ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΚΑΤΑΡΤΙΣΗΣ ΩΡΟΛΟΓΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ: ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΤΟ ΣΥΣΤΗΜΑ ECL^{i}PS^{e}

Από την Ελένη Ψαρά

Πολυτεχνείο Κρήτης

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Μια διπλωματική εργασία που παρουσιάστηκε στο Πολυτεχνείο Κρήτης για την εκπλήρωση των απαιτήσεων απόκτησης διπλώματος Ηλεκτρονικού Μηχανικού και Μηχανικού Ηλεκτρονικών Υπολογιστών

Χανιά 2004

ΠΕΡΙΛΗΨΗ

Η κατάρτιση ωρολογίου προγράμματος για Πανεπιστήμια ή σχολεία αποτελεί ένα αρκετά σύνθετο πρόβλημα, του οποίου η χειρονακτική εύρεση λύσης απαιτεί πολύ κόπο και χρόνο. Το σύνολο όλων των λύσεων, το οποίο είναι ο χώρος αναζήτησης του προβλήματος, είναι πολύ μεγάλος, τουλάχιστον σε πραγματικά προβλήματα. Μία αποδεκτή λύση είναι εκείνη η οποία ικανοποιεί όλους τους περιορισμούς του προβλήματος. Το πρόβλημα περιπλέκεται στην περίπτωση που κάποιος επιθυμεί την κατάρτιση ενός ποιοτικού ωρολογίου προγράμματος σύμφωνα με κάποια ευριστικά κριτήρια.

Στην παρούσα διπλωματική διατριβή, αναπτύχθηκε ένα σύστημα κατάρτισης ωρολογίου προγράμματος. Ο σχεδιασμός του μοντέλου πραγματοποιήθηκε με βάση τη θεωρία των προβλημάτων ικανοποίησης περιορισμών (constraint satisfaction problems, CSPs), δίνοντας έτσι στον χρήστη την δυνατότητα να εκφράσει τους περιορισμούς που ισχύουν στο εκάστοτε πρόβλημα σύμφωνα με τις προτιμήσεις του. Σκοπός του συστήματος είναι η ανάθεση των διαλέξεων όλων των μαθημάτων στο χρόνο και στις αίθουσες έτσι ώστε η λύση που προκύπτει να ικανοποιεί όλους τους περιορισμούς του προβλήματος, καθώς επίσης να ικανοποιεί τα κριτήρια εκείνα βάση των οποίων θεωρείται αποδεκτή από τον χρήστη σε σχέση με κάποιες άλλες λύσεις που επίσης ικανοποιούν τους περιορισμούς άλλα δεν θεωρούνται τόσο ποιοτικές. Αυτό πραγματοποιήθηκε μέσω ενός αλγορίθμου ευριστικής αναζήτησης, καθώς επίσης και με την βοήθεια τεχνικών συνέπειας και διάδοσης περιορισμών (constraint propagation) που έχουν αναπτυχθεί για τέτοιου είδους προβλήματα. Στην περίπτωση που ο χρήστης δώσει στο σύστημα δεδομένα τα οποία δεν ικανοποιούν τους περιορισμούς τότε το σύστημα τερματίζει και ζητά από τον χρήστη να χαλαρώσει κάποιους από τους περιορισμούς που έχει θέσει ή να πραγματοποιήσει κάποιες αλλαγές στα δεδομένα. Στην περίπτωση που βρεθεί λύση το σύστημα ενημερώνει τον χρήστη και αναμένει απάντηση ως προς το αν επιθυμεί επόμενη λύση ή το τερματισμό της αναζήτησης. Εάν ο χρήστης επιθυμεί επόμενη λύση το σύστημα θα αναζητήσει λύση της οποίας η συνάρτηση κόστους (cost function) θα είναι ίση ή μικρότερη της προηγούμενης.

Η υλοποίηση του συστήματος πραγματοποιήθηκε μέσω του συστήματος της ECL^{i}PS^e , η οποία αποτελεί μία γλώσσα λογικού προγραμματισμού με περιορισμούς. Αποτελεί ουσιαστικά ένα Prolog σύστημα, ενισχυμένο με τις διάφορες επεκτάσεις που παρέχουν μία ικανοποιητική ευελιξία για την αντιμετώπιση διαφορετικών προβλημάτων.

Κατασκευάστηκε επίσης ένα γραφικό εργαλείο για την κατασκευή και επεξεργασία ωρολογίου προγράμματος. Το εργαλείο αυτό υποστηρίζει πλήρως την διαχείριση των δεδομένων του προβλήματος από τον χρήστη, σύμφωνα με το μοντέλο που αναπτύξαμε. Τέλος, δόθηκε έμφαση στην, κατά το δυνατόν, διευκόλυνση του χρήστη στο χειρισμό του εργαλείου.

ΕΥΧΑΡΙΣΤΙΕΣ

Η συγγραφέας θα ήθελε να ευχαριστήσει τον καθηγητή κ. Μανόλη Κουμπαράκη για την επίβλεψη και καθοδήγησή του στην εκπόνηση αυτής της διπλωματικής εργασίας. Θα ήθελε επίσης να ευχαριστήσει τους καθηγητές κ. Βασίλη Σαμολαδά και κ. Ευριπίδη Πετράκη για τον χρόνο που αφιέρωσαν στην ανάγνωση της και τις παρατηρήσεις τους. Επίσης, οφείλει ένα ιδιαίτερο ευχαριστώ στους Γιώργο και Γιάννη Κοτόπουλο για τις πολύτιμες συμβουλές και βοήθειά τους.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Περίληψη – Η Κατάρτιση Ωρολογίου Προγράμματος στα Πανεπιστήμια

Το πρόβλημα της κατάρτισης ωρολογίου προγράμματος μπορεί να οριστεί ως ένα πρόβλημα χρονικού προγραμματισμού όπου ένας συγκεκριμένος αριθμός διαλέξεων, κάθε μία από τις οποίες μπορεί να ανήκει σε μία ή και περισσότερες ομάδες μαθημάτων, πρέπει να ανατεθεί στο χρόνο. Κάθε διάλεξη απαιτεί κάποιους πόρους (αίθουσες και καθηγητές), ο αριθμός των οποίων είναι περιορισμένος, όπως επίσης πρέπει να ικανοποιούνται και κάποιοι αυστηροί περιορισμοί, όπως για παράδειγμα *δύο διαλέξεις οι οποίες διδάσκονται από τον ίδιο καθηγητή δεν μπορούν να συμπίπτουν χρονικά*, ή κάπως λιγότερο αυστηροί περιορισμοί οι οποίοι μπορεί και να μην ικανοποιούνται απαραίτητα, όπως για παράδειγμα *μια διάλεξη η οποία παρακολουθείται από μεγάλο αριθμό φοιτητών πρέπει να ανατίθεται σε αίθουσα μεγάλης χωρητικότητας.*

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. Ανήκει στις τάξεις των NP-complete προβλημάτων και ο χρόνος αναζήτησης λύσης αυξάνει εκθετικά σε σχέση με τον αριθμό των δεδομένων. Για το λόγω αυτό είναι πολύ σημαντικό τόσο το μοντέλο που θα αναπτυχθεί για το πρόβλημα όσο και ο αλγόριθμος αναζήτησης λύσης που θα χρησιμοποιηθεί στην αποδοτικότητα του συστήματος.

Ένα πρόβλημα κατάρτισης ωρολογίου προγράμματος μπορεί να έχει πολλές λύσεις 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, πιθανώς μέσω μιας εξαντλητικής αναζήτησης του χώρου.

Η μοντελοποίηση του προβλήματος κατάρτισης ωρολογίου προγράμματος μπορεί να γίνει βάση της θεωρίας των προβλημάτων ικανοποίησης περιορισμών (constraint satisfaction problems, CSPs) και αυτό γιατί από μόνο του το πρόβλημα εμπεριέχει πολλούς περιορισμούς. Κατά τη διάρκεια των ετών, αναπτύχθηκαν ποικίλες τεχνικές αναζήτησης στον χώρο της τεχνητής νοημοσύνης για την αντιμετώπιση τέτοιου είδους προβλημάτων, όπως για παράδειγμα αλγόριθμοι τοπικής αναζήτησης όπως ο tabu search [4], γενετικοί αλγόριθμοι [15], ευριστικής αναζήτησης [1, 5, 6, 12, 13, 16], reducing domains strategy [7, 8, 9, 10, 11] και βελτιστοποίησης [2, 3, 5].

Πρέπει να σημειωθεί ότι δεν είναι δυνατή η ύπαρξη μοντέλου για το πρόβλημα κατάρτιση 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. Έτσι είναι πιθανόν μία εφαρμογή η οποία έχει αναπτυχθεί για κάποιο ίδρυμα να θεωρείται ανεπαρκής για κάποιο άλλο ή ακόμα και για το ίδιο το ίδρυμα σε περίπτωση που κάποιες αλλαγές πραγματοποιηθούν. Παρόλα αυτά 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.

Ο λογικός προγραμματισμός με περιορισμούς (constraint logic programming) έχει χρησιμοποιηθεί εκτενώς στην αντιμετώπιση των προβλημάτων κατάρτισης ωρολογίου προγράμματος. Η διαδικασία επίλυσης του συγκεκριμένου προβλήματος, συνήθως πραγματοποιείται σε δύο στάδια. Το πρώτο στάδιο αφορά την επιβολή των περιορισμών και την διερεύνηση τους. Οι περιορισμοί αυτοί πρέπει να επιβληθούν πάνω στο σύνολο των μεταβλητών του μοντέλου που έχει αναπτυχθεί για το πρόβλημα. Το δεύτερο στάδιο αφορά την ανάθεση τιμών, και πιο συγκεκριμένα των κατάλληλων τιμών, στις προαναφερθέντες μεταβλητές. Η επιτυχής μετάβαση και από τα δύο στάδια οδηγεί στην κατάρτιση ενός ωρολογίου προγράμματος το οποίο μπορεί να χαρακτηριστεί ως αποδεκτό από όλα τα ενδιαφερόμενα μέλη.

1.2 Οργάνωση της διατριβής

Στο δεύτερο κεφάλαιο ασχοληθήκαμε με τα προβλήματα ικανοποίησης περιορισμών, τις τεχνικές και αλγορίθμους που έχουν αναπτυχθεί για τέτοιου είδους προβλήματα. Γίνεται μία παρουσίαση του συστήματος της ECL^{i}PS^e και μία σύντομη αναφορά παρόμοιων εμπορικών συστημάτων.

Στο τρίτο κεφάλαιο παρουσιάζεται το θεωρητικό μοντέλου που έχουμε αναπτύξει για το πρόβλημα της κατάρτισης ωρολογίου προγράμματος, το οποίο ανήκει στην ομάδα προβλημάτων ικανοποίησης περιορισμών. Παρουσιάζονται οι περιορισμοί που πρέπει να ικανοποιούνται καθώς επίσης τα κριτήρια βάση των οποίων μία λύση θεωρείται κατάλληλη.

Στο τέταρτο κεφάλαιο περιγράφεται η υλοποίηση του μηχανισμού κατάρτισης ωρολογίου προγράμματος. Πιο αναλυτικά, περιγράφονται η υλοποίηση των περιορισμών, του ευριστικού αλγόριθμου αναζήτησης που έχουμε αναπτύξει καθώς επίσης και του τρόπου επικοινωνίας του συστήματος της ECL^{i}PS^e με το σύστημά μας.

Στο πέμπτο κεφάλαιο περιγράφεται το γραφικό περιβάλλον για την διαχείριση του ωρολογίου προγράμματος από τους χρήστες.

Στο έκτο και τελευταίο κεφάλαιο γίνεται μία ανακεφαλαίωση, αναφέρονται κάποια συμπεράσματα καθώς επίσης γίνονται και κάποιες επισημάνσεις για μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Σχετική δουλειά

2.1 Εισαγωγή

Το κεφάλαιο αυτό ασχολείται με τα προβλήματα ικανοποίησης περιορισμών, τις τεχνικές και αλγορίθμους που έχουν αναπτυχθεί για τέτοιου είδους προβλήματα. Γίνεται μία παρουσίαση του συστήματος της ECL^{i}PS^e και μία σύντομη αναφορά παρόμοιων εμπορικών συστημάτων.

Η παρουσίαση και τα παραδείγματα αυτού του κεφαλαίου βασίζονται στο κεφάλαιο 5 του βιβλίου Artificial Intelligence A Modern Approach των Stuart Russell και Peter Norvig, καθώς επίσης στο κεφάλαιο 3 του βιβλίου Programming with Constraints, An Introduction των Kim Marriott και Peter J. Stuckey.

2.2 Προβλήματα Ικανοποίησης Περιορισμών (Constraint Satisfaction Problems, CSP)

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".

Ορισμός 2.1 [28, 26]

Ένα πρόβλημα ικανοποίησης περιορισμών, αποτελείται από ένα σύνολο περιορισμών C, το οποίο εφαρμόζεται πάνω σε ένα σύνολο μεταβλητών $x_1,x_2,...,x_n$, 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 CSP αναπαριστά τον περιορισμό $C \wedge x_1 \in D(x_1) \wedge ... \wedge x_n \in D(x_n)$.

Παράδειγμα 2.1 [28]

Το πρόβλημα του χρωματισμού χάρτη (map coloring problem) αποτελεί ένα από τα αρχέτυπα CSP προβλήματα. 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 2.1 και ότι έχουμε τα χρώματα κόκκινο, κίτρινο και μπλε.

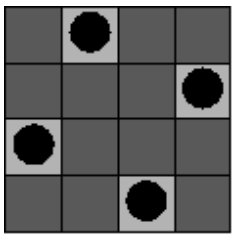
Σχήμα 2.1 : Το πρόβλημα χρωματισμού χάρτη

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 δεν μπορεί να έχει το ίδιο χρώμα:

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

Παράδειγμα 2.2 [28]

Ένα ακόμα γνωστό CSP πρόβλημα είναι το πρόβλημα των *N* βασιλισσών (N-queens). Στο πρόβλημα αυτό πρέπει να τοποθετηθούν *N* βασίλισσες πάνε σε μία σκακιέρα μεγέθους *N×N* έτσι ώστε όλες οι βασίλισσες να είναι σε διαφορετικές γραμμές, στήλες και διαγώνιους. Ας θεωρήσουμε το πρόβλημα των $4^{ων}$ βασιλισσών.



Σχήμα 2.2 : Το πρόβλημα των $4^{ων}$ βασιλισσών

Μπορούμε να τυποποιήσουμε το πρόβλημα ως CSP θεωρώντας ότι κάθε βασίλισσα *i* έχει δύο μεταβλητές, την μεταβλητή R_i και C_i 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 δεν 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 δεν μπορούν να βρίσκονται στην ίδια στήλη, και τέλος οι περιορισμοί

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

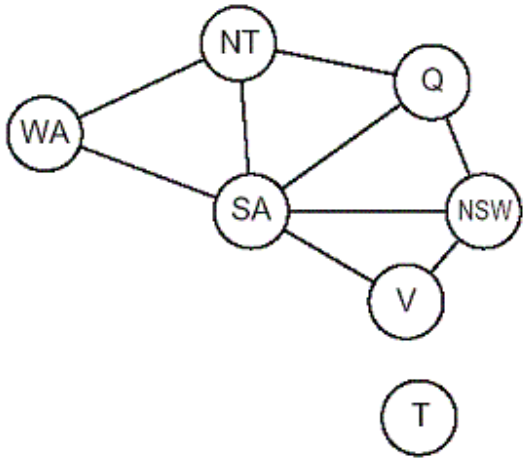
και

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

εξασφαλίζουν ότι δύο βασίλισσες δεν μπορούν να βρίσκονται στην ίδια διαγώνιο. Μία λύση στο παραπάνω πρόβλημα είναι αυτή του σχήματος 2.2 .

2.3 Αναπαράσταση περιορισμών με γράφο

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. Χαρακτηριστικό παράδειγμα αναπαράστασης CSP με γράφο είναι το παράδειγμα χρωματισμού χάρτη που αναφέραμε παραπάνω στο παράδειγμα 2.1 . Η απεικόνιση του παραπάνω προβλήματος με γράφο φαίνεται στο σχήμα 2.3 .



Σχήμα 2.3 : Αναπαράσταση περιορισμών με γράφο

2.4 Προβλήματα Ικανοποίησης Περιορισμών με πεπερασμένα πεδία

Το απλούστερο είδος CSP προβλημάτων είναι εκείνα που περιέχουν διακριτές μεταβλητές με πεπερασμένα πεδία τιμών. Χαρακτηριστικά παραδείγματα το παράδειγμα του χρωματισμού χάρτη και εκείνο των *N* βασιλισσών. Στην ομάδα των CSP με πεπερασμένα πεδία ανήκουν και τα προβλήματα εκείνα των οποίων οι μεταβλητές μπορούν να πάρουν 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 $O(d^n)$, 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 δεν μπορούμε να περιμένουμε ότι ο χρόνος επίλυσης μπορεί να είναι μικρότερος του εκθετικού.

2.5 Προβλήματα Ικανοποίησης Περιορισμών με μη πεπερασμένα πεδία

Οι διακριτές μεταβλητές μπορούν επίσης να έχουν μη πεπερασμένα πεδία, για παράδειγμα το σύνολο των ακεραίων αριθμών ή το σύνολο των αλφαριθμητικών. Ας θεωρήσουμε ότι έχουμε δύο εργασίες J_1 και J_2 τις οποίες πρέπει να αναθέσουμε στο χρόνο. Εάν τώρα θεωρήσουμε ότι κάθε εργασία J_i έχει μία μεταβλητή $StartJ_i$ 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 J_1 απαιτεί *5* ημέρες για να διεκπεραιωθεί και ισχύει ότι πρέπει να προηγείται της εργασίας J_2, τότε χρειαζόμαστε μία γλώσσα περιορισμών με αλγεβρικές ανισότητες όπως $StartJ_1 + 5 \leq StartJ_2$. 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 ακεραίων πεδίων. Σε κάποιες περιπτώσεις, μπορούμε να μετατρέψουμε ένα CSP 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.

2.6 Περιορισμοί

Επιπλέον εξετάζοντας τον τύπο των μεταβλητών που μπορούν να υπάρξουν σε ένα CSP, είναι χρήσιμο να εξετάσουμε και το είδος των περιορισμών.

2.6.1 Μοναδιαίος Περιορισμός (Unary Constraint)

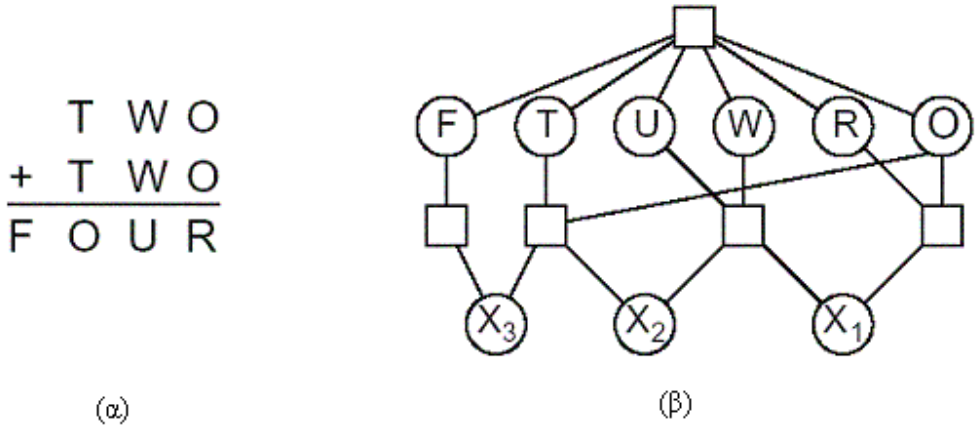
Ο απλούστερος τύπος περιορισμού που υπάρχει είναι ο μοναδιαίος περιορισμός, ο οποίος περιορίζει την τιμή μίας μοναδικής μεταβλητής. Για παράδειγμα στο παράδειγμα του χρωματισμού γράφου μπορεί η μεταβλητή *WA* να μην επιτρέπεται να έχει το χρώμα κόκκινο. Κάθε μοναδιαίος περιορισμός μπορεί εξαλειφθεί απλά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

2.6.2 Δυαδικός Περιορισμός (Binary Constraint)

Ένας δυαδικός περιορισμός συσχετίζει τις τιμές δύο μεταβλητών, για παράδειγμα $X \neq Y$ αποτελεί ένα δυαδικό περιορισμό στον οποίο οι μεταβλητές X και *Y* δεν μπορούν να έχουν τις ίδιες τιμές. Οι αναπαράσταση δυαδικών περιορισμών μπορεί να γίνει με γράφο όπως στο σχήμα 2.3 .

2.6.3 Υψηλού βαθμού Περιορισμός (High-order Constraint)

Οι Υψηλού βαθμού περιορισμοί συσχετίζουν τις τιμές τριών και περισσοτέρων μεταβλητών. Ένα οικείο παράδειγμα είναι αυτό του κρυπταριθμητικού αινίγματος του σχήματος 2.4(α) .



