 88 | 101 | 50 | 25 | 21 | 22 | 25 | 25 | 113 | 26 | 25 | 4018 |
| Προσομοίωση_2 | 104 | 92 | 72 | 73 | 28 | 31 | 33 | 34 | 27 | 33 | 20 | 29 | 1446 |
| Προσομοίωση_3 | 115 | 104 | 109 | 54 | 21 | 22 | 22 | 24 | 66 | 26 | 29 | 27 | 23 |
| Προσομοίωση_4 | 90 | 107 | 84 | 47 | 23 | 23 | 21 | 23 | 22 | 21 | 25 | 25 | 22 |
| Προσομοίωση_5 | 87 | 85 | 65 | 64 | 21 | 22 | 22 | 21 | 20 | 20 | 23 | 24 | 6397 |
| Προσομοίωση_6 | 83 | 118 | 101 | 57 | 37 | 30 | 27 | 22 | 22 | 22 | 28 | 29 | 23 |
| Προσομοίωση_7 | 103 | 95 | 103 | 78 | 25 | 20 | 32 | 20 | 23 | 25 | 26 | 24 | 28 |
| Προσομοίωση_8 | 102 | 90 | 69 | 73 | 26 | 27 | 30 | 30 | 24 | 30 | 28 | 27 | 3446 |
| Προσομοίωση_9 | 98 | 64 | 103 | 63 | 23 | 25 | 28 | 23 | 25 | 24 | 29 | 27 | 23 |
| Προσομοίωση_10 | 81 | 59 | 32 | 48 | 25 | 26 | 27 | 25 | 25 | 25 | 27 | 28 | 27 |
| Προσομοίωση_11 | 104 | 100 | 76 | 59 | 23 | 24 | 22 | 22 | 22 | 22 | 24 | 25 | 22 |
| Προσομοίωση_12 | 120 | 33 | 30 | 92 | 53 | 23 | 23 | 22 | 22 | 21 | 27 | 27 | 22 |
| Προσομοίωση_13 | 82 | 58 | 36 | 75 | 25 | 29 | 27 | 24 | 24 | 24 | 27 | 30 | 27 |
| Προσομοίωση_14 | 102 | 61 | 71 | 89 | 45 | 25 | 24 | 26 | 28 | 33 | 33 | 29 | 27 |
| Προσομοίωση_15 | 106 | 88 | 47 | 67 | 52 | 23 | 23 | 22 | 22 | 23 | 24 | 26 | 7035 |

**Παράρτημα 2:** Χρόνοι εκτέλεσης (msec) προσομοιώσεων αλγορίθμων για το εύκολο πρόβλημα

| **ΑΝΑΘΕΣΗ** (%) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDPa | hDpB | hEpA | hEpB | hFpA | hFpB | HR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 87.90 | 91.92 | 87.88 | 90.91 | 89.90 | 92.92 | 89.90 | 91.92 | 89.90 | 91.92 | 91.92 | 91.92 | 92.93 |
| Προσομοίωση_2 | 86.87 | 93.94 | 90.91 | 93.94 | 86.87 | 93.94 | 86.87 | 93.94 | 88.89 | 95.96 | 86.87 | 93.94 | 84.85 |
| Προσομοίωση_3 | 90.10 | 92.08 | 88.12 | 91.09 | 90.10 | 92.08 | 90.10 | 92.08 | 92.10 | 92.08 | 89.11 | 92.08 | 88.12 |
| Προσομοίωση_4 | 92 | 92 | 92 | 90 | 94 | 92 | 92 | 94 | 94 | 96 | 92 | 94 | 90 |
| Προσομοίωση_5 | 87 | 89 | 96 | 90 | 89 | 89 | 91 | 95 | 91 | 95 | 89 | 93 | 85 |
| Προσομοίωση_6 | 100 | 96.77 | 96.77 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Προσομοίωση_7 | 87 | 92 | 92 | 92 | 89 | 92 | 91 | 96 | 89 | 96 | 93 | 96 | 81 |
| Προσομοίωση_8 | 92 | 92 | 90 | 92 | 94 | 92 | 94 | 94 | 92 | 96 | 94 | 89 | 89 |
| Προσομοίωση_9 | 93 | 98 | 94 | 92 | 90 | 95 | 90 | 93 | 88 | 93 | 90 | 93 | 87 |
| Προσομοίωση_10 | 98 | 100 | 91 | 94 | 97 | 97 | 97 | 95 | 95 | 95 | 97 | 95 | 84 |
| Προσομοίωση_11 | 94 | 100 | 93 | 97 | 87 | 97 | 91 | 95 | 93 | 96 | 91 | 93 | 86 |
| Προσομοίωση_12 | 95.92 | 100 | 94.90 | 96.94 | 92.86 | 96.94 | 92.86 | 92.86 | 92.86 | 90.86 | 90.82 | 92.86 | 91.84 |
| Προσομοίωση_13 | 95.92 | 97.86 | 97.86 | 97.86 | 92.86 | 94.90 | 90.82 | 90.82 | 90.82 | 90.82 | 89.80 | 90.82 | 81.63 |
| Προσομοίωση_14 | 100 | 100 | 96.91 | 97.94 | 97.94 | 95.88 | 100 | 100 | 100 | 100 | 97.94 | 100 | 93.81 |
| Προσομοίωση_15 | 100 | 97.94 | 97.94 | 97.94 | 100 | 97.94 | 100 | 95.88 | 97.94 | 95.88 | 94.85 | 95.88 | 90.72 |