Σχήμα 2.4 : Το πρόβλημα του κρυπταριθμητικού αινίγματος και η αναπαράστασή του με γράφο

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. Οι επιπλέον τέσσερις περιορισμοί για κάθε

στήλη του αινίγματος που συσχετίζουν τις μεταβλητές μπορούν να εκφραστούν από τις σχέσεις

$$O + O = R + 10 \cdot X_1$$
$$X_1 + W + W = U + 10 \cdot X_2$$
$$X_2 + T + T = O + 10 \cdot X_3$$
$$X_3 = F$$

όπου οι μεταβλητές X_1, X_2 και X_3 είναι βοηθητικές μεταβλητές και αναπαριστούν το κρατούμενο *0* ή *1*. Οι Υψηλού βαθμού περιορισμοί μπορούν να απεικονιστούν από ένα υπερ-γράφο περιορισμών (constraint hypergraph) όπως αυτός του σχήματος 2.4(β), στον οποίο κάθε περιορισμός απεικονίζεται με ένα τετράγωνο και οι ακμές οι οποίες συνδέουν τον περιορισμό με τις συσχετιζόμενες μεταβλητές.

2.6.4 Αυστηροί και Εύκαμπτοι Περιορισμοί (Hard and Soft Constraints)

Οι περιορισμοί που έχουμε περιγράψει μέχρι τώρα αποτελούν αυστηρούς περιορισμούς η παραβίαση των οποίων αποκλείει κάθε δυνητική λύση. Υπάρχουν όμως και περιπτώσεις πραγματικών εφαρμογών με CSP, 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 [2, 3] ή και τοπικής αναζήτησης όπως είναι ο tabu search [4]. 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 τις 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.

2.7 Αλγόριθμος Χρονολογικής Οπισθοδρόμησης (Chronological Backtracking)

Ο όρος *αναζήτηση με χρονολογική οπισθοδρόμηση* ,όπως αναφέρεται στο [26], χρησιμοποιείται για να περιγράψει έναν αλγόριθμο αναζήτησης *κατά βάθος* ο οποίος επιλέγει τιμές για μία μεταβλητή κάθε φορά και κάθε φορά που ένας κλάδος της αναζήτησης αποτυγχάνει, τότε επιστρέφει στην προηγούμενη μεταβλητή και επιλέγει διαφορετική τιμή γι' αυτήν. Αποκαλείται *χρονολογική οπισθοδρόμηση* γιατί σε κάθε αποτυχία επιστρέφει στο πιο πρόσφατο σημείο διακλάδωσης. Στο σχήμα 2.5 παρουσιάζεται ο αλγόριθμος οπισθοδρόμησης.

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING([],csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment according to CONSTRAINTS(csp) then
        result ← RECURSIVE-BACKTRACKING([var = value – assignment], csp)
        if result ≠ failure then return result
    end
    return failure
```

Σχήμα 2.5 : Αλγόριθμος χρονολογικής οπισθοδρόμησης

Όπως φαίνεται στο σχήμα 2.5 ο αλγόριθμος περιέχει την γραμμή
var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[*csp*], *assignment, csp*) όπου η μέθοδος SELECT-UNASSIGNED-VARIABLE επιλέγει την επόμενη μεταβλητή του προβλήματος στην οποία θα γίνει ανάθεση τιμής. Η επιλογή της μεταβλητής μπορεί να γίνει βάση των *ευριστικών διάταξης μεταβλητών* που θα αναφέρουμε στην παράγραφο 2.8 ή βάση άλλων ευριστικών που έχει θεωρήσει ο προγραμματιστής. Ομοίως η μέθοδος ORDER-DOMAIN-VALUES διατάσει τις τιμές της μεταβλητής που έχει επιλεγεί για ανάθεση. Η διάταξη των τιμών μπορεί να γίνει βάση των *ευριστικών διάταξης τιμών* που θα αναφέρουμε στην παράγραφο 2.8 ή βάση άλλων ευριστικών που έχει θεωρήσει ο προγραμματιστής.

2.8 Ευριστικές Διάταξης Μεταβλητών και Τιμών

Η διάταξη με την οποία επιλέγονται οι μεταβλητές προκειμένου να γίνει ανάθεση τιμών, καθώς και η διάταξη τιμών με την οποία αναθέτονται στην επιλεγμένη μεταβλητή, παίζουν καθοριστικό ρόλο την ταχύτητα εύρεσης λύσης για έναν αλγόριθμο. Δεδομένου του ότι οι αλγόριθμοι αναζήτησης που εφαρμόζονται σε CSP είναι κυρίως οι *αναζήτησης κατά βάθος,* όπως αυτός της *αναζήτησης με χρονολογική οπισθοδρόμηση* που αναφέραμε στην παράγραφο 2.7, είναι πολύ σημαντικό να μπορέσουμε να κατευθύνουμε την αναζήτηση προς την περιοχή που είναι πιο πιθανόν να υπάρχει λύση.

2.8.1 Η Περισσότερο Περιορισμένη Μεταβλητή (Most Constrained Variable)

Ας θεωρήσουμε πάλι το παράδειγμα του χρωματισμού γράφου 2.1 και ας υποθέσουμε ότι έχει γίνει ανάθεση τιμών στις μεταβλητές *WA = κόκκινο* και *NT = πράσινο* όπως φαίνεται στο σχήμα 2.6.

Σχήμα 2.6 : Το πρόβλημα του χρωματισμού χάρτη εφαρμόζοντας την στρατηγική της περισσότερο περιορισμένης μεταβλητής

Όπως φαίνεται και από το σχήμα υπάρχει μία μόνο πιθανή τιμή για την μεταβλητή *SA = μπλε*, επομένως είναι λογικό η επόμενη μεταβλητή που θα επιλεγεί να είναι *SA = μπλε* παρά να επιλεγεί η *Q*. Στην πραγματικότητα κάνοντας ανάθεση τιμής στην μεταβλητή *SA*, οι επιλογές τιμών για τις μεταβλητές *Q, NSW* και *V* περιορίζονται υποχρεωτικά. Η ιδέα να επιλέγουμε για ανάθεση, την μεταβλητή με τις λιγότερες επιτρεπόμενες τιμές ονομάζεται *ευριστική της περισσότερο περιορισμένης μεταβλητής*. Έτσι, εάν υπάρχει μεταβλητή *X* με μηδέν επιτρεπόμενες τιμές τότε η ευριστική θα επιλέξει την μεταβλητή και θα αποτύχει, αποφεύγοντας έτσι την άσκοπη αναζήτηση για όλες τις μεταβλητές καθώς θα υπάρχει πάντα αποτυχία όταν η *X* θα επιλέγεται τελευταία.

2.8.2 Η Περισσότερο Περιοριστική Μεταβλητή (Most Constraining Variable)

Επανερχόμενοι πάλι στο παράδειγμα 2.1, παρατηρούμε ότι η ευριστική της περισσότερο περιορισμένης μεταβλητής δεν βοηθάει καθόλου στην επιλογή της πρώτης μεταβλητής που θέλουμε να αναθέσουμε και αυτό γιατί αρχικά όλες οι μεταβλητές έχουν τις ίδιες επιτρεπόμενες τιμές. Σε αυτή την περίπτωση μπορεί να γίνει χρήση της *ευριστικής της περισσότερο περιοριστικής μεταβλητής*. Με την χρήση της παραπάνω ευριστικής επιχειρούμε να μειώσουμε τον παράγοντα διακλάδωσης (branching factor) των μελλοντικών επιλογών, κάνοντας ανάθεση τιμής στην μεταβλητή εκείνη η οποία μετέχει στον μεγαλύτερο αριθμό περιορισμών. Από το σχήμα 2.1 βλέπουμε ότι η μεταβλητή που μετέχει στους περισσότερους περιορισμούς είναι η *SA*. Επομένως επιλέγοντας αρχικά την *SA* το πρόβλημα επιλύεται χωρίς εσφαλμένα βήματα. Η *ευριστικής της περισσότερο περιοριστικής μεταβλητής* αποτελεί συνήθως καλύτερο οδηγό από την *ευριστική της περισσότερο περιορισμένης μεταβλητής*. Παρόλα αυτά, η *ευριστική της περισσότερο περιορισμένης μεταβλητής* μπορεί να χρησιμοποιηθεί στην περίπτωση όπου δύο μεταβλητές μετέχουν στον ίδιο αριθμό περιορισμών, αυτό σημαίνει ότι μπορούμε να έχουμε τον συνδυασμό των δύο ευριστικών.

2.8.3 Η Λιγότερο Περιοριστική Τιμή (Least Constraining Value)

Εφόσον μία μεταβλητή έχει επιλεγεί, ο αλγόριθμος πρέπει να αποφασίσει την διάταξη με την οποία θα εξετάσει τις τιμές. Σε αυτήν την περίπτωση η *ευριστική της λιγότερο περιοριστικής τιμής* αποδεικνύεται πολύ αποτελεσματική. Με βάση αυτόν τον κανόνα, επιλέγετε η τιμή εκείνη η οποία επιτρέπει τις περισσότερες επιλογές τιμών για τις γειτονικές μεταβλητές. Για παράδειγμα ας θεωρήσουμε ξανά ότι έχει γίνει η εξής ανάθεση τιμών στις μεταβλητές *WA = κόκκινο* και *NT = πράσινο* όπως φαίνεται στο σχήμα 2.6 , και ότι η επόμενη επιλογή θα είναι η μεταβλητή *Q*. Το *μπλε* χρώμα σίγουρα θα αποτελούσε μία κακή επιλογή καθώς θα άφηνε κενό το σύνολο των επιτρεπόμενων τιμών της μεταβλητής *SA*. Με βάση την *ευριστική της λιγότερο περιοριστικής τιμής* θα επιλεγόταν το *κόκκινο* χρώμα έναντι του *μπλε*. Γενικά, η ευριστική προσπαθεί να επιτρέψει την μέγιστη ευελιξία στην ανάθεση τιμών για τις επόμενες μεταβλητές.

2.9 Διάδοση περιορισμού (Constraint propagation) και Τεχνικές συνέπειας

Κάποιες φορές ο χώρος αναζήτησης μπορεί να μειωθεί δραστικά, εξετάζοντας τις συνέπειες που επιφέρει η μερική ανάθεση τιμών στις μεταβλητές όπου δεν τους 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. Η γενικός όρος για αυτήν την διαδικασία ονομάζεται *διάδοση περιορισμού*.

2.9.1 Εμπρόσθιος έλεγχος (Forward checking)

Το απλούστερο είδος διάδοσης περιορισμού ονομάζεται *εμπρόσθιος έλεγχος*. Κάθε φορά που σε μία μεταβλητή *X* γίνεται ανάθεση τιμής, η διαδικασία *εμπρόσθιου ελέγχου* εξετάζει κάθε μη ανατεθειμένη τιμή της *Y*, η οποία είναι συνδεδεμένη με την μεταβλητή *X* μέσω ενός περιορισμού, και διαγράφει από το πεδίο τιμών της *Y* όποια τιμή είναι ασυνεπής ως προς την επιλεγμένη τιμή της *X*. Στο σχήμα 2.7 φαίνεται η διαδικασία αναζήτησης λύση για το πρόβλημα χρωματισμού χάρτη με *εμπρόσθιο έλεγχο*.

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Αρχικά πεδία	Κ Π Μ	Κ Π Μ	Κ Π Μ	Κ Π Μ	Κ Π Μ	Κ Π Μ	Κ Π Μ
Μετά WA = κόκκινο	Κ	Π Μ	Κ Π Μ	Κ Π Μ	Κ Π Μ	Π Μ	Κ Π Μ
Μετά Q = πράσινο	Κ	Μ	Π	Κ Μ	Κ Π Μ	Μ	Κ Π Μ
Μετά V = μπλε	Κ	Μ	Π	Κ	Μ		Κ Π Μ

Σχήμα 2.7 : Παράδειγμα εμπρόσθιου ελέγχου

Υπάρχουν δύο σημαντικά σημεία στα οποία πρέπει να εστιάσουμε σε αυτό το παράδειγμα. Το πρώτο είναι όταν οι μεταβλητές *WA = κόκκινο* και *Q = πράσινο*, τότε τα πεδία των *NT* και *SA* μειώνονται έχοντας μοναδική τιμή. Μειώσαμε την διακλάδωση για αυτές τις μεταβλητές, διαδίνοντας την πληροφορία από τις *WA* και *Q*. Η *ευριστική της περισσότερο περιορισμένης μεταβλητής*, η οποία μπορεί να συνεργαστεί καλά με τον *εμπρόσθιο έλεγχο*, θα επιλέξει αυτόματα τις μεταβλητές *SA* και *NT*. Το δεύτερο σημείο

είναι όταν μετά την ανάθεση της *V = μπλε*, το πεδίο της *SA* γίνεται κενό. Τότε ο *εμπρόσθιος έλεγχος* εντοπίζει ότι η μερική ανάθεση τιμών {*WA = κόκκινο,Q = πράσινο* και *V = μπλε*} είναι ασυνεπής ως προς τους περιορισμούς του προβλήματος και ο αλγόριθμος αναζήτησης οπισθοδρομεί αμέσως.

Παρά το ότι ο *εμπρόσθιος έλεγχος* μπορεί να ανιχνεύσει αρκετές ασυνέπειες, δεν μπορεί να τις ανιχνεύσει όλες. Για παράδειγμα, η μερική ανάθεση {*WA = κόκκινο* και *Q=πράσινο*} είναι ήδη ασυνεπής με τους περιορισμούς, αλλά ο *εμπρόσθιος έλεγχος* δεν μπορεί να το ανιχνεύσει άμεσα. Στην τέταρτη γραμμή του σχήματος 2.7 φαίνεται η δυσκολία: τόσο η *NT* όσο και η *SA* αναγκάζονται να πάρουν την τιμή *μπλε*, λόγω όμως του ότι γειτνιάζουν δεν μπορούν να έχουν το ίδιο χρώμα. Ο *εμπρόσθιος έλεγχος* διαδίδει την πληροφορία των περιορισμών από ανατεθειμένες μεταβλητές στις μη ανατεθειμένες που γειτνιάζουν, αλλά δεν κάνει τα επόμενο βήμα να διαδώσει πληροφορία και μεταξύ των μη ανατεθειμένων μεταβλητών. Προφανώς, μπορούμε να εντοπίσουμε όποια ασυνέπεια προσπαθώντας να λύσουμε το πρόβλημα και αποτυγχάνοντας σε κάθε διαδρομή που ακολουθούμε, αλλά έτσι χάνουμε την ουσία καθώς θα ήταν προτιμότερο αν με κάποιο πιο γρήγορο τρόπο μπορούσαμε νωρίτερα να εντοπίσουμε τις ασυνέπειες έτσι ώστε να αποφύγουμε την άσκοπη αναζήτηση.

2.9.2 Συνέπεια κόμβων (Node-consistency) και Συνέπεια τόξων (Arc-consistency)

Ορισμός 2.2 [28]

Ένας περιορισμός *c* ικανοποιεί την συνέπεια κόμβων για ένα πεδίο *D* στην περίπτωση που, είτε ο αριθμός των μεταβλητών που μετέχουν στον περιορισμό είναι διάφορος του *1*, είτε όταν μόνο μία μεταβλητή *x* μετέχει στον περιορισμό *c* τότε θα πρέπει για κάθε τιμή $d \in D(x)$, το $\{x \rightarrow d\}$ να αποτελεί λύση του *c*.

Ένα CSP με περιορισμό $c_1 \wedge \ldots \wedge c_n$ και πεδίο *D* ικανοποιεί την συνέπεια κόμβων εάν κάθε περιορισμός c_i ικανοποιεί την συνέπεια κόμβων ως προς το *D* για $1 \leq i \leq n$.

Ορισμός 2.3 [28]

Ένας περιορισμός *c* ικανοποιεί την συνέπεια τόξων για ένα πεδίο *D* στην περίπτωση που, είτε ο αριθμός των μεταβλητών που μετέχουν στον περιορισμό είναι διάφορος του *2*, είτε όταν μόνο δύο μεταβλητές *x* και *y* μετέχουν στον περιορισμό *c* τότε θα πρέπει για κάθε τιμή $d_x \in D(x)$ να υπάρχει τιμή $d_y \in D(y)$ τέτοια ώστε $\{x \rightarrow d_x, y \rightarrow d_y\}$ να αποτελεί λύση του *c* και για κάθε τιμή $d_y \in D(y)$ να υπάρχει τιμή $d_x \in D(x)$ τέτοια ώστε $\{x \rightarrow d_x, y \rightarrow d_y\}$ να αποτελεί λύση του *c*.

Ένα CSP με περιορισμό $c_1 \wedge \ldots \wedge c_n$ και πεδίο *D* ικανοποιεί την συνέπεια τόξων εάν κάθε περιορισμός c_i ικανοποιεί την συνέπεια τόξων ως προς το *D* για $1 \leq i \leq n$.

Για παράδειγμα, το παράδειγμα 2.1 ικανοποιεί τόσο την *συνέπεια κόμβων* όσο και την *συνέπεια τόξων*. Ικανοποιεί την *συνέπεια κόμβων* επειδή δεν υπάρχει περιορισμός στον οποίο να μετέχει μία μόνο μεταβλητή. Κάθε ανισότητα στο παράδειγμα ικανοποιεί την συνέπεια τόξων καθώς για κάθε χρώμα d_x στο πεδίο τιμών της πρώτης μεταβλητής *x* που μετέχει στην ανισότητα υπάρχει χρώμα d_y διαφορετικό του d_x στο πεδίο της δεύτερης μεταβλητής *y* και αντίστροφα. Έτσι όλοι οι περιορισμοί ικανοποιούν την συνέπεια τόξων. Το ίδιο μπορούμε να πούμε και για το παράδειγμα 2.2.

Εάν τροποποιήσουμε το παράδειγμα του χρωματισμού χάρτη και επιτρέψουμε το πεδίο τιμών να έχει μόνο δύο χρώματα, τότε το πρόβλημα δεν ικανοποιείται καθώς δεν υπάρχει

τρόπος τέτοιος ώστε να χρωματίσουμε τις τρεις περιοχές *WA, NT* και *SA* με δύο μόνον χρώματα. Παρόλα αυτά ικανοποιεί την *συνέπεια κόμβων* και *τόξων*, αποδεικνύοντας ότι ένα CSP μπορεί να μην ικανοποιείται ακόμα και όταν ικανοποιεί την *συνέπεια κόμβου* και *τόξου*.

Ας θεωρήσουμε τώρα τον περιορισμό $X < Y \wedge Y < Z$ και τις μεταβλητές *X,Y* και *Z* με πεδίο το *{1,2}*. Το παράδειγμα ικανοποιεί την *συνέπεια κόμβων* καθώς δεν υπάρχει περιορισμός στον οποίον να μετέχει μία μόνο μεταβλητή. Δεν ικανοποιεί όμως την *συνέπεια τόξων* και αυτό γιατί, αν θεωρήσουμε τον περιορισμό $X < Y$ και την τιμή *1* για την μεταβλητή *Y*, τότε δεν υπάρχει τιμή στο πεδίο του *X* η οποία να ικανοποιεί τον περιορισμό.

Ένα παράδειγμα CSP το οποίο δεν ικανοποιεί την *συνέπεια κόμβων* αλλά ούτε και την *συνέπεια τόξων* είναι ο περιορισμός $X < Y \wedge Y < Z \wedge Z \leq 2$ με πεδίο τιμών *D* όπου $D(X) = D(Y) = D(Z) = \{1,2,3\}$. Δεν ικανοποιεί την *συνέπεια κόμβων* γιατί η τιμή *3* για την μεταβλητή *Z* δεν είναι συνεπής ως προς τον περιορισμό $Z \leq 2$. Δεν ικανοποιεί επίσης την *συνέπεια τόξων* για τους ίδιους λόγους που αναφέραμε στην προηγούμενη παράγραφο. Εφαρμόζοντας τώρα την *συνέπεια κόμβων* θα προκύψει η εξής αλλαγή, τα πεδία τιμών των μεταβλητών *X* και *Y* θα παραμείνουν τα ίδια *{1,2,3}* αλλά το πεδίο τιμών της μεταβλητής *Z* θα γίνει *{1,2}*. Εφαρμόζοντας τώρα την *συνέπεια τόξων*, ο περιορισμός $X< Y$ θα ελεγχθεί με αποτέλεσμα το νέο πεδίο τιμών που θα προκύψει για την μεταβλητή Χ θα είναι *{1,2}*. Η τιμή *3* αφαιρείται από το πεδίο τιμών της *X* και αυτό γιατί δεν υπάρχει τιμή στο πεδίο της μεταβλητής *Y* τέτοιο ώστε να ικανοποιείται ο περιορισμός $3<Y$. Παρόμοια το νέο πεδίο τιμών που προκύπτει για την μεταβλητή *Y* θα είναι *{2,3}*. Εξετάζοντας τώρα τον περιορισμό $Y< Z$, το πεδίο τιμών της μεταβλητής *Y* υπολογίζεται να είναι το $\varnothing$ εφόσον δεν υπάρχουν τιμές μικρότερες από τις τιμές που μπορεί να πάρει η *Z*. Το πεδίο της *Z* επίσης γίνεται $\varnothing$. Στην συνέχεια το πεδίο της *X* θα υπολογιστεί και αυτό $\varnothing$ και τέλος το σύστημα θα επιστρέψει ότι το πρόβλημα είναι ανικανοποίητο.

Ο έλεγχος με *συνέπεια κόμβων* και *τόξων* μπορεί να εφαρμοστεί ως προεπεξεργασία πριν την έναρξη της διαδικασίας αναζήτησης, αλλά και στη συνέχεια ως διεργασία διάδοσης περιορισμού, κάθε φορά που γίνεται ανάθεση τιμής σε μία μεταβλητή κατά την διάρκεια της αναζήτησης. Και στις δύο περιπτώσεις η διαδικασία πρέπει να είναι επαναλαμβανόμενη έως ότου να μην υπάρχουν ασυνέπειες, και τούτο γιατί κάθε φορά που αφαιρείται μία τιμή από το πεδίο μιας μεταβλητής μία καινούργια ασυνέπεια μπορεί να εμφανίζεται. Παρά το γεγονός ότι η εφαρμογή του αλγόριθμου *συνέπειας κόμβων* και *τόξων* στοιχίζει περισσότερο από τον *εμπρόσθιο έλεγχο*, το πρόσθετο αυτό κόστος συνήθως αξίζει, καθώς είναι μία γρήγορη μέθοδος η οποία είναι πιο ισχυρή του *εμπρόσθιου έλεγχου*.

2.9.3 Συνέπεια υπερ-τόξων (Hyper-arc consistency)

Η *συνέπεια κόμβων* και *τόξων* λειτουργεί πολύ καλά, περικόπτοντας τις τιμές των πεδίων, στις περιπτώσεις των δυαδικών CSP. Παρόλα αυτά, δεν λειτουργεί καλά για περιπτώσεις προβλημάτων που περιέχουν περιορισμούς, στους οποίους μετέχουν περισσότερες από δύο μεταβλητές, καθώς αυτοί οι περιορισμοί δεν λαμβάνονται υπόψη κατά την εκτέλεση της συνέπειας ελέγχου. Θα ήταν καλό να επεκταθεί αυτή η τεχνική, έτσι ώστε να μπορεί να χειρίζεται περιορισμούς στους οποίους μετέχουν περισσότερες μεταβλητές.

Η *συνέπεια υπερ-τόξων* επεκτείνει τη συνθήκη για την *συνέπεια τόξων* που ισχύει για κάθε περιορισμό ανεξαρτήτως του αριθμού μεταβλητών που περιέχει.

Ορισμός 2.4 [28]
Ένας περιορισμός c ικανοποιεί την συνέπεια υπερ-τόξων για ένα πεδίο D εάν για κάθε μεταβλητή x που μετέχει στον c και για κάθε τιμή $d \in D(x)$, υπάρχουν τιμές για τις υπόλοιπες μεταβλητές του c, για παράδειγμα $x_1, x_2, \ldots, x_k$, τέτοιες ώστε για κάθε τιμή $d_j \in D(x_j)$ με $1 \leq j \leq k$ και $\{x \rightarrow d, x_2 \rightarrow d_2, \ldots, x_k \rightarrow d_k\}$να αποτελεί λύση του c.
Ένα CSP με περιορισμό $c_1 \wedge \ldots \wedge c_n$ και πεδίο D ικανοποιεί την συνέπεια υπερ-τόξων εάν κάθε περιορισμός c_i ικανοποιεί την συνέπεια υπερ-τόξων ως προς το D για $1 \leq i \leq n$.

Η *συνέπεια υπέρ-τόξων* αποτελεί γενίκευση της *συνέπειας κόμβων* και *τόξων*. Στη περίπτωση που σε έναν περιορισμό μετέχουν δύο μεταβλητές, τότε η *συνέπεια υπερ-τόξων* ισοδυναμεί με την *συνέπεια τόξων*, ενώ στην περίπτωση που μετέχει μία μόνο μεταβλητή τότε ισοδυναμεί με την *συνέπεια κόμβων*.
Δυστυχώς, η εφαρμογή της *συνέπειας υπερ-τόξων* μπορεί να αποβεί δαπανηρή για απλούς περιορισμούς στους οποίους μετέχουν περισσότερο από δύο μεταβλητές. Ας θεωρήσουμε τον περιορισμό $X=3Y + 5Z$ με πεδίο τιμών D όπου $D(X)=\{2,3,4,5,6,7\}$, $D(Y)=\{0,1,2\}$και $D(Z)=\{-1,0,1,2\}$. Εφαρμόζοντας την *συνέπεια υπέρ-τόξων* σε αυτόν τον περιορισμό θα πρέπει να καθορίσει ποιες τιμές της μεταβλητής X είναι επιτρεπτές. Θεωρώντας την τιμή *2* για παράδειγμα, είναι συνεπής μόνο στην περίπτωση που υπάρχει λύση για τον περιορισμό $2 = 3Y + 5Z$ όπου $Y \in \{0,1,2\}$ και $Z \in \{-1,0,1,2\}$. Κάτι τέτοιο είναι σημαντικό και πολύ δύσκολο να ελεγχθεί. Εάν οι τιμές, οι οποίες δεν ικανοποιούν την *συνέπεια υπερ-τόξων*, αφαιρεθούν τότε προκύπτει ένα νέο πεδίο τιμών D_1 όπου $D_1(X)=\{3,5,6\}$, $D_1(Y)=\{0,1,2\}$και $D_1(Z)=\{0,1\}$.
Ένας πιο σύνθετος περιορισμός θα ήταν ο $12X + 107Y - 17Z + 5T = 2$ όπου τα αρχικά πεδία για όλες τις μεταβλητές είναι $\{0,1,\ldots,1000\}$. Μπορούμε να παρατηρήσουμε ότι εφαρμογή της *συνέπειας υπερ-τόξων* για τον παραπάνω περιορισμό είναι πολύ δαπανηρή και πόσο μάλλον για ένα αυθαίρετο περιορισμό όπου η εφαρμογή της *συνέπειας υπερ-τόξων* μπορεί να αποτελέσει NP-Hard πρόβλημα. Συνεπώς η *συνέπεια υπερ-τόξων* είναι πολύ δαπανηρή για να μπορεί να χρησιμοποιηθεί σε σχέση πάντα με τα οφέλη που απορρέουν από την χρήση της.

2.9.4 Συνέπεια ορίων (Bound-consistency)

Οι μέθοδοι που έχουν αναφερθεί έως τώρα μπορούν να χρησιμοποιηθούν στη επίλυση αριθμητικών CSP καθώς αποτελούν ένα συγκεκριμένο τύπο CSP. Πράγματι, το παράδειγμα 2.2 και αυτό του κρυπταριθμητικού αινίγματος μπορούν να επιλυθούν ως αριθμητικά CSP. Παρόλα αυτά, ο περιορισμός σε ακέραιους και αριθμητικούς περιορισμούς μας επιτρέπει να ορίσουμε ένα νέο είδος συνέπειας, την *συνέπεια ορίων*. Υπάρχουν δύο ιδέες γύρω από την *συνέπεια ορίων*. Η πρώτη είναι να προσεγγίσουμε το πεδίο τιμών μίας μεταβλητής ακέραιων χρησιμοποιώντας ανώτερο και κατώτερο όριο. Η δεύτερη είναι να χρησιμοποιήσουμε στους περιορισμούς, συνέπεια για πραγματικούς αριθμούς παρά συνέπεια για ακέραιους.

Ορισμός 2.5 [28]
Ένα σύνολο τιμών, *[l..u]*, αντιπροσωπεύει το σύνολο ακέραιων αριθμών *{l,l+1,...,u}*εάν l≤ u, διαφορετικά αντιπροσωπεύει το κενό σύνολο.
Εάν *D* είναι ένα πεδίο από ακεραίους, τότε $min_D(x)$ είναι το ελάχιστο στοιχείο του *D(x)* και $max_D(x)$ είναι το μέγιστο στοιχείο του *D(x)*.

Ορισμός 2.6 [28]
Ένας αριθμητικός περιορισμός *c* ικανοποιεί την συνέπεια ορίων για ένα πεδίο *D* εάν για κάθε μεταβλητή *x* που μετέχει στον *c*, υπάρχουν :

- ανάθεση πραγματικών αριθμών, $d_1,d_2,...,d_k$, για τις υπόλοιπες μεταβλητές του *c*, $x_1,x_2,...,x_k$, τέτοια ώστε $min_D(x_j) \le d_j \le max_D(x_j)$ για κάθε d_j και $\{x \rightarrow min_D(x), x_1 \rightarrow d_1, ..., x_k \rightarrow d_k\}$ να αποτελεί λύση του *c* και
- ανάθεση πραγματικών αριθμών, $d'_1,d'_2,...,d'_k$, για τις $x_1,x_2,...,x_k$, τέτοια ώστε $min_D(x_j) \le d'_j \le max_D(x_j)$ για κάθε d'_j και $\{x \rightarrow max_D(x), x_1 \rightarrow d'_1, ..., x_k \rightarrow d'_k\}$ να αποτελεί λύση του *c*.

Ένα CSP με περιορισμό $c_1 \wedge ... \wedge c_n$ και πεδίο *D* ικανοποιεί την συνέπεια ορίων εάν κάθε περιορισμός c_i ικανοποιεί την συνέπεια ορίων ως προς το *D* για $1 \le i \le n$.

Ας θεωρήσουμε το παράδειγμα, *X=3Y + 5Z* με πεδίο τιμών *D* όπου *D(X)={2..7}*, *D(Y)={0..2}*και *D(Z)={-1..2}*. Ο περιορισμός αυτός δεν ικανοποιεί την *συνέπεια ορίων* ως προς το *D*. Για να το αποδείξουμε, ας θεωρήσουμε ότι ο περιορισμός μπορεί να εκφραστεί ως *5Z = X – 3Y*, ο οποίος ισοδυναμεί με την προηγούμενη έκφραση του. Εάν τώρα η *Z* πάρει την μέγιστη της τιμή *2* , τότε το αριστερό μέλος της έκφρασης θα έχει τιμή *10*, αλλά η μέγιστη τιμή του αριστερού μέλους *X – 3Y* είναι *7 – 3×0 = 7*. Επομένως το σύνολο τιμών της *Z* πρέπει να αλλάξει έτσι ώστε να ικανοποιείται η *συνέπεια ορίων*. Το νέο πεδίο D_1 που προκύπτει είναι το $D_1(X)=\{2..7\}$, $D_1(Y)=\{0..2\}$και $D_1(Z)=\{0..1\}$, το οποίο ικανοποιεί την *συνέπεια ορίων*.
Ας θεωρήσουμε τώρα τον περιορισμό *X =Y + Z*. Ξαναγράφοντας τον περιορισμό σε τρεις διαφορετικές μορφές προκύπτει:

$$X = Y + Z,\ Y = X - Z \text{ και } Z = X - Y$$

Σύμφωνα με τον συλλογισμό που προηγούμενου παραδείγματος για τις μέγιστες και ελάχιστες τιμές της δεξιάς πλευράς των παραπάνω εκφράσεων προκύπτει ότι

$$\begin{aligned} X &\ge min_D(Y) + min_D(Z), & X &\le max_D(Y) + max_D(Z), \\ Y &\ge min_D(X) - max_D(Z), & Y &\le max_D(X) - min_D(Z), \\ Z &\ge min_D(X) - max_D(Y), & Z &\le max_D(X) - min_D(Y). \end{aligned}$$

Από αυτές τις ανισότητες μπορούμε να εξάγουμε κανόνες οι οποίοι να διασφαλίζουν την *συνέπεια ορίων*. Αν τώρα θεωρήσουμε τα πεδία *D(X)={4..8}, D(Y)={0..3}* και *D(Z)={2..2}*τότε σύμφωνα με τους παραπάνω κανόνες θα προκύψει ότι

$$\begin{aligned} 2 \le X \le 5, \\ 2 \le Y \le 6, \\ 1 \le Z \le 8 \end{aligned}$$

και επομένως ανανεώνοντας το πεδίο τιμών προκύπτει ότι $D(X)=\{4..5\}$, $D(Y)=\{2..3\}$ και $D(Z)=\{2..2\}$ χωρίς να αφαιρείται κάποια από τις λύσεις του περιορισμού.

Σύμφωνα τώρα με τους κανόνες που χρησιμοποιήσαμε και το νέο πεδίο τιμών προκύπτει ότι

$$4 \le X \le 5,$$
$$2 \le Y \le 3,$$
$$1 \le Z \le 3$$

Το πεδίο τιμών ικανοποιεί όλους τους περιορισμούς και ο περιορισμός $X = Y + Z$ ικανοποιεί την *συνέπεια ορίων* ως προς το πεδίο.

Μία ενδιαφέρουσα περίπτωση περιορισμού ο οποίος εκφράζει γραμμική ανισότητα είναι ο $Y \neq Z$. Στην περίπτωση αυτή οι απαιτήσεις για *συνέπεια ορίων* είναι αρκετά ασθενής καθώς από την στιγμή που κάθε μεταβλητή έχει ένα πεδίο με δύο ή περισσότερες πιθανές τιμές τότε η ανισότητα ικανοποιεί την *συνέπεια ορίων*. Παρόλα αυτά όταν μία μεταβλητή, έστω η Z, έχει μία σταθερή τιμή, δηλαδή $min_D(Z) \equiv max_D(Z)$, τότε ισχύει ότι $Y \neq min_D(Z)$ και ουσιαστικά $Y \neq max_D(Z)$. Η μόνη περίπτωση η ανισότητα να μην ικανοποιεί την *συνέπεια ορίων* είναι όταν η ελάχιστη ή η μέγιστη τιμή της Y να είναι ίση με την σταθερή τιμή της Z. Σε αυτήν την περίπτωση αφαιρούμε την τιμή αυτή από το πεδίο της Y και αυξάνουμε ή μειώνουμε το αντίστοιχο όριο. Για παράδειγμα, έστω ότι οι μεταβλητές έχουν πεδία $D(Y)=\{2..3\}$και $D(Z)=\{2..2\}$, τότε η ελάχιστη τιμή 2 της Y δεν επιτρέπετε από τον περιορισμό. Για τον λόγο αυτό ενημερώνουμε το πεδίο και το νέο πεδίο που προκύπτει είναι $D(Y)=\{3..3\}$και $D(Z)=\{2..2\}$.

Εντούτοις, υπάρχουν κάποιοι μη γραμμικοί περιορισμοί για τους οποίους είναι δύσκολο να εξάγεις κανόνες διάδοσης περιορισμών. Ας πάρουμε για παράδειγμα την περίπτωση του περιορισμού $X = Y \times Z$. Ας θεωρήσουμε τώρα ότι όλοι οι αριθμοί είναι αυστηρώς θετικοί, στην περίπτωση αυτή οι κανόνες διάδοσης περιορισμών μπορούν να βασιστούν στις ανισότητες

$$X \ge min_D(Y) \times min_D(Z),$$
$$X \le max_D(Y) \times max_D(Z),$$
$$Y \ge min_D(X) / max_D(Z),$$
$$Y \le max_D(X) / min_D(Z),$$
$$Z \ge min_D(X) / max_D(Y),$$
$$Z \le max_D(X) / min_D(Y)$$

Αν τώρα θεωρήσουμε το πεδίο $D(X)=\{1..4\}$, $D(Y)=\{1..2\}$και $D(Z)=\{1..4\}$, τότε από τις ανισότητες προκύπτει ότι

$$X \ge 1,\ X \le 8,\ Y \ge 1,\ Y \le 4,\ Z \ge 1,\ Z \le 4$$

Παρατηρούμε ότι κανόνες βασιζόμενοι στις ανισότητες δεν επιφέρουν καμία αλλαγή στο πεδίο τιμών. Με την πιθανότητα όμως οι μεταβλητές να πάρουν την τιμή μηδέν, η εξαγωγή κανόνων γίνετε πιο πολύπλοκη.

Γενικά μπορούμε να υπολογίσουμε τα όρια της X εξετάζοντας όλους τους πιθανούς συνδυασμούς των ακραίων τιμών για τις Y και Z. Συνεπώς,

$$X \ge minimun\ \{min_D(Y) \times min_D(Z),\ min_D(Y) \times max_D(Z),$$
$$max_D(Y) \times min_D(Z),\ max_D(Y) \times max_D(Z)\},$$
$$X \le maximun\ \{min_D(Y) \times min_D(Z),\ min_D(Y) \times max_D(Z),$$
$$max_D(Y) \times min_D(Z),\ max_D(Y) \times max_D(Z)\},$$

Θεωρώντας τώρα το πεδίο τιμών $D(X)=\{4..8\}$, $D(Y)=\{0..3\}$ και $D(Z)=\{-2..2\}$ προκύπτει ότι