**Παράρτημα 3:** Αναθέσεις (%) προσομοιώσεων αλγορίθμων για το μέτριο πρόβλημα

| ΧΡΟΝΟΣ (msec) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDpA | hDpB | hEpA | hEpB | hFpA | hFpB | HR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 87 | 57 | 46 | 51 | 42 | 43 | 41 | 42 | 41 | 44 | 43 | 50 | 13121 |
| Προσομοίωση_2 | 232 | 98 | 40 | 40 | 40 | 44 | 39 | 41 | 44 | 41 | 40 | 56 | 12437 |
| Προσομοίωση_3 | 239 | 48 | 46 | 47 | 48 | 48 | 45 | 42 | 42 | 46 | 46 | 53 | 12830 |
| Προσομοίωση_4 | 258 | 54 | 47 | 48 | 39 | 42 | 52 | 41 | 48 | 48 | 44 | 46 | 12334 |
| Προσομοίωση_5 | 219 | 60 | 42 | 43 | 40 | 44 | 40 | 45 | 49 | 44 | 40 | 49 | 13570 |
| Προσομοίωση_6 | 244 | 123 | 47 | 46 | 46 | 38 | 38 | 38 | 38 | 44 | 43 | 55 | 28820 |
| Προσομοίωση_7 | 249 | 75 | 49 | 47 | 44 | 44 | 39 | 42 | 40 | 44 | 39 | 46 | 12560 |
| Προσομοίωση_8 | 242 | 125 | 49 | 50 | 42 | 40 | 39 | 39 | 42 | 45 | 43 | 48 | 12980 |
| Προσομοίωση_9 | 251 | 53 | 43 | 50 | 45 | 55 | 44 | 44 | 41 | 42 | 41 | 47 | 13084 |
| Προσομοίωση_10 | 298 | 44 | 48 | 60 | 48 | 71 | 44 | 39 | 42 | 44 | 44 | 46 | 13336 |
| Προσομοίωση_11 | 291 | 45 | 52 | 59 | 47 | 73 | 46 | 42 | 44 | 45 | 45 | 47 | 12336 |
| Προσομοίωση_12 | 291 | 63 | 48 | 51 | 49 | 51 | 48 | 60 | 47 | 47 | 49 | 54 | 14558 |
| Προσομοίωση_13 | 268 | 58 | 48 | 86 | 48 | 47 | 47 | 45 | 46 | 47 | 50 | 49 | 14037 |
| Προσομοίωση_14 | 288 | 57 | 53 | 52 | 54 | 49 | 53 | 52 | 53 | 55 | 51 | 52 | 14949 |
| Προσομοίωση_15 | 284 | 50 | 51 | 50 | 52 | 47 | 54 | 50 | 49 | 55 | 50 | 52 | 14949 |