$$X \geq minimun\ \{0\ ,\ 0\ ,-\ 6\ ,\ 6\} = -\ 6,$$
$$X \leq maximun\ \{0\ ,\ 0\ ,-\ 6\ ,\ 6\} = \ 6$$

Οι κανόνες διάδοσης περιορισμών για τις υπόλοιπες μεταβλητές είναι πιο περίπλοκοι καθώς οι τιμές μπορούν να πάρουν τιμές κοντά στο μηδέν. Δεδομένου των $D(X)=\{4..8\}$ και $D(Z)=\{-2..2\}$,οποιαδήποτε μη μηδενική τιμή d της Y, είναι συνεπής ως προς την *συνέπεια ορίων* καθώς η ανάθεση $\{X\rightarrow 4,\ Y\rightarrow d,\ Z\rightarrow(4/d)\}$αποτελεί λύση του $X = Y \times Z$ καθώς $-2 \leq 4/d \leq 2$ για κάθε d όπου $|d| \geq 2$.

Επιπλέον η ανανέωση της Y είναι πολύπλοκη, καθώς για την Z ισχύει ότι $min_D(Z) < 0$ και $max_D(Z) > 0$ και επομένως, οι αναθέσεις τιμών που χρησιμοποιούνται για τον έλεγχο της ικανοποίησης της *συνέπειας ορίων* για τη Y, μπορούν να περιέχουν αυθαίρετα μικρές θετικές και αρνητικές τιμές πραγματικών αριθμών για την Z. Αυτό σημαίνει ότι ο περιορισμός του πολλαπλασιασμού ικανοποιεί την *συνέπεια ορίων* για οποιαδήποτε τιμή της Y. Για τον λόγο αυτό όσο θα ισχύει ότι $min_D(Z) < 0$ και $max_D(Z) > 0$, η *συνέπεια ορίων* δεν μπορεί να χρησιμοποιηθεί για να μειώσει το πεδίο τιμών της Y. Αν τώρα δεν είχαμε λάβει υπόψη μας την συνθήκη, τότε για τις X και Z θα προέκυπταν οι ανισότητες

$$Y \geq minimun\ \{min_D(X) / min_D(Z)\ ,\ min_D(X) / max_D(Z),$$
$$max_D(X) / min_D(Z),\ max_D(X) / max_D(Z)\},$$
$$Y \leq maximun\ \{min_D(X) / min_D(Z)\ ,\ min_D(X) / max_D(Z),$$
$$max_D(X) / min_D(Z),\ max_D(X) / max_D(Z)\}$$

όπου η διαίρεση ενός θετικού αριθμού με το μηδέν δίνει ∞ και η διαίρεση ενός αρνητικού αριθμού με το μηδέν δίνει $-\infty$. Η διαίρεση $0/0$ θεωρούμε ότι δίνει $-\infty$ για το δεξί μέλος της ανισότητας $\geq$ και ∞ για το δεξί μέλος της ανισότητας $\leq$. Χρησιμοποιώντας τον ίδιο συλλογισμό, οι ανισότητες που προκύπτουν για την Z όταν $min_D(Y) \geq 0$ και $max_D(Y) \leq 0$, είναι

$$Z \geq minimun\ \{min_D(X) / min_D(Y)\ ,\ min_D(X) / max_D(Y),$$
$$max_D(X) / min_D(Y),\ max_D(X) / max_D(Y)\},$$
$$Z \leq maximun\ \{min_D(X) / min_D(Y)\ ,\ min_D(X) / max_D(Y),$$
$$max_D(X) / min_D(Y),\ max_D(X) / max_D(Y)\}$$

2.9.5 Γενικευμένη συνέπεια (Generalized consistency)

Μία αδυναμία των βασισμένων στην συνέπεια προσεγγίσεων είναι ότι κάθε περιορισμός εξετάζεται ανεξάρτητα από την ύπαρξη άλλων. Κάποιες φορές η γνώση για τους άλλους περιορισμούς μπορεί να βελτιώσει δραματικά την περικοπή τιμών των πεδίων. Για τον λόγο αυτό, είναι πολύ συνηθισμένη η χρήση "σύνθετων" περιορισμών, οι οποίοι αποτελούνται από την σύζευξη απλούστερων περιορισμών, με εξειδικευμένους κανόνες διάδοσης περιορισμών.

2.9.5.1 Ο Περιορισμός alldifferent [28]

Ας θεωρήσουμε τον εξειδικευμένο περιορισμό alldifferent($\{V_1,...,V_2\}$) ο οποίος ικανοποιείται όταν οι μεταβλητές $V_1,...,V_2$ του ορίσματος έχουν διαφορετικές τιμές. Αντί

αυτού του περιορισμού θα μπορούσαμε να χρησιμοποιήσουμε την σύζευξη περιορισμών ανισοτήτων. Για παράδειγμα ο περιορισμός alldifferent(*{X,Y,Z}*) θα μπορούσε να αντικατασταθεί από τον περιορισμό $X \neq Y \wedge X \neq Z \wedge Y \neq Z$. Το μειονέκτημα αυτής της έκφρασης είναι ότι η μέθοδος της *συνέπειας τόξων* για αυτόν τον περιορισμό είναι πολύ αδύναμη, καθώς κάθε ανισότητα εξετάζεται ανεξάρτητα από τις υπόλοιπες και όχι σε συνδυασμό με αυτές. Για παράδειγμα, ο περιορισμός $X \neq Y \wedge X \neq Z \wedge Y \neq Z$ με πεδίο *D(X)={1,2}, D(Y)={1,2}*και *D(Z)={1,2}* δεν έχει λύσεις, καθώς υπάρχουν μόνο δύο πιθανές τιμές για τις τρεις μεταβλητές γεγονός το οποίο η μέθοδος της *συνέπειας τόξων* δεν μπορεί να ανιχνεύσει.

Οι εξειδικευμένοι κανόνες διάδοσης περιορισμών του alldifferent μπορούν να αντιμετωπίσουν τέτοιες καταστάσεις και να βρίσκουν την αποτυχία ύπαρξης λύσης πιο εύκολα. Βασικά ο αλγόριθμος του alldifferent λειτουργεί ως εξής, αφαιρεί τις τιμές των σταθερών μεταβλητών από τις υπόλοιπες μεταβλητές και στη συνέχεια ελέγχει εάν ο αριθμός των μη σταθερών μεταβλητών δεν είναι μεγαλύτερος του συνολικού αριθμού τιμών που έχουν απομείνει για τις μη σταθερές μεταβλητές.

Για παράδειγμα, έστω ο περιορισμός alldifferent(*{X, Y , Z}*) και το πεδίο *D(X)={1,2}, D(Y)={1,2}* και *D(Z)={1,2}*. Ο αλγόριθμος θα υπολογίσει ότι ο αριθμός των μη σταθερών μεταβλητών είναι *3* και ο αριθμός των τιμών είναι *2*. Αυτομάτως θα επιστρέψει ότι δεν υπάρχει λύση.

Παρατηρούμε όμως ότι ο αλγόριθμος δεν είναι πλήρης. Για παράδειγμα, ας θεωρήσουμε τον περιορισμό alldifferent(*{X,Y,Z,T}*) με πεδίο *D(X)={1,2}, D(Y)={1,2}, D(Z)={1,2}* και *D(T)={2,3,4,5}*. Δεν υπάρχει ανάθεση τιμών τέτοια ώστε να ικανοποιείται ο περιορισμός, γεγονός όμως το οποίο αποκρύπτεται από το μεγάλο πεδίο της *T*.

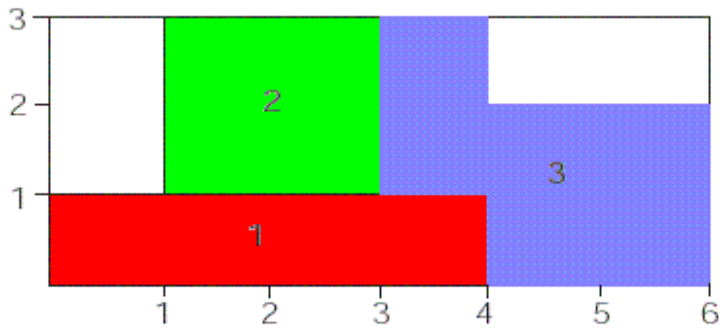
2.9.5.2 Ο Περιορισμός cumulative [28]

Ένα άλλο παράδειγμα σύνθετου περιορισμού, ο οποίος είναι πολύ χρήσιμος σε διάφορες πρακτικές εφαρμογές, είναι ο cumulative. Ο περιορισμός αυτός εισήχθηκε για να λύσει προβλήματα χρονικού προγραμματισμού (scheduling) και τοποθέτησης (placement). Ο περιορισμός

$$\text{cumulative}([S_1,...,S_m],[D_1,...,D_m],[R_1,...,R_m],L)$$

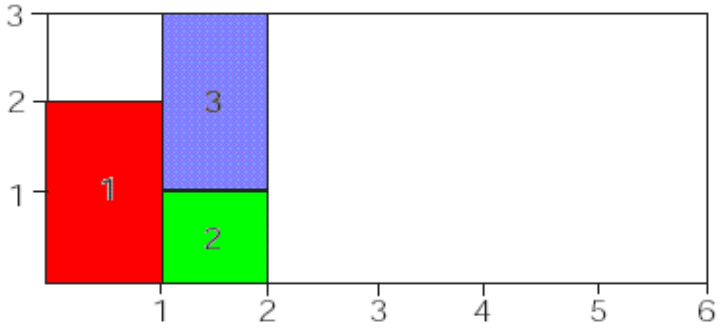
περιορίζει τις μεταβλητές ώστε να ικανοποιείται ένα απλό πρόβλημα χρονικού προγραμματισμού. Υπάρχουν m εργασίες για προγραμματισμό με χρονικές στιγμές έναρξης $S_1,..,S_m$ και διάρκεια $D_1,...,D_m$, οι οποίες απαιτούν $R_1,...,R_m$ μονάδες ενός μόνο πόρου. Το πολύ L μονάδες του πόρου είναι διαθέσιμες κάθε χρονική στιγμή. Οι μέθοδοι συνέπειας για τον περιορισμό cumulative που έχουν αναπτυχθεί είναι πολύ πολύπλοκες καθώς υπάρχουν πολυάριθμες πιθανότητες εξαγωγής νέας πληροφορίας εξαρτώμενες από το ποιες μεταβλητές είναι γνωστές.

Ας μελετήσουμε τρεις βασικές εφαρμογές του περιορισμού cumulative όπως παρουσιάζονται στο [6]. Ο οριζόντιος άξονας και στις τρεις περιπτώσεις αντιστοιχεί στις χρονικές περιόδους, ενώ ο κάθετος άξονας στις μονάδες του πόρου που καταναλώνονται κάθε χρονική περίοδο.



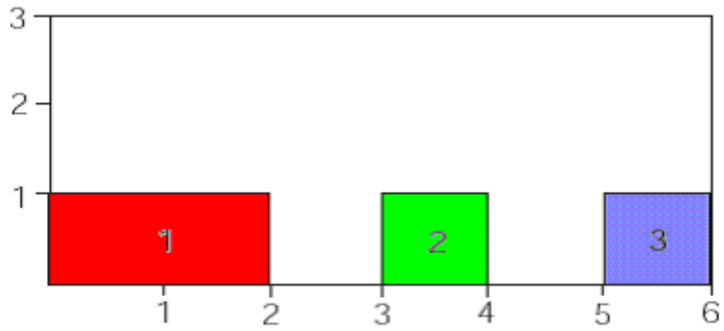
Σχήμα 2.8 : cumulative([1,2,4],[4,2,3],[1,2,2],3)

1) Στο σχήμα 2.8, υπάρχουν τρεις εργασίες οι οποίες πρέπει να προγραμματιστούν: η εργασία *1* απαιτεί μία μονάδα του πόρου και έχει διάρκεια τεσσάρων συνεχόμενων χρονικών περιόδων, οι εργασίες *2* και *3* απαιτούν δύο μονάδες του πόρου και έχουν διάρκεια δύο και τριών αντίστοιχα χρονικών περιόδων. Κάθε χρονική στιγμή το συνολικό ποσό των μονάδων του πόρου που καταναλώνονται πρέπει να είναι μικρότερο ή ίσο του τρία.



Σχήμα 2.9 : cumulative([1,2,2],[1,1,1],[2,1,2],3)

2) Στο σχήμα 2.9, οι χρονικές διάρκειες όλων των εργασιών είναι ίσες με ένα. Αυτή η συγκεκριμένη περίπτωση αντιστοιχεί στο πρόβλημα της συσκευασίας κουτιών (bin-packing) όπου *m* κουτιά σταθερής χωρητικότητας και *n* αντικείμενα σταθερού μεγέθους πρέπει να καταμερισθούν στα κουτιά.



Σχήμα 2.10 : cumulative([1,4,6],[2,1,1],[1,1,1],1)

3) Στο παράδειγμα του σχήματος 2.10, κάθε χρονική στιγμή το συνολικό ποσό των μονάδων του πόρου που καταναλώνονται πρέπει να είναι ίσο του ένα. Αυτό αντιστοιχεί σε πρόβλημα χρονικού προγραμματισμού όπου οι εργασίες δεν μπορούν να συμπίπτουν χρονικά καθώς μοιράζονται τον ίδιο πόρο με χωρητικότητα μιας μονάδας.

2.9.5.3 Ο Περιορισμός element [28]

Το τελευταίο παράδειγμα σύνθετου περιορισμού είναι ο περιορισμός element. Ο περιορισμός element μιμείται την συμπεριφορά ενός πίνακα. Ο περιορισμός

$$\text{element}(I, [V_1, \ldots, V_m], X)$$

υποστηρίζει την σχέση ότι, εάν $I = i$, τότε $X = V_i$. Μπορούμε να το κατανοήσουμε, θεωρώντας την λίστα $[V_1, \ldots, V_m]$ ως έναν πίνακα υ, όπου $X = \upsilon[I]$.

Για παράδειγμα, ας θεωρήσουμε ότι η μεταβλητή X πρέπει να είναι ίση με $I^2 - 3$ όπου το πεδίο τιμών της I είναι $[0..5]$. Αυτό μπορεί να περιγραφεί από τον περιορισμό element(I, $[-3, -2, 1, 6, 13, 22]$, X). Οι τεχνικές συνέπειας για τον περιορισμό element μπορούν να μειώσουν το πεδίο της I σύμφωνα με την πληροφορία της X και το αντίστροφο. Για παράδειγμα, σύμφωνα με το παραπάνω περιορισμό element εάν το πεδίο της X είναι ίσο με $[-20..20]$ τότε αυτόματα το πεδίο της X θα μειωθεί στις τιμές $\{-3,-2, 1,6,13\}$ και το πεδίο της I στις τιμές $[0..4]$ έτσι ώστε να υπάρχει συνέπεια.

2.10 Γλώσσες λογικού προγραμματισμού με περιορισμούς (Constraint Logic Programming, CLP)

Ο προγραμματισμός με περιορισμούς αποτελεί μία από τις συναρπαστικότερες εξελίξεις στις γλώσσες προγραμματισμού της τελευταίας δεκαετίας. Βασιζόμενος σε ένα ισχυρό θεωρητικό υπόβαθρο, εξελίσσεται ως μία από τις μεθόδους μοντελοποίησης διαφόρων τύπου προβλημάτων βελτιστοποίησης και κυρίως εκείνων στα οποία υπεισέρχονται ετερογενείς περιορισμοί και πολύπλοκη αναζήτηση.

Προγενέστερες γλώσσες προγραμματισμού απεικόνιζαν στενά την βαθύτερη φυσική αρχιτεκτονική του υπολογιστή. Μέχρι τότε, η κύρια κατεύθυνση στο σχεδιασμό

γλωσσών προγραμματισμού ήταν να δοθεί στον προγραμματιστή η ελευθερία να καθορίσει αντικείμενα και διαδικασίες, τα οποία αντιστοιχούν σε οντότητες και λειτουργίες του πεδίου εφαρμογής. Οι αντικειμενοστραφείς γλώσσες εξασφαλίζουν καλούς μηχανισμούς για τον καθορισμό στοιχείων του προγράμματος που να προσεγγίζουν την συμπεριφορά των οντοτήτων σε ένα καθορισμένο πρόβλημα. Ωστόσο, παραδοσιακές γλώσσες προγραμματισμού εξασφαλίζουν μικρή υποστήριξη για τον καθορισμό σχέσεων ή περιορισμών, μεταξύ των καθορισμένων οντοτήτων του προγράμματος. Είναι ο ρόλος του προγραμματιστή να διατηρήσει ρητά αυτές τις σχέσεις και να βρει αντικείμενα τα οποία τις ικανοποιούν.

Για παράδειγμα, ας σκεφτούμε το νόμο του Ωμ, $V = I \times R$, ο οποίος περιγράφει την σχέση μεταξύ της τάσης ρεύματος V, της έντασης I και της αντίστασης R σε ένα αντιστάτη. Σε μία παραδοσιακή γλώσσα προγραμματισμού, ο προγραμματιστής δεν μπορεί να χρησιμοποιήσει την σχέση ως έχει, αντιθέτως θα πρέπει κάθε φορά να συμπεριλαμβάνει δηλώσεις που υπολογίζουν την τιμή της μίας μόνο μεταβλητής, ενώ οι υπόλοιπες δύο πρέπει να έχουν ήδη τιμή. Έτσι η I μπορεί να υπολογιστεί από τις V και R, χρησιμοποιώντας την σχέση $I: = V/R$. Εάν όμως πρέπει να υπολογιστεί η τιμή της R συναρτούμενη των άλλων δύο μεταβλητών τότε απαιτείται η δήλωση μίας διαφορετικής σχέσης $R: = V/ I$.

Είναι λογικό να απαιτείται, από τον προγραμματιστή να δηλώσει με σαφήνεια τις σχέσεις μεταξύ των αντικειμένων του προγράμματος, κυρίως για εφαρμογές στις οποίες οι σχέσεις είναι απλές και στατικές. Παρόλα αυτά, για πολλές εφαρμογές, το κεντρικό σημείο δυσκολίας του προβλήματος είναι η μοντελοποίηση των σχέσεων και η εύρεση αντικειμένων που θα τις ικανοποιούν. Για τον λόγο αυτό, από τα τέλη του 60', είχε εκφραστεί ενδιαφέρων για γλώσσες προγραμματισμού οι οποίες επιτρέπουν στον προγραμματιστή απλά να δηλώνει τις σχέσεις μεταξύ των αντικειμένων. Είναι ο ρόλος της βαθύτερης υλοποίησης του συστήματος να εξασφαλίζει ότι αυτές οι σχέσεις ή περιορισμοί ισχύουν. Τέτοιου είδους γλώσσες ονομάζονται *γλώσσες λογικού προγραμματισμού με περιορισμούς*.

Στην βιβλιογραφία έχει συμφωνηθεί ότι ένας αλγόριθμος αποτελείται από δύο κύρια στοιχεία, την λογική και τον έλεγχο [2]. Η λογική είναι υπεύθυνη για το τι κάνει ο αλγόριθμος, ενώ ο έλεγχος για το πώς το κάνει. Μία ιδανική μεθοδολογία προγραμματισμού θα έπρεπε αρχικά να ενδιαφερθεί για την λογική (τι επιθυμεί να υπολογίσει) και έπειτα για τον έλεγχο (πως επιτυγχάνεται μία λύση). Η ανωτερότητα του *λογικού προγραμματισμού* προέρχεται από το γεγονός ότι παρέχει τα μέσα ώστε να διαχωρίζονται οι δύο παραπάνω αρχές.

Ο *λογικός προγραμματισμός* είναι απόλυτα εφαρμόσιμος στην επίλυση προβλημάτων αναζήτησης στα οποία περιέχονται οντότητες που συσχετίζονται μέσω συγκεκριμένων περιορισμών. Ένας εσωτερικός μηχανισμός ανάλυσης (resolution) και συμπερασμού (inference) χρησιμοποιείτε για την ανεύρεση λύσης. Αυτό οδηγεί σε μία κατά-βάθος αναζήτηση στο χώρο του προβλήματος, η οποία είναι γνωστή ως στρατηγική παραγωγής και ελέγχου (generate-and-test). Οι περιορισμοί λειτουργούν ως έλεγχος, εξαλείφοντας από το χώρο αναζήτησης διαδρομές οι οποίες οδηγούν σε αδιέξοδα. Παρόλα αυτά, αυτό μπορεί να μην οδηγεί σε μία αποδεκτή αποδοτικότητα για μεγάλα προβλήματα.

Ο πραγματικός σκοπός του λογικού προγραμματισμού με περιορισμούς είναι να εμπλουτίσει το λογικό προγραμματισμό με τέτοιο τρόπο ώστε να διατηρούνται όλες οι μοναδικές του ιδιότητες, ενώ εισάγεται ένας νέος μηχανισμός που αφορά την επιβολή και λύση των περιορισμών. Αντικειμενικός σκοπός είναι ο εμπλουτισμός του *λογικού προγραμματισμού* με τρόπο ώστε η αποδοτικότητα να αποτελεί τον κύριο στόχο. Αυτού

του είδους η φιλοσοφία είναι γνωστή ως CLP(X) σχήμα, όπου το X μπορεί να ανήκει σε ένα οποιοδήποτε πιθανό σύνολο οντοτήτων, σύμφωνα με την εφαρμογή που επιθυμούμε να αναπτύξουμε. Για παράδειγμα, εάν το πρόγραμμα χειρίζεται πραγματικούς αριθμούς τότε χρησιμοποιούμε το CLP($\Re$)σχήμα.

Ο απαιτούμενος εμπλουτισμός για τον λογικό προγραμματισμό με περιορισμούς προέρχεται από την δουλειά που έχει γίνει στην περιοχή των τεχνικών διάδοση περιορισμού και συνέπειας. Πρωταρχικός σκοπός αυτών των μεθόδων είναι να αποφθεχθεί το παράδειγμα της παραγωγής και ελέγχου και ο μείωση του χώρου αναζήτησης να επιτυγχάνεται μέσω της επίλυσης των περιορισμών που έχουν επιβληθεί. Προς αυτήν την κατεύθυνση, απαιτείται λιγότερη προσπάθεια για την εύρεση λύση, γιατί το σύστημα από μόνο του αναλαμβάνει να απορρίψει διαδρομές στο χώρο αναζήτησης, αποφασίζοντας ότι δεν οδηγούν σε λύση, κάποιες φορές πριν ακόμα ξεκινήσει η διαδικασία αναζήτησης. Αυτού του είδους η αναζήτηση ονομάζεται στρατηγική περιορισμού και παραγωγής (constraint and generate). Με την εισαγωγή του λογικού προγραμματισμού με περιορισμούς, η λύση μεγάλης κλάσης προβλημάτων έγινε πιο εφικτή. Σε αυτού του είδους τα προβλήματα εμπεριέχονται και τα προβλήματα χρονικού προγραμματισμού αλλά και κατάρτισης ωρολογίων προγραμμάτων.

2.10.1 Το σύστημα της ECL^{i}PS^e

Το σύστημα της ECL^{i}PS^e αποτελεί ένα παράδειγμα γλώσσας λογικού προγραμματισμού με περιορισμούς. Έχει αναπτυχθεί στο Ευρωπαϊκό ερευνητικό κέντρο βιομηχανίας υπολογιστή, ECRC (European Computer - Industry Research Center), και αποτελεί διάδοχο του συστήματος CHIP, της πρώτης γλώσσας λογικού προγραμματισμού με περιορισμούς για επίλυση προβλημάτων με πεπερασμένα πεδία.

Η ECL^{i}PS^e αποτελεί ουσιαστικά ένα Prolog σύστημα, ενισχυμένο με τις διάφορες επεκτάσεις που παρέχουν μία ικανοποιητική ευελιξία για την αντιμετώπιση διαφορετικών προβλημάτων. Αυτή η ευελιξία προέρχεται από έναν ιδιαίτερο τύπο στοιχείων που υποστηρίζεται από τη γλώσσα και ονομάζεται μετα-όρος (metaterm). Αυτός ο τύπος στοιχείων παρέχει την βάση για την ανάπτυξη διαφόρων βιβλιοθηκών. Παραδείγματα αποτελούν οι βιβλιοθήκες για περιγραφή περιορισμών όπως η βιβλιοθήκη για πεπερασμένα πεδία τιμών fd (Finite Domains), η βιβλιοθήκη fd_global και η βιβλιοθήκη edge_finder. Οι παραπάνω βιβλιοθήκες έχουν χρησιμοποιηθεί εκτενέστατα όσο αφορά το μέρος της υλοποίησης σε αυτή την διπλωματική και άρα θα ήταν θεμιτό να αναφερθούμε στην λειτουργικότητά τους.

Η βιβλιοθήκη fd παρέχει έναν επιλυτή για πεπερασμένα πεδία (finite-domain solver) που μπορεί να εφαρμόζει περιορισμούς μόνο πάνω σε ακέραιους, σε πεπερασμένα πεδία μη αριθμητικών τιμών όπως {red,blue,green} και σε μαθηματικές εκφράσεις περιορισμένου τύπου. Για παράδειγμα, ο επιλυτή μπορεί να χειριστεί μόνο γραμμικές εκφράσεις όπως $2*X - Z \#< X - 3$. Ο αλγόριθμος που χρησιμοποιείται για διατήρηση της συνέπεια είναι της *συνέπειας ορίων*, ο οποίος παρουσιάζει πλεονέκτημα σε σχέση με την *συνέπεια τόξων* σε περιπτώσεις γραμμικών εκφράσεων όπου οι μεταβλητές είναι περισσότερες από δύο, μειονεκτεί όμως σε εκφράσεις του τύπου $X \neq Y$ καθώς περιορίζει τα όρια του πεδίου κάθε μεταβλητής αλλά αδυνατεί να εξαλείψει ενδιάμεσες τιμές.

Η βιβλιοθήκη fd υποστηρίζει τον περιορισμό ενός πεδίου τιμών *#::* μιας μεταβλητής πεδίου, ο οποίος δίνει ένα άνω και κάτω όριο στις τιμές της μεταβλητής. Για παράδειγμα η δήλωση *X #:: 1..5, Y #:: [red,blue,green], Z #::[1,3,5..8]* ορίζει ότι οι μεταβλητές *X, Y*

και *Z* θα έχουν αντίστοιχα πεδία τιμών τα *{1,2,3,4,5}, {red,blue,green}* και *{1,3,5,6,7,8}*. Υποστηρίζει αριθμητικές εκφράσεις όπως ισότητα και ανισότητα, όπου συντακτικά προηγείται μία δίεση *#,* π.χ. *#=, #>, #<,##,#<=,#>=*. Οι συζεύξεις και διαζεύξεις περιορισμών παρέχονται μέσω των κατηγορημάτων *#/* και *#\/*. Τέλος υποστηρίζει τον περιορισμό element(*I,List,X*) των οποίο έχουμε περιγράψει στην Παράγραφο 2.8.5.3.

Η βιβλιοθήκη fd_global υποστηρίζει μία πληθώρα περιορισμών, κάθε ένας από τους οποίους παίρνει σαν όρισμα μία λίστα από μεταβλητές πεπερασμένου πεδίου ακαθόριστου μήκους. Τέτοιοι περιορισμοί αποκαλούνται "global" περιορισμοί . Παραδείγματα τέτοιων περιορισμών είναι οι alldifferent/1, maxlist/2, occurrences/3 και sorted/2.

Η βιβλιοθήκη edge finder υποστηρίζει "global" περιορισμούς για εφαρμογές χρονικού προγραμματισμού. Οι περιορισμοί παίρνουν μία λίστα από διεργασίες (χρονικές στιγμές έναρξης, διάρκεια και απαιτούμενοι πόροι) και το μέγιστο επίπεδο χρησιμοποιήσιμων πόρων. Παραδείγματα τέτοιων περιορισμών είναι οι cumulative/4 και disjunctive/2.

Για μία γλώσσα προγραμματισμού με περιορισμούς όπως η ECL^{i}PS^e, υπάρχει μία συγκεκριμένη στρατηγική μέσω της οποίας τα προβλήματα ικανοποίησης περιορισμών πρέπει να αντιμετωπίζονται. Αυτή η τακτική αποτελείτε από τρία βασικά βήματα:

- Καθορισμός των μεταβλητών και των πεδίων τους. Αυτές οι μεταβλητές αναπαριστούν τις οντότητες του προβλήματος.
- Επιβολή των περιορισμών πάνω στις μεταβλητές οι οποίοι καθορίζουν τις σχέσεις μεταξύ τους. Αυτοί ουσιαστικά οι περιορισμοί χρησιμοποιούνται για να περιορίσουν τον χώρο αναζήτησης και επιπλέον λειτουργούν ως όρια του χώρου μέσα στον οποίο περιορίζουν τις λύσεις των προβλημάτων υπό εξέταση.
- Έναρξη της διαδικασίας ανάθεσης τιμών στις μεταβλητές η οποία σε συνεργασία με τον εσωτερικό μηχανισμό διάδοσης περιορισμού της Prolog επιστρέφει τις λύσεις του προβλήματος όταν αυτές υπάρχουν. Το βήμα αυτό μπορεί να ονομαστεί και μαρκάρισμα (labeling).

2.10.2 Άλλα συστήματα

Άλλα συστήματα τα οποία έχουν αναπτυχθεί και που υποστηρίζουν λογικό προγραμματισμό με περιορισμούς είναι τα εξής:

- **CHIP :** αποτελεί τον πρόδρομο των περισσότερων εμπορικών συστημάτων για περιορισμούς. Στερείται ευελιξίας για στρατηγικές αναζήτησης και εποικοδομητικών διαζεύξεων. Παρόλα αυτά αποτελεί ένα επιτυχημένο εμπορικό εργαλείο. [9]

- **clp(FD) :** αποτελεί μία γλώσσα προγραμματισμού για περιορισμούς γραμμένη σε C. Αποτελεί το γρηγορότερο σύστημα για πεπερασμένα πεδία, το οποίο διατίθεται ελεύθερο. Παρόλα αυτά παρέχει μόνο βασικά συστατικά του προγραμματισμού με περιορισμούς. [5]

- **ILOG SOLVER :** αποτελεί μία βιβλιοθήκη για προγραμματισμό με περιορισμούς, γραμμένη σε C++. [1]

- **OZ :** έχει αναπτυχθεί από το DFKI στο Saarbrücken της Γερμανίας. Αποτελεί μία γλώσσα προγραμματισμού η οποία υποστηρίζει συγχρόνως συναρτησιακό, αντικειμενοστραφή προγραμματισμό και προγραμματισμό με περιορισμούς.[5]

- **DOMLOG :** αποτελεί μία γλώσσα η οποία παρέχει ευριστικές καθορισμένες από τον χρήστη για μεθόδους αναζήτησης και παρέχει μεγάλη ευελιξία στην δήλωση περιορισμών. [7]

- **CHR (Constraint Handling Rules):** αποτελεί μία δηλωτική γλώσσα υψηλού επιπέδου η οποία είναι ειδικά σχεδιασμένη για την υλοποίηση επιλυτών περιορισμών (constraint solvers). Με την CHR μπορεί κάποιος να εισάγει περιορισμούς καθορισμένους από τον χρήστη σε μία άλλη γλώσσα, όπως η Prolog, η Lisp ή οποιαδήποτε άλλη γλώσσα. [16]

- **UNILANG :** αποτελεί μία γλώσσα η οποία έχει αναπτυχθεί ειδικά για αναπαράσταση προβλημάτων κατάρτισης ωρολογίων προγραμμάτων. Παρέχει υποστήριξη για διαφορετικά συγγενικά είδη προβλημάτων κατάρτισης ωρολογίου προγράμματος, όπως για σχολείο, για πανεπιστήμιο και εξεταστικής περιόδου. [17]

2.11 Περίληψη

Στο κεφάλαιο αυτό ασχοληθήκαμε με προβλήματα ικανοποίησης περιορισμών. Μελετήθηκαν τα είδη περιορισμών που υπάρχουν καθώς επίσης και αλγόριθμοι διάδοσης περιορισμών και των τεχνικών συνέπειας για τέτοιου είδους προβλήματα. Είδαμε πως η μέθοδος της *συνέπειας ορίων* υπερτερεί των άλλων μεθόδων σε περιπτώσεις γραμμικών περιορισμών για μεταβλητές πεπερασμένων πεδίων και επιπλέον πως μπορεί να αυξηθεί η απόδοση του συστήματος χρησιμοποιώντας μεθόδους γενικευμένης συνέπειας, κάνοντας χρήση "σύνθετων" περιορισμών όπως του cumulative, alldifferent και άλλων, σε ειδικές περιπτώσεις. Μελετήθηκαν οι ευριστικές διάταξης μεταβλητών και τιμών που μπορούν να χρησιμοποιηθούν σε αλγορίθμους αναζήτησης λύσης *κατά βάθος* ή ακόμα και συνδυασμό των παραπάνω. Επίσης παρατηρήσαμε πως οι *ευριστικές διάταξης μεταβλητών και τιμών* μπορούν να βοηθήσουν το σύστημα να προσεγγίζει ευκολότερα την λύση, στην περίπτωση που αυτή υπάρχει.

Τέλος έγινε μία παρουσίαση του συστήματος της ECL^{i}PS^e καθώς επίσης και της στρατηγικής που ακολουθείται για την αντιμετώπιση των προβλημάτων ικανοποίησης περιορισμών. Στην συνέχεια κάναμε μία σύντομη αναφορά σε παρόμοια εμπορικά συστήματα.

Κεφάλαιο 3

Μοντελοποίηση του Προβλήματος Κατάρτισης Ωρολογίου Προγράμματος ως Πρόβλημα Ικανοποίησης Περιορισμών

3.1 Εισαγωγή

Σκοπός αυτού του κεφαλαίου είναι η παρουσίαση του θεωρητικού μοντέλου που έχουμε αναπτύξει για το πρόβλημα της κατάρτισης ωρολογίου προγράμματος. Το μοντέλο που έχει αναπτυχθεί ανήκει στην ομάδα προβλημάτων ικανοποίησης περιορισμών, ενώ προκύπτουν οι εξής διαπιστώσεις:

- Το μοντέλο που θα παρουσιάσουμε επιλέχθηκε με βάση τους κανόνες λειτουργίας του Πολυτεχνείου Κρήτης.
- Το πρόγραμμα των μαθημάτων καθορίζεται πριν οι φοιτητές δηλώσουν μαθήματα. Αυτό σημαίνει ότι το πρόγραμμα δεν προκύπτει με βάση τις επιλογές των φοιτητών αλλά ούτε και με βάση των πραγματικό αριθμό των φοιτητών που παρακολουθεί το κάθε μάθημα. Παρόλα αυτά, γίνεται χρήση των στατιστικών στοιχείων προηγούμενων ετών για να καθοριστεί το μέγεθος του αριθμού φοιτητών που θα παρακολουθήσει το κάθε μάθημα, προσεγγιστικά. Αυτός είναι και ο κύριος λόγος για τον οποίο επιλέχθηκε μία περιγραφική αναπαράσταση του αριθμού φοιτητών για κάθε μάθημα (μικρός, μεσαίος ή μεγάλος) και όχι μία αριθμητική αναπαράσταση.
- Η ανάθεση των καθηγητών στις διαλέξεις είναι προκαθορισμένη από τους ίδιους τους καθηγητές του Πολυτεχνείου Κρήτης.
- Το πρόγραμμα που προκύπτει αφορά όλες τις σχολές του Πολυτεχνείου Κρήτης και συμπεριλαμβάνει όλα τα μαθήματα που προσφέρει ως ενιαίος φορέας.

3.2 Το πρόβλημα κατάρτισης ωρολογίου προγράμματος

- Το ωρολόγιο πρόγραμμα αποτελείται από D σε αριθμό ημέρες και η κάθε ημέρα έχει H σε αριθμό ώρες διδασκαλίας διάρκειας μιας ώρας η κάθε μία.
- Επομένως, οι χρονικές στιγμές στις οποίες θα διδάσκονται τα μαθήματα θα είναι $D * H$ σε αριθμό. Ας θεωρήσουμε το σύνολο $P = [1 ...,D*H]$ όπου $P \subseteq \mathbf{Z}$ υποσύνολο του συνόλου των ακεραίων περιέχει τις χρονικές στιγμές στις οποίες μπορεί να διδακτή οποιοδήποτε μάθημα.