**Παράρτημα 4:** Χρόνοι εκτέλεσης (msec) προσομοιώσεων αλγορίθμων για το μέτριο πρόβλημα

| **ΑΝΑΘΕΣΗ** (%) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDpA | hDpB | hEpA | hEpB | hFpA | hFpB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 84.52 | 84.52 | 81.94 | 82.58 | 87.10 | 87.10 | 84.52 | 85.81 | 87.10 | 87.10 | 87.10 | 83.23 |
| Προσομοίωση_2 | 85.81 | 87.10 | 84.52 | 87.10 | 88.39 | 88.39 | 89.68 | 89.68 | 90.67 | 89.68 | 89.68 | 87.10 |
| Προσομοίωση_3 | 90.32 | 89.03 | 87.10 | 86.45 | 93.55 | 90.32 | 93.55 | 90.32 | 91.61 | 90.32 | 90.97 | 90.97 |
| Προσομοίωση_4 | 87.66 | 87.66 | 87.01 | 89.61 | 88.96 | 87.66 | 90.26 | 87.66 | 88.31 | 87.66 | 86.36 | 87.01 |
| Προσομοίωση_5 | 86.36 | 87.66 | 86.36 | 89.61 | 87.66 | 87.66 | 88.96 | 87.66 | 87.01 | 87.66 | 88.96 | 88.31 |
| Προσομοίωση_6 | 84.42 | 88.96 | 88.96 | 90.91 | 87.66 | 90.91 | 87.66 | 89.61 | 85.71 | 87.66 | 84.42 | 89.61 |
| Προσομοίωση_7 | 84.42 | 88.96 | 85.71 | 89.61 | 89.61 | 89.61 | 89.61 | 87.66 | 84.42 | 87.66 | 84.66 | 89.61 |
| Προσομοίωση_8 | 87.01 | 90.26 | 87.01 | 90.91 | 93.51 | 92.21 | 92.21 | 92.91 | 90.26 | 88.96 | 87.66 | 90.91 |
| Προσομοίωση_9 | 85.06 | 91.56 | 91.56 | 92.21 | 94.16 | 91.56 | 95.45 | 92.86 | 93.51 | 92.86 | 86.36 | 94.8 |
| Προσομοίωση_10 | 90.20 | 90.20 | 89.54 | 88.24 | 92.16 | 90.20 | 92.16 | 92.81 | 93.46 | 92.81 | 88.24 | 92.16 |
| Προσομοίωση_11 | 90.85 | 86.27 | 90.20 | 89.54 | 95.42 | 90.20 | 95.42 | 91.50 | 94.12 | 92.81 | 95.42 | 90.85 |
| Προσομοίωση_12 | 94.12 | 88.89 | 90.20 | 89.54 | 96.08 | 90.20 | 96.08 | 91.50 | 96.08 | 92.81 | 94.77 | 90.85 |
| Προσομοίωση_13 | 94.77 | 89.54 | 86.93 | 89.54 | 92.16 | 90.85 | 92.16 | 89.54 | 93.46 | 92.16 | 92.16 | 88.24 |
| Προσομοίωση_14 | 94.12 | 90.85 | 90.85 | 92.16 | 95.42 | 90.85 | 96.08 | 92.16 | 94.77 | 92.16 | 95.42 | 90.85 |
| Προσομοίωση_15 | 95.30 | 91.95 | 91.95 | 93.29 | 95.30 | 91.95 | 97.32 | 93.29 | 94.63 | 91.95 | 95.30 | 91.95 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_16 | 95.97 | 95.97 | 89.93 | 91.28 | 95.30 | 94.63 | 95.30 | 94.63 | 94.63 | 93.29 | 93.96 | 94.63 |
| Προσομοίωση_17 | 95.30 | 94.63 | 89.93 | 88.59 | 95.97 | 93.29 | 95.97 | 94.63 | 95.97 | 94.63 | 95.97 | 94.63 |
| Προσομοίωση_18 | 94.63 | 95.97 | 93.29 | 93.29 | 97.99 | 94.63 | 96.64 | 95.97 | 95.30 | 94.63 | 97.99 | 95.97 |
| Προσομοίωση_19 | 100 | 95.97 | 91.95 | 94.63 | 97.99 | 94.63 | 97.99 | 95.97 | 95.30 | 94.63 | 97.32 | 95.97 |