- Το πανεπιστήμιο έχει καθηγητές, όπου $T=\{t_1 ,...,t_n\}$ το σύνολο των καθηγητών. Έστω $notavailable: T \times P \rightarrow \{true,false\}$ μία συνάρτηση η οποία απεικονίζει κάθε καθηγητή $t \in T$, αν είναι διαθέσιμος την χρονική στιγμή $p \in P$, επιστρέφοντας *true* ή *false* αναλόγως.

- Το πανεπιστήμιο έχει αίθουσες, όπου $R=\{r_1,...,r_n\}$ το σύνολο των αιθουσών. Έστω $notavailable : R \times P \rightarrow \{true,false\}$ μία συνάρτηση η οποία απεικονίζει κάθε αίθουσα $r \in R$, αν είναι διαθέσιμη την χρονική στιγμή $p \in P$, επιστρέφοντας *true* ή *false* αναλόγως. Μία αίθουσα διακρίνεται σε μικρή, μεσαία ή μεγάλη σε σχέση με την χωρητικότητα της. Έστω λοιπόν $capacity : R \rightarrow \{big, medium, small\}$ μία συνάρτηση η οποία απεικονίζει την χωρητικότητα κάθε αίθουσας $r \in R$, η οποία μπορεί να είναι *big (μεγάλη), medium (μεσαία)* ή *small (μικρή)*. Επίσης μία αίθουσα διακρίνεται σε αίθουσα διδασκαλίας ή εργαστηριακή σε σχέση με την χρηστικότητα.
- Το πανεπιστήμιο παρέχει στους φοιτητές του μαθήματα, όπου $C=\{c_1,...,c_n\}$ το σύνολο των μαθημάτων. Ένα μάθημα μπορεί να είναι υποχρεωτικό ή επιλογής καθώς επίσης παρακολουθείται από μικρό, μεσαίο ή μεγάλο αριθμό φοιτητών. Έστω λοιπόν $students_number : C \rightarrow \{big, medium, small\}$μία συνάρτηση η οποία απεικονίζει κάθε μάθημα $c \in C$ στον αριθμό φοιτητών που την παρακολουθούνε, ο οποίος μπορεί να είναι *big (μεγάλος), medium (μεσαίος) ή small (μικρός)*. Κάθε μάθημα αποτελείται από μία ή περισσότερες διαλέξεις.
- Θεωρώ ως $L=\{l_1,...,l_n\}$ το σύνολο των διαλέξεων όλων των μαθημάτων. Κάθε διάλεξη l έχει χρονική στιγμή έναρξης S και διάρκεια d, επομένως η χρονική στιγμή λήξης μίας διάλεξης μπορεί να υπολογιστεί μέσω του αθροίσματος $S + d$. Μία διάλεξη διακρίνεται σε διάλεξη, εργαστήριο ή ασκήσεις. Έστω $course:L \rightarrow C$ μία συνάρτηση η οποία απεικονίζει κάθε διάλεξη $l \in L$ στο αντίστοιχο μάθημα $c \in C$ στο οποίο ανήκει. Ομοίως, έστω $teacher: L \rightarrow T$ μία συνάρτηση η οποία απεικονίζει κάθε διάλεξη $l \in L$ στον αντίστοιχο καθηγητή $t \in T$ από τον οποίο διδάσκεται και έστω $room: L \rightarrow R$ μία συνάρτηση η οποία απεικονίζει κάθε διάλεξη $l \in L$ στην αντίστοιχη αίθουσα $r \in R$ στην οποία διδάσκεται.
- Θεωρώ ως $G=\{g_1,...,g_n\}$, $g_i \subseteq C$ το σύνολο των ομάδων των μαθημάτων. Τα μαθήματα της ίδιας ομάδας δεν μπορούν να διδάσκονται συγχρόνως, ενώ ένα μάθημα επιτρέπετε να ανήκει σε περισσότερες από μία ομάδες μαθημάτων.

3.3 Μεταβλητές και Πεδία ορισμού

Για κάθε διάλεξη l θεωρώ ότι η χρονική στιγμή έναρξης διδασκαλίας S αποτελεί μία μεταβλητή πεπερασμένου πεδίου, με πεδίο ορισμού το σύνολο των ακεραίων αριθμών $1,2,\ldots,D*H$ όπου D ο αριθμός των ημερών και H ο αριθμός των ωρών διδασκαλίας για κάθε ημέρα του προγράμματος.

Για παράδειγμα έστω ότι το πανεπιστήμιο λειτουργεί *5* ημέρες την εβδομάδα και *12* ώρες την ημέρα, τότε το πρόγραμμα που θα προκύψει θα έχει $5*12= 60$ χρονικές στιγμές στις οποίες μπορούν να ξεκινάνε τα μαθήματα.

Για κάθε διάλεξη l θεωρώ ότι η ανάθεση της αίθουσας γίνεται σε μία μεταβλητή VR η οποία έχει ως πεδίο ορισμού το σύνολο των ακεραίων αριθμών $1,2,\ldots,|R|$ όπου R το σύνολο των αιθουσών. Σε κάθε αριθμό αντιστοιχεί και μία αίθουσα. Για παράδειγμα, έστω ότι $R=\{r_1, r_2, r_3, r_4\}$ τότε κάθε μεταβλητή VR θα έχει ως πεδίο το σύνολο $\{1,2,3,4\}$ όπου η τιμή 1 θα αντιστοιχεί στην αίθουσα r_1, η τιμή 2 θα αντιστοιχεί στην αίθουσα r_2, κοκ.

Για κάθε διάλεξη l θεωρώ ότι η ανάθεσή του καθηγητή σε αυτήν είναι προκαθορισμένη βάση των κανόνων λειτουργίας του Πολυτεχνείο Κρήτης και επομένως δεν απαιτείται ο

καθορισμός μεταβλητής πεδίου η οποία θα αντιπροσωπεύει τον καθηγητή που θα διδάσκει την εκάστοτε διάλεξη.

3.4 Περιορισμοί

Οι περιορισμοί του συστήματος διακρίνονται σε αυστηρούς και εύκαμπτους όπως έχουμε αναφέρει στην παράγραφο 2.6.4.

3.4.1 Αυστηροί Περιορισμοί

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

1. 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 το 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

   $$S \neq (j*H+1) - i \quad \text{για } \forall i,j \text{ όπου } 1 \leq j \leq D,\ 1 \leq i < d.$$

   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 στο σχήμα 3.1 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**

Σχήμα 3.1 : Παράδειγμα περιορισμού 1

2. Δύο διαλέξεις του ίδιου μαθήματος δεν μπορούν να διδάσκονται την ίδια ημέρα.

Έστω δύο διαλέξεις *l* και *l΄* οι οποίες ανήκουν στο ίδιο μάθημα. Οι διαλέξεις *l* και *l΄* έχουν χρονικές στιγμές έναρξης *S* και *S΄*αντίστοιχα. Αν τώρα για κάθε ημέρα οι ώρες διδασκαλίας είναι *H* τότε θα πρέπει να ισχύει ότι

$$course(l) = course(l') \Rightarrow (S-1)/H \neq (S'-1)/H$$

Για παράδειγμα, ας θεωρήσουμε πάλι ότι το πρόβλημα μας αποτελείται από 3 ημέρες και 5 ώρες διδασκαλίας για κάθε ημέρα. Σε αυτή την περίπτωση για τις διαλέξεις *l* και *l΄* θα ισχύει ότι η μεταβλητές *S* και *S΄*και θα έχουν ως πεδίο τιμών το σύνολο {1..15},έκαστος. Αν τώρα αναθέσουμε στην μεταβλητή *S* την τιμή 7 τότε θα ισχύει ότι *(S-1) /H* = (7-1)/5 = 1 (Θυμίζουμε ότι οι πράξεις αφορούν ακέραιες και όχι πραγματικές τιμές). Για να ικανοποιείται τώρα η παραπάνω σχέση θα πρέπει είτε *(S΄-1)*/5<1 ή *(S΄-1)*/5 >1. Οι τιμές 6,7,8,9 και 10 της *S΄*δεν ικανοποιούν την σχέση και επομένως το νέο πεδίο τιμών της *S΄* θα είναι το {1..5,11..15}, το οποίο απεικονίζεται και στο σχήμα 3.2 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

Σχήμα 3.2 : Παράδειγμα περιορισμού 2

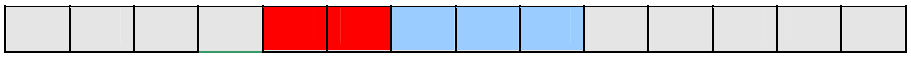
3. Ένας καθηγητής δεν μπορεί να παραδίδει συγχρόνως δύο διαλέξεις.

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 δεν 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

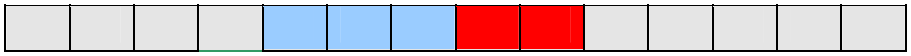
$$teacher(l) = teacher(l') \Rightarrow (S + d \leq S') \vee (S' + d' \leq S)$$

Για παράδειγμα, ας θεωρήσουμε ότι διαλέξεις *l* και *l΄* διδάσκονται από τον ίδιο καθηγητή και επιπλέον έχουν διάρκεια *2* και *3* ωρών αντίστοιχα. Όπως αναφέραμε και παραπάνω, θα πρέπει είτε η *l* να διδάσκεται *2* ώρες νωρίτερα της *l΄* όπως

φαίνεται στο σχήμα 3.3(α), είτε η *l΄* να διδάσκεται *3* ώρες νωρίτερα της *l* όπως φαίνεται στο σχήμα 3.3(β).



Σχήμα 3.3(α)



Σχήμα 3.3(β) : Παράδειγμα περιορισμού 3

4. Δύο διαλέξεις που ανήκουν σε διαφορετικά μαθήματα και τα μαθήματα τους ανήκουν στην ίδια ομάδα μαθημάτων, δεν 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

$$(course(l) \neq course(l')) \wedge (course(l) \in g) \wedge (course(l') \in g) \Rightarrow (S+d \leq S') \vee (S'+d' \leq S)$$

   Αντίστοιχο παράδειγμα είναι αυτό του σχήματος 3.3.

5. Δύο διαλέξεις δεν 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 δεν 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

$$room(l) = room(l') \Rightarrow (S+d \leq S') \vee (S'+d' \leq S)$$

   Αντίστοιχο παράδειγμα είναι αυτό του σχήματος 3.3.

6. Μία διάλεξη δεν 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

$$(\ teacher(l) = t\ ) \wedge notavailiable(t,p) \wedge ((p-i-1)\ /H = (\ p-1)/H) \Rightarrow S+i \neq p$$
για $\forall\ p,i$ όπου $p \in P,\ 0 \leq i < d$

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 στο σχήμα 3.4 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

Σχήμα 3.4 : Παράδειγμα περιορισμού 6

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 ο εφόσον η διάλεξη έχει διάρκεια *3* ώρες δεν μπορεί να ξεκινήσει τις χρονικές στιγμές *2* και *3* γιατί θα απαιτούντο να ήταν ο καθηγητής στο μάθημα την χρονική στιγμή *4*. Με την ίδια λογική θα έπρεπε να διαγραφούν λόγω την μη διαθεσιμότητας του την χρονική στιγμή *11* οι τιμές *10* και *9*, επειδή όμως ανήκουν σε διαφορετική ημέρα από την *11* δεν διαγράφονται.

7. Δεν 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

$$(room(l) = r) \wedge notavailiable(r,p) \wedge ((p - i - 1)/H = (p - 1)/H) \Rightarrow S + i \neq p$$
για $\forall\ p,i$ όπου $p \in P,\ 0 \leq i < d$

Παρόμοιο παράδειγμα μπορούμε να θεωρήσουμε αυτό του σχήματος 3.4.

8. Μία διάλεξη μπορεί να διδάσκεται σε συγκεκριμένες αίθουσες.

9. Μία διάλεξη, η οποία είναι εργαστηριακή, δεν μπορεί να διδάσκεται σε μη εργαστηριακή αίθουσα.

10. Μία διάλεξη, η οποία δεν είναι εργαστηριακή, δεν μπορεί να διδάσκεται σε εργαστηριακή αίθουσα.

11. Η έναρξη της διδασκαλίας μίας διάλεξη πρέπει να γίνεται σε συγκεκριμένες χρονικές στιγμές.

3.4.2 Εύκαμπτοι Περιορισμοί

Οι εύκαμπτοι περιορισμοί που μπορεί και να μην ικανοποιούνται απαραίτητα από το σύστημα είναι οι εξής:

1. Μία διάλεξη με μεγάλο, μεσαίο ή μικρό αριθμό φοιτητών πρέπει να διδάσκεται σε αίθουσα αντίστοιχης χωρητικότητας.

   Έστω μία διάλεξη l, η οποία ανήκει στο μάθημα c. Τότε για κάθε αίθουσα $r \in R$ θα ισχύει ότι

$$(course(l) = c) \wedge (students\_number(c) \neq capacity(r)) \Rightarrow room(l) \neq r$$

3.5 Λύσεις και Ποιοτικά κριτήρια

Λύση του προβλήματος αποτελεί οποιαδήποτε πλήρης ανάθεση τιμών στις μεταβλητές του CSP προβλήματος που έχουμε ορίσει, η οποία ικανοποιεί όλους τους αυστηρούς περιορισμούς. Σκοπός μας δεν είναι να βρούμε απαραίτητα τη βέλτιστη λύση, γιατί κάτι τέτοιο μπορεί να αποβεί χρονοβόρο, αλλά μία αρκετά καλή λύση η οποία να ικανοποιεί τις απαιτήσεις του χρήστη όπως αυτές έχουν ορισθεί.

Η ποιότητα ενός ωρολογίου προγράμματος καθορίζεται βάση τριών κριτηρίων τα οποία συνδυάζονται γραμμικά σε σχέση με κάποια καθορισμένα βάρη, παράγοντας την αντικειμενική συνάρτηση (objective function), η οποία πρέπει να έχει όσο το δυνατόν μικρότερη τιμή. Τα κριτήρια 1 και 2 αναφέρονται στο [1], ενώ το κριτήριο 3 αποτελεί τροποποιημένη ιδέα η οποία προέκυψε από τα [4, 6, 12, 14, 18].

1) Ομοιόμορφη κατανομή των διδακτικών ωρών για κάθε ομάδα μαθημάτων κατά την διάρκεια της εβδομάδας. Κάτι τέτοιο εκφράζεται ως το συνολικό άθροισμα για

κάθε ομάδα μαθημάτων της διαφοράς μεταξύ του μέγιστου και ελάχιστου αριθμού διδακτικών ωρών κάθε ημέρας κατά την διάρκεια της εβδομάδας:

$$\Sigma_{g \in G}(max_{1 \le day \le D} \{\Sigma_{l \in L}\ d : course(l) \in g, ((S-1)/H+1) = day\} - min_{1 \le day \le D} \{ \Sigma_{l \in L}\ d : course(l) \in g, ((S-1)/H+1) = day \})$$

2) Ελαχιστοποίηση του αριθμού των κενών που υπάρχουν μεταξύ των μαθημάτων της ίδιας ομάδας, κατά την διάρκεια της κάθε ημέρας. Κάτι τέτοιο εκφράζεται ως το συνολικό άθροισμα για κάθε ομάδα μαθημάτων, της διαφοράς μεταξύ του αθροίσματος για κάθε ημέρα της διαφοράς της μέγιστης λήξης μείον της ελάχιστης έναρξης των διαλέξεων που ανήκουν στην ίδια ομάδα, και του αθροίσματος των συνολικών διδακτικών όρων τους :

$$\Sigma_{g \in G} \Sigma_{1 \le day \le D} max(\{S + d : ((S-1)/H+1) = day, l \in L, course(l) \in g\} \cup \{0\}) - min(\{S : ((S-1)/H+1) = day, l \in L, course(l) \in g\} \cup \{0\}) - \Sigma_{l \in L} \{ d : course(l) \in g\}$$

3) Την κατανομή των αιθουσών ανάλογα με το μέγεθός τους στις διαλέξεις σε σχέση με τον αριθμό των φοιτητών κάθε διάλεξης.

Θεωρώ την συνάρτηση $weight(x)$ $\begin{cases} 1, & x = small \\ 5, & x = medium \\ 12, & x = big \end{cases}$ η οποία απεικονίζει κάθε μέγεθος $x \in \{big, medium, small\}$ σε μία τιμή βάρους *1,5* και *12* αντίστοιχα. Παρατηρούμε ότι όσο μεγαλύτερο είναι το μέγεθος τόσο μεγαλύτερο είναι και το βάρος που του αντιστοιχεί. Επομένως η κατανομή των αιθουσών εκφράζεται ως το συνολικό άθροισμα για κάθε διάλεξη $l \in L$ της ελάχιστης τιμής, της απόλυτης διαφοράς του βάρους $weight(x_1)$ του αριθμού των φοιτητών x_1 που παρακολουθούν την διάλεξη l μείον του βάρους $weight(x_2)$ της χωρητικότητας x_2 της αίθουσας $r \in R$ στην οποία διδάσκεται.

$$\Sigma_{l \in L}\ min_{r \in R}\{ | weight(x_1) - weight(x_2)| : x_1 = students\_number(c), c = course(l) \\ x_2 = capacity(r), r = room(l)\}$$

Από την παραπάνω σχέση αν εξετάσουμε τις απόλυτες τιμές της διαφοράς $|weight(x_1) - weight(x_2)|$ προκύπτουν οι εξής συνδυασμοί

$$| weight(x_1) - weight(x_2)| \begin{cases} 0, & x_1 = x_2 \\ 7, & x_1 = big \text{ και } x_2 = medium \\ 11, & x_1 = big \text{ και } x_2 = small \\ 7, & x_1 = medium \text{ και } x_2 = big \\ 4, & x_1 = medium \text{ και } x_2 = small \\ 11, & x_1 = small \text{ και } x_2 = big \\ 4, & x_1 = small \text{ και } x_2 = medium \end{cases}$$

Από τους παραπάνω συνδυασμούς είναι φανερό ότι εφόσον επιλέγουμε την ελάχιστη απόλυτη διαφορά, τότε θα επιλέγονται κάθε φορά για κάθε διάλεξη

αίθουσες ίσου μεγέθους με τον αριθμό φοιτητών που την παρακολουθούν. Σε περίπτωση που δεν υπάρχει αίθουσα αντίστοιχου μεγέθους τότε εάν ο αριθμός των φοιτητών είναι

- $x_1 = big$, τότε θα προτιμηθεί μία αίθουσα $x_2 = medium$, καθώς η απόλυτη διαφορά που προκύπτει $7<11$ για $x_2 = small$.
- $x_1 = medium$, τότε θα προτιμηθεί μία αίθουσα $x_2 = small$, καθώς η απόλυτη διαφορά που προκύπτει $4<7$ για $x_2 = big$.
- $x_1 = small$, τότε θα προτιμηθεί μία αίθουσα $x_2 = medium$, καθώς η απόλυτη διαφορά που προκύπτει $4<11$ για $x_2 = big$.

Εάν τώρα θέλουμε να ορίσουμε την αντικειμενική συνάρτηση του προβλήματος, τότε αν θεωρήσουμε ως *cost1* το αποτέλεσμα του πρώτου κριτηρίου, *cost2* το αποτέλεσμα του δεύτερου κριτηρίου και τέλος *cost3* το αποτέλεσμα του τρίτου κριτηρίου, τότε η αντικειμενική συνάρτηση *objective_function* θα είναι ίση με

$$objective\_function = 10 * cost1 + cost2 + cost3$$

από τον ορισμό της συνάρτησης παρατηρούμε ότι το πρώτο κριτήριο έχει μεγαλύτερο βάρος σε σχέση με τα άλλα δύο, γεγονός το οποίο σημαίνει ότι θεωρείται πιο σημαντική η ικανοποίηση του. Τα κριτήρια 2 και 3 είναι ισοβαρή, γεγονός που τα καθιστά ισότιμα.

3.6 Περίληψη

Στο κεφάλαιο αυτό παρουσιάστηκε το μοντέλο του προβλήματος, βασιζόμενο στα προβλήματα ικανοποίησης περιορισμών για πεπερασμένα πεδία καθώς επίσης και των εύκαμπτων και αυστηρών περιορισμών που διέπουν το πρόβλημά μας. Η ποιότητα του ωρολογίου προγράμματος μας καθορίστηκε βάση τριών κριτηρίων τα οποία συνδυάστηκαν γραμμικά σε σχέση με κάποια καθορισμένα βάρη, παράγοντας την αντικειμενική συνάρτηση του προβλήματος.

Κεφάλαιο 4

Υλοποίηση και Μέθοδοι αναζήτησης

4.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση του μηχανισμού κατάρτισης ωρολογίου προγράμματος. Πιο αναλυτικά, περιγράφεται η υλοποίηση των περιορισμών, του ευριστικού αλγόριθμου αναζήτησης που έχουμε αναπτύξει καθώς επίσης και του τρόπου επικοινωνίας του συστήματος της ECL^{i}PS^e με το σύστημά μας.

Σκοπός μας είναι η ανάθεση των διαλέξεων όλων των μαθημάτων στο χρόνο και στις αίθουσες έτσι ώστε η λύση που προκύπτει να ικανοποιεί όλους τους περιορισμούς του προβλήματος, καθώς επίσης να ικανοποιεί τα κριτήρια εκείνα βάση των οποίων θεωρείται αποδεκτή από τον χρήστη σε σχέση με κάποιες άλλες λύσεις που επίσης ικανοποιούν τους περιορισμούς άλλα δεν θεωρούνται τόσο "ποιοτικές". Θεωρούμε ότι η ανάθεση των καθηγητών στα μαθήματα είναι προκαθορισμένη. Σε περίπτωση που ο χρήστης δώσει στο σύστημα δεδομένα τα οποία δεν ικανοποιούν τους περιορισμούς τότε το σύστημα τερματίζει και ζητά από τον χρήστη να χαλαρώσει κάποιους από τους περιορισμούς που έχει θέσει ή να κάνει κάποια αλλαγή στα δεδομένα. Σε περίπτωση τώρα που δεν υπάρχει λύση αλλά το σύστημα αδυνατεί να το εντοπίσει εξ' αρχής, τότε ή θα πρέπει να επιδοθεί σε μία εξαντλητική αναζήτηση του χώρου αναζήτησης, γεγονός το οποίο για μεγάλο αριθμό δεδομένων είναι αδύνατων λόγω του μεγάλου μεγέθους, ή μπορούμε να θέσουμε ένα χρονικό φράγμα μέσα στο οποίο ο αλγόριθμος πρέπει να έχει επιστρέψει λύση διαφορετικά το σύστημα τερματίζει. Σε περίπτωση που βρεθεί λύση το σύστημα ενημερώνει τον χρήστη και αναμένει απάντηση ως προς το αν επιθυμεί επόμενη λύση ή το τερματισμό της αναζήτησης. Στη περίπτωση που ο χρήστης επιθυμεί επόμενη λύση το σύστημα θα αναζητήσει λύση της οποίας η συνάρτηση κόστους θα είναι ίση ή μικρότερη της προηγούμενης.

4.2 Αναπαράσταση Δεδομένων

Όλες οι πληροφορίες σχετικά με τις λεπτομέρειες των μαθημάτων, των μελών διδακτικού προσωπικού, των αιθουσών κ.λ.π., κωδικοποιούνται με έναν ιδιαίτερο τρόπο. Πιο συγκεκριμένα:

1. Κάθε καθηγητής κωδικοποιείται με το γεγονός (fact) του κατηγορήματος (predicate) *teacher/2* το οποίο αποτελείται από τα εξής χαρακτηριστικά γνωρίσματα: Το όνομα του *Name* το οποίο μπορεί να είναι και αριθμός, και μία λίστα *Notavailable* η οποία περιέχει τις χρονικές στιγμές στις οποίες ο καθηγητής **δεν** είναι διαθέσιμος. Για παράδειγμα, *teacher(1,[2,3,7])* το οποίο αναφέρεται στον καθηγητή *1* ο οποίος δεν είναι διαθέσιμος τις χρονικές στιγμές *2,3* και *7*.

2. Κάθε αίθουσα κωδικοποιείται με το γεγονός του κατηγορήματος *room/3* το οποίο αποτελείται από τα εξής χαρακτηριστικά γνωρίσματα: Το όνομα της αίθουσας *Name* το οποίο μπορεί να είναι και αριθμός, την χωρητικότητα *Capacity* η οποία μπορεί να πάρει τιμές *small, medium* και *big*, και μία λίστα *Notavailable* η οποία περιέχει τις χρονικές στιγμές στις οποίες η αίθουσα **δεν** είναι διαθέσιμη. Για παράδειγμα, *room(20,small,[5])* το οποίο αναφέρεται στην αίθουσα *20* η οποία δεν είναι διαθέσιμη την χρονική στιγμή *5* και έχει μικρή χωρητικότητα.

3. Κάθε διάλεξη κωδικοποιείται με το γεγονός του κατηγορήματος *lecture/6* το οποίο αποτελείται από τα εξής χαρακτηριστικά γνωρίσματα: Το όνομα της διάλεξης *Name* το οποίο μπορεί να είναι και αριθμός, την μεταβλητή χρονικής στιγμής έναρξης *Start*, την λίστα *Start_times* η οποία περιέχει τις χρονικές στιγμές που πρέπει να γίνεται έναρξη διδασκαλίας της διάλεξης, τον χρόνο διάρκειας *Duration* που διδάσκεται, την μεταβλητή *Room* της αίθουσας στην οποία θα γίνει η παράδοση της διάλεξης και την λίστα *Rooms* η οποία περιέχει όλα τα ονόματα των αιθουσών στα οποία μπορεί η διάλεξη να διδαχθεί, με βάση τις επιθυμίες του χρήστη και το είδος της διάλεξης (διάλεξη, εργαστήριο ή ασκήσεις). Για παράδειγμα, *lecture(5,1,[1,2,3,10],2,10,[10,11])* το οποίο περιγράφει την διάλεξη *5*, η οποία αρχίζει την χρονική στιγμή *1*, έχει πιθανές στιγμές έναρξης *1,2,3* και *10*, έχει διάρκεια *2*, διδάσκεται στην αίθουσα *10* και μπορεί να διδαχθεί στις αίθουσες *10* και *11*.

4. Κάθε μάθημα κωδικοποιείται με το γεγονός του κατηγορήματος *course/4* το οποίο αποτελείται από τα εξής χαρακτηριστικά γνωρίσματα: Το όνομα του *Name* το οποίο μπορεί να είναι και αριθμός, την λίστα *Lectures* από σύνθετους όρους (compound terms), οι οποίοι αναπαριστούν τις διαλέξεις του μαθήματος, το μέγεθος του αριθμού φοιτητών που παρακολουθούν το μάθημα *Students_number* το οποίο μπορεί να πάρει τιμές *small, medium* και *big*, και το όνομα *Teacher_name* του καθηγητή που το διδάσκει. Για παράδειγμα, *course(3,[lecture(4,_,[],2,_,[2,3,9]),lecture(5,_,[],2,_,[2,3,9])],big,3)* το οποίο περιγράφει το μάθημα *3*, το οποίο έχει δύο διαλέξεις, παρακολουθείται από μεγάλο αριθμό φοιτητών και διδάσκεται από τον καθηγητή *3*. Οι διαλέξεις *4* και *5* του μαθήματος δεν ξέρουμε ποιες χρονικές στιγμές διδάσκονται καθώς η χρονική στιγμή έναρξης διδασκαλίας τους περιγράφεται από μία ανώνυμη μεταβλητή (_), μπορούν να διδαχθούν όλες τις χρονικές στιγμές καθώς η λίστα *Start_times* είναι κενή, έχουν διάρκεια *2*, δεν γνωρίζουμε σε ποια αίθουσα διδάσκονται καθώς η αίθουσα διδασκαλίας τους περιγράφεται από μία ανώνυμη μεταβλητή (_) και μπορούν να διδαχθούν στις αίθουσες *2,3* και *9*.

5. Κάθε ομάδα μαθημάτων κωδικοποιείται με το γεγονός του κατηγορήματος *group/2* το οποίο αποτελείται από τα εξής χαρακτηριστικά γνωρίσματα: Το όνομα της ομάδας *Name* και μία λίστα *Course_names* με τα ονόματα των μαθημάτων που της ανήκουν. Για παράδειγμα, *group(1,[1,2,3,5,6,7,9,10])* το οποίο περιγράφει την ομάδα *1* των μαθημάτων *1,2,3,5,6,7,9* και *10*.

Μία λύση του ωρολογίου προγράμματος αναπαρίσταται από μία λίστα σύνθετων όρων, οι οποίοι αναπαριστούν τα μαθήματα με τις διαλέξεις τους, χρησιμοποιώντας την

ορολογία της ECL^{i}PS^e. Σε κάθε σύνθετο όρο που αναπαριστά μία διάλεξη θα πρέπει να έχει γίνει ανάθεση τιμών στις μεταβλητές χρόνου και αίθουσας.

4.3 Υλοποίηση Περιορισμών

Για να υπάρξει λύση θα πρέπει να ικανοποιούνται όλοι οι αυστηροί περιορισμοί του μοντέλου μας. Κάποιοι περιορισμοί μπορούσαν να υλοποιηθούν με περισσότερους από έναν τρόπους. Η επιλογή τους πως τελικά θα υλοποιούταν ένας περιορισμός, είχε να κάνει καθαρά και μόνο με τον εσωτερικό μηχανισμό της ECL^{i}PS^e και την αλληλεπίδρασή του με κάθε μία από τις περιπτώσεις υλοποίησης ξεχωριστά. Τα κριτήρια μας, δεδομένου του ότι κάναμε κύρια χρήση της βιβλιοθήκης fd, ήταν με βάση του πια υλοποίηση παράγει το μικρότερο δέντρο παραγωγής (derivation tree), όπως αναφέρεται στα [27, 28], για τον εσωτερικό μηχανισμό της ECL^{i}PS^e και σε αλληλεπίδραση με τον αλγόριθμο της *συνέπειας ορίων,* τον οποίο χρησιμοποιεί η fd, πως μπορεί να περιορίσει το χώρο αναζήτησης πιο γρήγορα και αποτελεσματικά, όπως αναφέρεται στο [28].

- **Ο περιορισμός 1** διασφαλίζει ότι κάθε διάλεξη με διάρκεια Duration, θα διδάσκεται μέσα στα όρια της ημέρας και όχι έξω από αυτά, όπως αναφέρεται στα [6, 7, 8]. Εκφράζεται από τον κανόνα *setconstraint1(S,Duration,Hours,Days)* ο οποίος παίρνει σαν όρισμα την χρονική μεταβλητή έναρξης *S*, τη διάρκεια *Duration*, τον αριθμό των ωρών διδασκαλίας *Hours* και τον αριθμό των ημερών διδασκαλίας *Days*. Για κάθε διάλεξη όλων των μαθημάτων περιορίζουμε τις τιμές της μεταβλητής *S* αφαιρώντας όλες τις τιμές για τις οποίες το άθροισμα *S+Duration* ξεπερνά τα όρια κάθε ημέρας, μέσω της σχέσης *S ## I-J+1* με *I∈{Hours+1,2*Hours+1,..,Days*Hours+1}* και *J∈{2..Duration},* όπου η μεταβλητή *S* τίθεται διάφορη του αθροίσματος *I-J+1* για κάθε τιμή των *I* και *J*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets constraint '1' which is that no start time variable S  %
% can be evaluated such as the sum S + Duration is not     %
% among the limits of a day period.                        %
% Day limits = {Hours+1,2*Hours+1,..,Days*Hours+1}.  %
%setconstraint1(S,Duration,Hours,Days)                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setconstraint1(S,Duration,Hours,Days):-
   I is Days*Hours+1,
   J is Duration,
   foreachi(S,J,Hours,I).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%eliminates values from domain(S)                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
foreachi(_,_,_,1):-!.
foreachi(S,J,Hours,I):-      %for(int i=Days*Hours+1;i>=Hours+1;i=i-Hours)
   I#>=Hours+1,
   foreachj(S,I,J),
   NI is I-Hours,
   foreachi(S,J,Hours,NI).
```

```
foreachj(_,_,1):-!.
foreachj(S,I,J):-            %for(int j=Duration;j>=2;j--)
   J#>=2,
   S##I-J+1,                 % S##i-j+1
   NJ is J-1,
   foreachj(S,I,NJ).
```

- **Ο περιορισμός 2** διασφαλίζει ότι δύο διαλέξεις του ίδιου μαθήματος δεν μπορούν να διδάσκονται την ίδια ημέρα, όπως αναφέρεται στα [1,2,3,4,5,6,7,8,9,14,15,16, 18]. Εκφράζεται από τον κανόνα *setconstraint2(List_Courses,Hours,Days)* ο οποίος παίρνει σαν ορίσματα την λίστα *List_Courses* η οποία περιέχει όλα τα μαθήματα, τον αριθμό των ωρών διδασκαλίας *Hours* και τον αριθμό των ημερών διδασκαλίας *Days*. Ουσιαστικά για κάθε μάθημα δημιουργείται μία λίστα *DayList* η οποία περιέχει τις μεταβλητές με τις ημέρες που θα διδακτή κάθε διάλεξη του μαθήματος. Αν τώρα *D* είναι η χρονική μεταβλητή της ημέρας και *S* η χρονική μεταβλητή της έναρξης, τότε αυτές οι δύο μεταβλητές συνδέονται μέσω του περιορισμού *element(S,L,D)* όπου *L* η λίστα που αντιστοιχεί κάθε χρονική στιγμή έναρξης στην αντίστοιχη ημέρα. Θέτοντας τώρα τον περιορισμό *alldifferent(DayList)*, όπως αναφέρεται στα [1, 2, 6], διασφαλίζεται ότι κάθε ημέρα *D* της λίστας θα έχει μοναδική τιμή και επομένως θα διαγραφούν αυτομάτως από όλες τις μεταβλητές *S* των υπόλοιπων διαλέξεων του μαθήματος οι τιμές εκείνες που αντιστοιχούν στην ίδια ημέρα *D* κάθε φορά που θα γίνεται ανάθεση σε μία μεταβλητή *S*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets constraint '2' which is that lectures about the same course %
% must not be scheduled in the same day                          %
% setconstraint2(List_Courses,Hours,Days)                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setconstraint2(List_Courses,Hours,Days):-
   foreachday(Hours,Days,L),        % construct list of days ex.[1,1,1,2,2,2,3,3,3]
   makeDayslist(List_Courses,L).  % construct variables of Days