**Παράρτημα 5:** Αναθέσεις (%) προσομοιώσεων αλγορίθμων για το δύσκολο πρόβλημα

| **ΧΡΟΝΟΣ** (msec) | hApA | hApB | hBpA | hBpB | hCpA | hCpB | hDpA | hDpB | hEpA | hEpB | hFpA | hFpB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_1 | 330 | 77 | 77 | 74 | 72 | 76 | 75 | 70 | 71 | 85 | 75 | 80 |
| Προσομοίωση_2 | 262 | 76 | 69 | 70 | 73 | 71 | 86 | 69 | 65 | 69 | 77 | 79 |
| Προσομοίωση_3 | 317 | 86 | 71 | 74 | 88 | 82 | 71 | 75 | 71 | 71 | 93 | 82 |
| Προσομοίωση_4 | 301 | 79 | 73 | 70 | 73 | 82 | 88 | 82 | 83 | 86 | 84 | 87 |
| Προσομοίωση_5 | 321 | 87 | 74 | 76 | 75 | 73 | 87 | 77 | 77 | 77 | 83 | 84 |
| Προσομοίωση_6 | 246 | 81 | 75 | 77 | 78 | 98 | 77 | 79 | 86 | 100 | 86 | 88 |
| Προσομοίωση_7 | 257 | 80 | 76 | 80 | 79 | 92 | 76 | 75 | 77 | 85 | 85 | 106 |
| Προσομοίωση_8 | 324 | 91 | 84 | 83 | 83 | 83 | 81 | 96 | 100 | 87 | 88 | 100 |
| Προσομοίωση_9 | 128 | 96 | 84 | 84 | 92 | 89 | 84 | 96 | 82 | 80 | 98 | 91 |
| Προσομοίωση_10 | 333 | 91 | 95 | 85 | 93 | 95 | 81 | 99 | 79 | 86 | 91 | 96 |
| Προσομοίωση_11 | 166 | 92 | 91 | 94 | 100 | 89 | 91 | 93 | 87 | 97 | 95 | 110 |
| Προσομοίωση_12 | 327 | 91 | 86 | 89 | 104 | 89 | 94 | 94 | 97 | 93 | 94 | 100 |
| Προσομοίωση_13 | 358 | 99 | 92 | 96 | 97 | 102 | 96 | 104 | 91 | 98 | 98 | 113 |
| Προσομοίωση_14 | 374 | 135 | 118 | 125 | 133 | 143 | 155 | 135 | 110 | 173 | 133 | 150 |
| Προσομοίωση_15 | 356 | 125 | 109 | 137 | 141 | 118 | 119 | 116 | 115 | 113 | 122 | 128 |
| Προσομοίωση_16 | 354 | 118 | 112 | 110 | 119 | 129 | 118 | 121 | 109 | 114 | 123 | 126 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Προσομοίωση_17 | 383 | 135 | 122 | 137 | 154 | 135 | 144 | 139 | 129 | 122 | 152 | 143 |
| Προσομοίωση_18 | 94.63 | 95.97 | 93.29 | 93.29 | 97.99 | 94.63 | 96.64 | 95.97 | 95.30 | 94.63 | 97.99 | 95.97 |
| Προσομοίωση_19 | 365 | 155 | 145 | 133 | 156 | 151 | 154 | 148 | 131 | 167 | 144 | 151 |

**Παράρτημα 6:** Χρόνοι εκτέλεσης (msec) προσομοιώσεων αλγορίθμων για το δύσκολο πρόβλημα

```
2
1
10
ΚΑΘΗΓΗΤΗΣ
_01
40
41
42
43
30
31
13
14
15
16
17
18
19
20
0
*
3
ΜΑΘΗΜΑ_01
n
1
Ω
*
2
ΜΑΘΗΜΑ_02
y
1
Ω
*
2
ΜΑΘΗΜΑ_03
n
7
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_02
41
42
43
44
0
*
2
ΜΑΘΗΜΑ_04
n
1
Ω
*
2
ΜΑΘΗΜΑ_05
n
1
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_03
25
26
27
28
29
20
31
32
33
0
*
2
ΜΑΘΗΜΑ_6
n
1
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_04
51
52
53
54
55
56
57
58
59
0
*
2
ΜΑΘΗΜΑ_7
n
1
Ω
*
2
ΜΑΘΗΜΑ_8
y
7
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_05
1
2
37
38
55
56
0
*
2
ΜΑΘΗΜΑ_9
n
1
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_06
40
41
28
29
0
*
2
ΜΑΘΗΜΑ_10
n
1
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_07
13
14
15
16
17
18
0
*
3
ΜΑΘΗΜΑ_11
n
1
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_08
30
31
32
33
34
35
9
10
0
*
3
ΜΑΘΗΜΑ_12
n
3
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_09
20
21
22
23
53
54
55
56
57
52
0
*
2
ΜΑΘΗΜΑ_13
n
3
Ω
*
3
ΜΑΘΗΜΑ_14
y
3
Ω
*
0
ΚΑΘΗΓΗΤΗΣ
_10
13
14
15
16
25
26
27
28
32
33
34
35
37
38
39
40
0
*
2
ΜΑΘΗΜΑ_15
n
3
Ω
*
2
ΜΑΘΗΜΑ_16
n
3
Ω
*
3
ΜΑΘΗΜΑ_17
y
5
Ω
*
2
ΜΑΘΗΜΑ_18
y
5
Ω
*

0
```

**Παράρτημα 7:** Ένα από τα αρχεία εισόδου δεδομένων του εύκολου προβλήματος