foreachday(_,0,[]):-!.
foreachday(Hour,Day,L):-          % for(int i=Days;i>=1;i--)
   Day#>=1,
   foreachHour(Hour,Day,L1),
   NDay is Day-1,
   foreachday(Hour,NDay,L2),
   append(L2,L1,L).

foreachHour(0,_,[]):-!.            %for(int j=Hours;j>=1;j--)
foreachHour(Hour,Day,[Day|L]):- % construct list of days
%ex.[1,1,1,2,2,2,3,3,3]
   Hour#>=1,
   NHour is Hour-1,
   foreachHour(NHour,Day,L).
```

```
makeDayslist([],_):-!.
makeDayslist([course(_,Ls,_,_)|Ts],L):-
  makeDay(Ls,L,DayList),
 fd : alldifferent(DayList), %Sets that all day D variables should be different
  makeDayslist(Ts,L).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% creates day D variables for each variable S      %
% makeDay([List_lectures,List_of_days,DayList)%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
makeDay([],_,[]):-!.
makeDay([lecture(_,S,_,_,_,_)|Ls],L,[D|DayList]):-
  element(S,L,D),% creates D (Day) variable for each variable S
  makeDay(Ls,L,DayList).
```

Ένας άλλος τρόπος ο οποίος χρησιμοποιείται στα [7, 8, 14] είναι ο εξής, έστω δύο διαλέξεις με χρονικές μεταβλητές έναρξης S_1 και S_2 αντίστοιχα και X μία μεταβλητή η οποία έχει ως πεδίο τιμών τις χρονικές στιγμές που αντιστοιχούν στα όρια της κάθε ημέρας. Αν τώρα το πεδίο τιμών των S_1 και S_2 ήταν *{1..240}* και η X είχε πεδίο τιμών *{48,96,144,19}* τότε η σχέση $S_1 < X \wedge X < S_2$ θα διασφάλιζε τον περιορισμό 2. Μία τέτοια έκφραση θα αύξανε το δέντρο παραγωγής του συστήματος, καθώς θα εισήγαγε περισσότερα σημεία επιλογής στο δέντρο, με αποτέλεσμα να έχουμε μεγαλύτερο χρονικό κόστος σε σχέση με την προηγούμενη μέθοδο λόγω του μεγάλου αριθμού άσκοπων οπισθοδρομήσεων. Επιπλέον, αναφέρεται στο [8] ότι η παραπάνω μέθοδος μπορεί να χρησιμοποιηθεί μόνο στην περίπτωση που οι διαλέξεις έχουν ίδια χαρακτηριστικά, γεγονός το οποίο είναι περιοριστικό καθώς δεν καλύπτει όλες τις περιπτώσεις.

- **Ο περιορισμός 3** διασφαλίζει ότι ένας καθηγητής δεν μπορεί να παραδίδει συγχρόνως δύο διαλέξεις, όπως αναφέρεται στα [1, 2, 3, 4, 5, 6, 14, 15, 16, 18]. Εκφράζεται από τον κανόνα *setconstraint3(Teachers, Hours,List_Courses)* ο οποίος παίρνει σαν ορίσματα την λίστα *List_Courses* η οποία περιέχει όλα τα μαθήματα, τον αριθμό των ωρών διδασκαλίας *Hours* και την λίστα *Teachers* η οποία περιέχει όλους τους καθηγητές. Για κάθε καθηγητή, εξασφαλίζει για όλες τις διαλέξεις των μαθημάτων που διδάσκει μέσω του περιορισμού *cumulative(Startlist,Durationlist, Resourceslist,1)*, ότι δεν μπορούν να συμπίπτουν χρονικά, όπου *Startlist* η λίστα με τις χρονικές μεταβλητές έναρξης *S* κάθε διάλεξης, *Durationlist* η λίστα με την διάρκεια κάθε διάλεξης και *Resourceslist* η λίστα που περιέχει τα ποσά των απαιτήσεων κάθε διάλεξης σε πόρους, η οποία στην προκειμένη περίπτωση είναι της μορφής π.χ. *[1,1,1,1]* όπως διασφαλίζεται από τον κανόνα *selectlectures/4* και αυτό γιατί αν θεωρήσουμε ότι οι καθηγητές αποτελούν πόρους του συστήματος τότε κάθε διάλεξη απαιτεί έναν καθηγητή. Τέλος το τέταρτο όρισμα του *cumulative/4* έχει την τιμή *1*, εξασφαλίζοντας έτσι ότι οι διαλέξεις δεν μπορούν να συμπίπτουν χρονικά.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets constraint '3' which is that any lectures have a common teacher %
% can not overlap in time.                                              %
% Also sets soft constraint 1 which is that any lecture can be taught   %
% only when its teacher is available.                                   %
%setconstraint3(Teachers, Hours, List_Courses)                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setconstraint3([],_,_):- !.
setconstraint3([teacher(T,Notavailable)|Ts],Hours,List_Courses):-
   selectcourses(T,List_Courses,Startlist,Durationlist,Resourceslist),
   setsoftconstraint1(Startlist,Durationlist,Hours,Notavailable),       % sets soft constraint 1
   cumulative(Startlist,Durationlist,Resourceslist,1),                  % sets constraint "3"
   setconstraint3(Ts,Hours,List_Courses).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%  selectcourses(Teacher,List_Courses,Startlist,Durationlist,Resourceslist)  %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
selectcourses(_,[],[],[],[]):- !.
selectcourses(T,[course(_,Ls,_,T)|Ts],Startlist,Durationlist,Resourceslist):-
   selectlectures(Ls,Startlist1,Durationlist1,Resourceslist1),
   selectcourses(T,Ts,Startlist2,Durationlist2,Resourceslist2),
   append(Startlist1,Startlist2,Startlist),
   append(Durationlist1,Durationlist2,Durationlist),
   append(Resourceslist1,Resourceslist2,Resourceslist),!.
selectcourses(T,[course(_,_,_,TN)|Ts],Startlist,Durationlist,Resourceslist):-
   T##TN,
   selectcourses(T,Ts,Startlist,Durationlist,Resourceslist).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%  selectlectures(List_lectures,Startlist,Durationlist,Resourceslist) %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
selectlectures([],[],[],[]):- !.
selectlectures([lecture(_,S,_,D,_,_)|Ls],[S|Startlist],[D|Durationlist],[1|Resourceslist]):-
   selectlectures(Ls,Startlist,Durationlist,Resourceslist).
```

Όπως αναφέρεται και στο [12], περιορισμοί αυτού του τύπου (όπου δεν επιτρέπεται να υπάρχει χρονική επικάλυψη των διεργασιών) μπορούν να εκφραστούν και μέσω διαζεύξεων. Ας θεωρήσουμε δύο διεργασίες με χρονικές μεταβλητές έναρξης *Si* και *Sj*, και διάρκεια *Di* και *Dj* αντίστοιχα.
Σύμφωνα με το [12] μία πρώτη μέθοδο είναι με την εισαγωγή σημείων επιλογής:

```
disjuctive(Si,Di,Sj,Dj) :-
   Si #>= Sj + Dj.
disjuctive(Si,Di,Sj,Dj) :-
   Sj #>= Si + Di.
```

Αυτή η μέθοδος δεν είναι και τόσο αποτελεσματική γιατί η διάζευξη δεν μπορεί να θεωρηθεί ως περιορισμός με τον τρόπο που έχει εκφραστεί. Επίσης εισάγονται πολλά σημεία επιλογής με αποτέλεσμα να οδηγούμαστε σε άσκοπες οπισθοδρομήσεις. Η δεύτερη μέθοδος που πρότεινε το [12] είναι η χρήση δαιμόνων (demons), η οποία υλοποιεί συνθήκες:

```
disjuctive(Si,Di,Sj,Dj) :-
  if  Si + Di > Sj then Sj + Dj <= Si,
  if  Sj + Dj > Si then Si + Di <= Sj.
```

Αυτή η μέθοδος είναι πιο αποτελεσματική από την προηγούμενη, παρουσιάζει όμως αδυναμία στην αφύπνιση των περιορισμών και στον περιορισμό του δέντρου αναζήτησης.

- **Ο περιορισμός 4** διασφαλίζει ότι δύο διαλέξεις που ανήκουν σε μαθήματα της ίδιας ομάδα μαθημάτων, δεν μπορούν να διδάσκονται συγχρόνως, όπως αναφέρεται στα [1, 7, 8, 9, 14]. Εκφράζεται από τον κανόνα *setconstraint4(List_Groups, List_Courses)* ο οποίος παίρνει σαν ορίσματα την λίστα *List_Courses*, η οποία περιέχει όλα τα μαθήματα, και την λίστα *List_Groups* η οποία περιέχει όλες τις ομάδες μαθημάτων. Για κάθε ομάδα μαθημάτων, εξασφαλίζει ότι όλες τις διαλέξεις των μαθημάτων που της ανήκουν, μέσω του περιορισμού *cumulative(Startlist, Durationlist,Resourceslist,1)*, ότι δεν μπορούν να συμπίπτουν χρονικά με ανάλογο τρόπο με αυτό του περιορισμού 3.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets constraint '4' which is that courses about the same group  %
% can not overlap in time.                                         %
% setconstraint4(List_Groups,List_Courses)                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setconstraint4([],_):-!.
setconstraint4([group(_,Ls)|Lg],List_Courses):-
  foreachcourse(Ls,List_Courses,Startlist,Durationlist,Resourceslist),
  cumulative(Startlist,Durationlist,Resourceslist,1),          %sets constraint 4
  setconstraint4(Lg,List_Courses).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%foreachcourse(List_of_Course_names,List_Courses,Startlist,Durationlist,Resourceslist)   %
%                                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
foreachcourse([],_,[],[],[]):-!.
foreachcourse([N|Ns],List_Courses,Startlist,Durationlist,Resourceslist):-
  getcourse(course(N,Ls,_,_),List_Courses),
  selectlectures(Ls,Startlist1,Durationlist1,Resourceslist1),
  foreachcourse(Ns,List_Courses,Startlist2,Durationlist2,Resourceslist2),
  append(Startlist1,Startlist2,Startlist),
  append(Durationlist1,Durationlist2,Durationlist),
  append(Resourceslist1,Resourceslist2,Resourceslist).


getcourse(course(N,Ls,Cap,T),List_Courses):-
```

```
once(member(course(N,Ls,Cap,T),List_Courses)).
```

- **Ο περιορισμός 5** διασφαλίζει ότι δύο διαλέξεις δεν μπορούν να διδάσκονται στην ίδια αίθουσα συγχρόνως, όπως αναφέρεται στα [1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18] και **ο περιορισμός 7** διασφαλίζει ότι δεν μπορεί να γίνει ανάθεση μίας αίθουσας σε ένα μάθημα τι χρονικές στιγμές που αυτή δεν είναι διαθέσιμη, όπως αναφέρεται στα [1, 2]. Οι περιορισμοί αυτοί εκφράζονται από τον κανόνα *setconstraint6(Rooms,Tasklist)* ο οποίος παίρνει σαν ορίσματα την λίστα *Rooms* η οποία περιέχει όλες τι αίθουσες και την λίστα *Tasklist* η οποία περιέχει όλες τις διεργασίες (διαλέξεις*) task(C,Ns,Cap,S,D,VR)* όπου *C* το μάθημα στο οποίο ανήκει η διάλεξη *Ns, Cap* ο αριθμός φοιτητών που παρακολουθούν το μάθημα, *S* η χρονική μεταβλητή έναρξης , *D* η διάρκεια της διάλεξης και *VR* η μεταβλητή της αίθουσας. Έχουμε δημιουργήσει για τον **περιορισμό 5**, εσωτερικές μεταβλητές *X#=N*(S+I-2)+VR* (παρόμοιες με αυτές που αναφέρονται στα [2, 7]) όπου *N* ο αριθμός των αιθουσών |*R*|, *S* η χρονική μεταβλητή έναρξης της αντίστοιχης διάλεξης, *I* παίρνει τιμές από *1* έως και τη διάρκεια *D* της διάλεξης και τέλος *VR* η μεταβλητή της αίθουσας. Η παραπάνω σχέση αντιστοιχεί κάθε συνδυασμό των μεταβλητών *S* και *VR* δοθέντος του *I* σε μία τιμή για τη μεταβλητή *X*. Σε κάθε διάλεξη θα αντιστοιχεί *D* αριθμός μεταβλητών *X* όσο δηλαδή και η διάρκεια της. Για παράδειγμα αν υπάρχουν *2* αίθουσες οι *(1,2)* και η διάλεξη έχει διάρκεια *2* τότε θα έχει μεταβλητές $X_1= 2*(S-1)+VR$ και $X_2= 2*S+VR$. Αν τώρα η μεταβλητή *S* παίρνει τιμές *(1,2)* τότε οι συνδυασμοί που προκύπτουν είναι ($X_1=1$ και $X_2=3$), ($X_1=2$ και $X_2=5$), ($X_1=3$ και $X_2=5$), ($X_1=4$ και $X_2=6$). Αν τώρα υπήρχε και δεύτερη διάλεξη με ακριβώς τα ίδια χαρακτηριστικά με την προηγούμενη τότε αν η πρώτη διάλεξη ξεκινούσε την χρονική στιγμή *1* και γινόταν στην αίθουσα *2* θα είχε τιμές $X_1=2$ και $X_2=5$ αναγκαστικά η δεύτερη θα απέκλειε τους συνδυασμούς *2* και *3*. Αυτό επιτυγχάνεται θέτοντας για όλες τις μεταβλητές *X* των περιορισμό *alldifferent/1*. Με παρόμοιο τρόπο για τον **περιορισμό 7** δημιουργούμε σταθερές αυτή την φορά *X is N*(S-1)+Max* όπου *N* ο αριθμός των αιθουσών |*R*|, *S* η χρονική στιγμή στην οποία η αίθουσα *Max* δεν είναι διαθέσιμη. Ουσιαστικά μετασχηματίζουμε τις χρονικές στιγμές που κάθε αίθουσα δεν είναι διαθέσιμη στις σταθερές *X*. Αν τώρα θέσουμε τον περιορισμό *alldifferent/1* σε όλες τις μεταβλητές *X* των **περιορισμών 5 και 7** τότε διασφαλίζονται και οι δύο περιορισμοί συγχρόνως.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets constraint '6' which is that if any lectures have the same room   %
% can not overlap in time. Also sets soft constraint2                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setconstraint6(Rooms,Tasklist):-
   length(Rooms,N),
   foreachtask(Tasklist,N,VariablesXlist1),
   setsoftconstraint2(Rooms,1,N,NotavailX),              %setsoftconstraint2
   append(VariablesXlist1,NotavailX,VariablesXlist),
   fd : alldifferent(VariablesXlist).  %setsoftconstraint2 and setconstraint6 together

foreachtask([],_,[]):- !.
foreachtask([task(_,_,_,S,D,VR)|Ls],N,VariablesXlist):-
```

```
  createvariablesX(S,D,VR,N,LX1),
  foreachtask(Ls,N,LX2),
  append(LX1,LX2,VariablesXlist).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% creates X variables where X=N*(S+I-2)+VR and   %
% S start time variable                           %
% I belongs to [1,...,Duration of S]              %
% VR room variable                                %
% N the number of rooms |R|                       %
% VariablesXlist the list of variables X          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
createvariablesX(_,0,_,_,[]):-!.
createvariablesX(S,I,VR,N,[X|VariablesXlist]):-
  I#>=1,
  X#=N*(S+I-2)+VR,
  NI is I-1,
  createvariablesX(S,NI,VR,N,VariablesXlist).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets soft constraint '2' which is that any lecture can be taught  %
% only when its room is available.                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setsoftconstraint2([],Max,N,[]):- Max is N+1,!.
% for(int Max=1;Max<=N{|R|};Max++)
setsoftconstraint2([room(_,_,Notavailable)|Rs],Max,N,NotavailX):-
  Max#<=N,
  setnotavailableroom(Max,N,Notavailable,NotavailX1),
  NMax is Max+1,
  setsoftconstraint2(Rs,NMax,N,NotavailX2),
  append(NotavailX1,NotavailX2,NotavailX).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%creates X variables similar of those of constraint 6                        %
%                                                                            %
%setnotavailable(Room,Number_of_rooms,List_Notavailable,List_NotavailX) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setnotavailableroom(_,_,[],[]):-!.
setnotavailableroom(Max,N,[S|Notavailable],[X|NotavailX]):-
  X is N*(S-1)+Max,
  setnotavailableroom(Max,N,Notavailable,NotavailX).
```

- **Ο περιορισμός 6** διασφαλίζει ότι μία διάλεξη δεν μπορεί να διδαχθεί τις χρονικές στιγμές που ο καθηγητής δεν είναι διαθέσιμος, όπως αναφέρεται στα [1, 2, 3, 4, 5, 6, 16]. Εκφράζεται από τον κανόνα *setsoftconstraint1(Startlist,Durationlist,Hours, Notavailablelist)* ο οποίος παίρνει σαν ορίσματα την λίστα *Startlist* η οποία περιέχει όλες τις χρονικές μεταβλητές έναρξης *S* των διαλέξεων που διδάσκονται από τον ίδιο καθηγητή, την λίστα *Durationlist* η οποία περιέχει τις διάρκειες των διαλέξεων, την σταθερά *Hours* που αντιστοιχεί στις ώρες διδασκαλίας για κάθε ημέρα και την λίστα *Notavailablelist* που περιέχει τις χρονικές στιγμές που ο καθηγητής δεν είναι

διαθέσιμος. Για κάθε διάλεξη εξασφαλίζεται **ο περιορισμός 6** μέσω του περιορισμού *S+J##Sn* , όπου *S* η χρονική μεταβλητή έναρξης, η μεταβλητή *J* που παίρνει τιμές *[0,D-1]* όπου *D* η διάρκεια της διάλεξης και *Sn* η χρονική στιγμή που ο καθηγητής δεν είναι διαθέσιμος.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sets soft constraint '1' which is that any lecture can be taught          %
% only when its teacher is available.                                       %
%setsoftconstraint1(Startlist, Durationlist,Hours,Notavailablelist)       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setsoftconstraint1([],[],_,_):- !.
setsoftconstraint1([S|Ls],[D|Ld],Hours,Notavailable):-
   setnotavailableprof(S,D,Hours,Notavailable),
   setsoftconstraint1(Ls,Ld,Hours,Notavailable).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%eliminates values from domain(S)                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setnotavailableprof(_,_,_,[]):-!.
setnotavailableprof(S,D,Hours,[Sn|Notavailable]):-
   J is D-1,
   foreachstate(S,Hours,Sn,J),                    %sets soft constraint 1
   setnotavailableprof(S,D,Hours,Notavailable).


foreachstate(_,_,_,-1):-!.
foreachstate(S,Hours,Sn,J):-                 %for(int j=Duration-1;j>=0;j--)
   J#>=0,
   Result1 is (Sn-J-1)//Hours,
   Result2 is (Sn-1)//Hours,
   if_in_the_same_day(S,Sn,J,Result1,Result2),
   NJ is J-1,
   foreachstate(S,Hours,Sn,NJ).

if_in_the_same_day(_,_,_,R1,R2):- R1 ## R2,!.
if_in_the_same_day(S,Sn,J,R1,R2):-
   R1 #= R2,
   S+J##Sn.                    %sets soft constraint 1
```

- **Οι περιορισμοί 8,9 και 10** διασφαλίζονται μέσω των δεδομένων εισαγωγής κατά την εκτέλεση του προγράμματος όπου εξασφαλίζουμε ότι για κάθε γεγονός *lecture(Name,Start,Start_times,Duration,Room,Rooms)* τα δεδομένα της λίστας *Rooms*, που περιέχει τις αίθουσες στις οποίες μπορεί να διδάσκεται η αίθουσα, θα διασφαλίζουν τους περιορισμούς. Επιλέξαμε οι περιορισμοί αυτοί να διασφαλίζονται όχι από τον ίδιο τον αλγόριθμο αλλά από την διεπαφή (interface) με τον χρήστη γιατί αποτελούν επιλογές του χρήστη.

- **Ο περιορισμός 11** διασφαλίζει ότι η έναρξη της διδασκαλίας κάθε διάλεξης πρέπει να γίνεται σε συγκεκριμένες χρονικές στιγμές. Σε περίπτωση που ο χρήστης δεν

έχει επιλέξει χρονικές στιγμές για μία διάλεξη, τότε η διάλεξη μπορεί να διδαχθεί οποιαδήποτε χρονική στιγμή μέσα στα πλαίσια του ωρολογίου προγράμματος. Ο περιορισμός αυτός εκφράζεται μέσω του κανόνα *set_Start_Time_Variable(S,LVS, Units)* όπου *S* η χρονική μεταβλητή έναρξης, *LVS* η λίστα με τις χρονικές στιγμές έναρξης και *Units* ο αριθμός των χρονικών στιγμών έναρξης.

```
set_Start_Time_Variable(S,[],Units):-
  S::[1..Units],!.                    %create Variable S {[1..Days*Hours]}
set_Start_Time_Variable(S,LVS,Units):-
  S::LVS,
  S#<=Units.
```

Οι εύκαμπτοι περιορισμοί μετέχουν στον αλγόριθμο αναζήτησης ως κριτήρια, οπότε θα τους αναφέρουμε παρακάτω.

4.4 Υλοποίηση Αλγορίθμου Χρονολογικής Οπισθοδρόμησης με Ευριστική Αναζήτηση

Για την εύρεση λύσης χρησιμοποιήσαμε τον αλγόριθμο χρονολογικής οπισθοδρόμησης με ευριστική αναζήτηση, όπου σύμφωνα με το [1] στην αναζήτηση πρώτης λύσης για το πρόβλημα, επιστρέφει αποτέλεσμα πιο γρήγορα σε σχέση με άλλους δημοφιλείς αλγορίθμους αναζήτησης, όπως ο iterative broadening (IB), depth-bounded discrepancy search (DDS) και limited discrepancy search (LDS). Ο αλγόριθμος large neighborhood search (LNS) δίνει επίσης καλά αποτελέσματα σύμφωνα με το [1] και ανήκει στους αλγορίθμους τοπικής αναζήτησης οι οποίοι χρησιμοποιούνται κυρίως σε προβλήματα βελτιστοποίησης.

Σύμφωνα με το [1], ο αλγόριθμος *χρονολογικής οπισθοδρόμησης* δίνει καλύτερα αποτελέσματα με βάση τις *ευριστικές διάταξης μεταβλητών* fairst-fail και Brelaz, όπου η *ευριστική διάταξης μεταβλητών* fairst-fail αποτελεί ουσιαστικά την ευριστική της *Περισσότερο Περιορισμένης Μεταβλητής ,* ενώ η *ευριστική διάταξης μεταβλητών* Brelaz αποτελεί συνδυασμό των ευριστικών της *Περισσότερο Περιοριστικής Μεταβλητής* και της *Περισσότερο Περιορισμένης Μεταβλητής* που θα αναφέρουμε παρακάτω. Τα αποτελέσματα για τον αλγόριθμο χρονολογικής οπισθοδρόμησης δεν είναι τόσο καλά στην περίπτωση των *ευριστικών διάταξης μεταβλητών* Rho, E(N) και Kappa, οι οποίες ανήκουν στην οικογένεια ευριστικών Kappa, για περισσότερες πληροφορίες ο αναγνώστης μπορεί να μελετήσει το [1].

```
labeling_dfs_ff([],_,_,_,Time,_,Heuristic,Heuristic):-
        NTime is cputime-Time,
        write_exdr(update_time,time(NTime)),
        flush(update_time),!.

labeling_dfs_ff(Tasklist,Bygroup,Byroom,Hours,Time,Room_heuristic,_,Heuristic):-
        deletefirstfailc(task(C,N,Num_st,S,D,VR),Tasklist,Rest),% sets variable heuristic
        makelistofvalues(C,S,D,Bygroup,Hours,Lval),          % sets value heristic
        member(triple(S,Heuristic1,NBygroup),Lval),      % evaluates S start time variable
        makelistofvalues(VR,Num_st,Byroom,LRval),      % sets value heristic for room variable
        member(pair(VR,Heuristic2),LRval),               % evaluates VR room variable
        NRoom_heuristic is Room_heuristic + Heuristic2,
```

```
NHeuristic is Heuristic1 + NRoom_heuristic,
labeling_dfs_ff(Rest,NBygroup,Byroom,Hours,Time,NRoom_heuristic,NHeuristic,Heuristic).
```

Πριν αναφέρουμε τα βήματα που ακολουθεί ο αλγόριθμος, πρέπει να επισημάνουμε το γεγονός ότι εφόσον υπάρχουν δύο μεταβλητές, χρόνου και αίθουσας, για κάθε διάλεξη για τις οποίες πρέπει να γίνει ανάθεση τιμών, κάποια από τις δύο πρέπει να προηγείται της ανάθεσης έναντι της άλλης. Στην προκειμένη περίπτωση επιλέγουμε την μεταβλητή χρόνου έναντι της μεταβλητής αίθουσας και αυτό γιατί, βάση της δικής μας προσέγγισης, η κύρια απόφαση αφορά τον χρόνο και όχι την αίθουσα, δηλαδή θεωρούμε ότι είναι πρωταρχικώς σκοπός μία διάλεξη να μπορεί να διδαχθεί στον χρόνο και έπειτα να βρεθεί αίθουσα διδασκαλίας. Στην περίπτωση που ένα από τα δύο δεν μπορεί να συμβεί, τότε ο αλγόριθμος οπισθοδρομεί.

Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

- **Επιλογή διάλεξης για ανάθεση τιμών στις μεταβλητές χρόνου *S* και αίθουσας *VR* :** η επιλογή της μεταβλητής πραγματοποιείται μέσω του κανόνα *deletefirstfailc(task(C,N,Num_st,S,D,VR),Tasklist,Rest),* όπου *task(C,N,Num_st,S,D,VR)* η διεργασία-διάλεξη της λίστας *Tasklist* και *Rest* η λίστα που απομένει αν από την λίστα *Tasklist* αφαιρέσουμε την επιλεγόμενη διεργασία. Ο κανόνας *deletefirstfailc/3* ουσιαστικά υλοποιεί την *ευριστική διάταξη μεταβλητών*, η οποία αποτελεί συνδυασμό των ευριστικών της *Περισσότερο Περιοριστικής Μεταβλητής* και της *Περισσότερο Περιορισμένης Μεταβλητής* που θα αναφέρουμε παρακάτω.
- **Διάταξη τιμών της μεταβλητής χρόνου *S* :** η διάταξη των τιμών της μεταβλητής χρόνου *S* πραγματοποιείται μέσω του κανόνα *makelistofvalues(C,S,D,Bygroup, Hours,Lval),* όπου διατάσσονται οι τιμές τις μεταβλητής στη λίστα *Lval* με την σειρά που οι τιμές πρέπει να ανατεθούν στην μεταβλητή σύμφωνα με τα κριτήρια που θα αναφέρουμε παρακάτω.
- **Ανάθεση τιμής στην μεταβλητή χρόνου *S* :** η ανάθεση τιμής της μεταβλητής χρόνου *S* πραγματοποιείται μέσω του κανόνα member(triple(S,Heuristic1,NBygroup), Lval). Στην περίπτωση που η τιμή που έχει επιλεγεί για την μεταβλητή δεν ικανοποιεί τους περιορισμούς, τότε επιλέγεται η επόμενη τιμής της λίστας *Lval*. Εάν δεν υπάρχει επόμενη τιμή να επιλεχθεί, τότε ο αλγόριθμος οπισθοδρομεί.
- **Διάταξη τιμών της μεταβλητής αίθουσας *VR* :** η διάταξη των τιμών της μεταβλητής αίθουσας *VR* πραγματοποιείται μέσω του κανόνα *makelistofvalues(VR, Num_st,Byroom,LRval)*, όπου διατάσσονται οι τιμές τις μεταβλητής στη λίστα *LRval* με την σειρά που οι τιμές πρέπει να ανατεθούν στην μεταβλητή σύμφωνα με τα κριτήρια που θα αναφέρουμε παρακάτω.
- **Ανάθεση τιμής στην μεταβλητή αίθουσας *VR* :** η ανάθεση τιμής της μεταβλητής αίθουσας *VR* πραγματοποιείται μέσω του κανόνα *member(pair(VR,Heuristic2),LRval)*. Στην περίπτωση που η τιμή που έχει επιλεγεί για την μεταβλητή δεν ικανοποιεί τους περιορισμούς, τότε επιλέγεται η επόμενη τιμής της λίστας *LRval*. Εάν δεν υπάρχει επόμενη τιμή να επιλεχθεί, τότε ο αλγόριθμος οπισθοδρομεί στην χρονική μεταβλητή έναρξης *S* επιλέγοντας την επόμενη της τιμής.
- **Υπολογισμό της συνάρτησης κόστους για τις αίθουσες :** ο υπολογισμός της συνάρτησης κόστους για τις αίθουσες πραγματοποιείται μέσω της σχέσης *NRoom_heuristic is Room_heuristic + Heuristic2*.

- **Υπολογισμό της συνολικής συνάρτησης κόστους :** ο υπολογισμός της συνολικής συνάρτησης κόστους πραγματοποιείται μέσω της σχέσης *NHeuristic is Heuristic1 + NRoom_heuristic.*

Στη περίπτωση τώρα που κανένας συνδυασμός τιμών του ζεύγους μεταβλητών χρόνου και αίθουσας δεν ικανοποιεί τους περιορισμούς, ο αλγόριθμος θα οπισθοδρομήσει στην προηγούμενη διάλεξη και θα κάνει καινούργια ανάθεση τιμών στις μεταβλητές της. Εάν τώρα δεν υπάρχει λύση τότε ο αλγόριθμος τερματίζει επιστρέφοντας *false*.

Στην περίπτωση η λίστα *Tasklist* να είναι κενή, τότε ο αλγόριθμος έχει βρει λύση και αποστέλλει τον χρόνο αναζήτησης της λύσης μέσω του κατηγορήματος *flush/1*.

4.4.1 Ευριστικές Διάταξης Μεταβλητών και Τιμών

Στην αναζήτηση λύσης με βάση την *Χρονολογική Οπισθοδρόμηση* κάναμε χρήση των ευριστικών διάταξης μεταβλητών και τιμών με σκοπό την πιο γρήγορη προσέγγιση λύση μέσα στο χώρο αναζήτησης και την εύρεση ποιοτικής λύσης. Η επιλογή μίας διάλεξης για την ανάθεση τιμών στις μεταβλητές της, ως προς τις άλλες διαλέξεις, στηρίζεται κυρίως στη μεταβλητή χρόνου *S* και όχι στη μεταβλητή αίθουσας και αυτό γιατί πρωταρχικός μας σκοπός είναι η διάλεξη να μπορεί να ανατεθεί χρονικά. Η ευριστική διάταξης μεταβλητών σε αυτή την περίπτωση για τις χρονικές μεταβλητές έναρξης S είναι ο συνδυασμός της *Περισσότερο Περιοριστική Μεταβλητή* με την *Περισσότερο Περιορισμένη Μεταβλητή*. Πιο συγκεκριμένα, από τις μεταβλητές που είναι υποψήφιες για ανάθεση τιμής επιλέγεται εκείνη η οποία συμμετέχει στους περισσότερους περιορισμούς (*Ευριστική της Περισσότερο Περιοριστικής Μεταβλητής*). Στην περίπτωση που δύο μεταβλητές συμμετέχουν στον ίδιο αριθμό περιορισμών, που είναι και ο μέγιστος, τότε επιλέγετε εκείνη η μεταβλητή με το μικρότερο πεδίο (*Ευριστική της Περισσότερο Περιορισμένης Μεταβλητής*).

Για παράδειγμα, έστω *5* διαλέξεις L_1, L_2, L_3, L_4 και L_5 με χρονικές μεταβλητές έναρξης S_1, S_2, S_3, S_4 και S_5 αντίστοιχα και πεδίο το διάστημα τιμών *{1..15}*. Αν τώρα η μεταβλητή S_3 συμμετέχει σε *15* περιορισμούς, οι μεταβλητές S_1, S_2 και S_5 σε *10*, και τέλος η S_4 σε *3* περιορισμούς τότε η επιλογή της πρώτης διάλεξης θα είναι εκείνης με τον μέγιστο αριθμό συμμετοχών σε περιορισμούς της χρονικής μεταβλητής και άρα της S_3. Έστω ότι στην μεταβλητή S_3 δίνεται η τιμή *1* και τα πεδία των υπολοίπων μεταβλητών επηρεάζονται όπως στο σχήμα 4.1.

	$S_1(10)$	$S_2(10)$	$S_3(15)$	$S_4(3)$	$S_5(10)$
Αρχικά πεδία	1..15	1..15	1..15	1..15	1..15
Μετά $S_3 = 1$	5..15	6..15	1	1..15	5..15
Μετά $S_2 = 6$	10..15	6	1	1..5,7..15	10..15
Μετά $S_1 = 10$	10	6	1	1..5,7..9,11..15	13..15
Μετά $S_5 = 13$	10	6	1	1..5,7..9	13
Μετά $S_4 = 1$	10	6	1	1	13

Σχήμα 4.1: Παράδειγμα 1 αναζήτηση λύσης με βάση την Χρονολογική Οπισθοδρόμηση

Παρατηρούμε ότι η επόμενη επιλογή μεταβλητής θα είναι μεταξύ των διαλέξεων L_1, L_2 και L_5 των οποίων οι μεταβλητές συμμετέχουν στον ίδιο αριθμό περιορισμών. Τα πεδία των μεταβλητών έχουν διαμορφωθεί ως εξής, για τις μεταβλητές S_1 και S_5 το διάστημα

τιμών *{5..15}* και για την μεταβλητή S_2 το διάστημα τιμών *{6..15}*. Εφόσον και οι τρεις μεταβλητές μετέχουν στον ίδιο αριθμό περιορισμών το κριτήριο βάση του οποίου θα γίνει η επιλογή της επόμενης μεταβλητής είναι εκείνο του μικρότερου πεδίου τιμών. Με βάση αυτό το κριτήριο η επιλογή της επόμενης μεταβλητής θα είναι η μεταβλητή S_2. Η επόμενη επιλογή μεταβλητής θα είναι πάλι μεταξύ των διαλέξεων L_1 και L_5. Στην περίπτωση όμως αυτή τα πεδία τιμών τους είναι ίσα ως προς το μέγεθος και επομένως οι επιλογές είναι ισότιμες και άρα η επιλογή γίνεται τυχαία, έστω της S_1. Στη συνέχεια θα επιλεχθεί η μεταβλητή S_5 και τέλος η μεταβλητή S_1 για ανάθεση.

Πρέπει να αναφέρουμε ότι αρχικά η επιλογή της ευριστικής διάταξης μεταβλητών ήταν της *Περισσότερο Περιορισμένης Μεταβλητής*, η οποία όμως δεν έδινε τόσο καλά αποτελέσματα για μεγάλο αριθμό δεδομένων τα οποία κατ' επέκταση μετείχαν και σε μεγάλο αριθμό περιορισμών. Με την χρήση της *Ευριστική της Περισσότερο Περιοριστικής Μεταβλητής* ως κύριο κριτήριο σε συνδυασμό με την *Ευριστική Περισσότερο Περιορισμένης Μεταβλητής* ο χρόνος αναζήτησης μειώθηκε δραστικά, δίνοντας πολύ καλύτερα αποτελέσματα.

Παρόλα αυτά, λόγω του γεγονότος ότι η διάρκεια κάθε διάλεξης του ίδιου μαθήματος ορίζεται από τον χρήστη και δεν παράγεται αυτόματα από το σύστημα έτσι ώστε να υπάρχει μία ομοιόμορφη κατανομή των ωρών στις διαλέξεις του ίδιου μαθήματος, υπάρχει πιθανότητα η διάρκεια κάποιων διαλέξεων να υπερβαίνει το *50%* των ωρών διδασκαλίας ανά ημέρα του ωρολογίου προγράμματος. Για παράδειγμα, έστω ότι οι ώρες διδασκαλίας ανά ημέρα του ωρολογίου προγράμματος είναι *14* και μία διάλεξη *L* έχει διάρκεια *10* ωρών, είναι φανερό ότι η διάλεξη αυτή θα έπρεπε να είναι από τις πρώτες που θα έπρεπε να προγραμματιστούν. Στη περίπτωση τώρα που η διάλεξη μετέχει σε μικρό αριθμό περιορισμών, τότε δυσκολεύεται η διαδικασία αναζήτησης λύσης στην περίπτωση που αυτή υπάρχει. Για παράδειγμα, εάν η διάλεξη *L* που αναφέραμε προηγουμένως μετέχει σε μία ομάδα μαθημάτων από τις *20* που μπορεί να υπάρχουν τότε εκείνο που θα συμβεί είναι ότι όλες οι διαλέξεις που μετέχουν σε μεγαλύτερο αριθμό περιορισμών από την *L* θα προηγηθούν στην ανάθεση τιμών. Όταν έρθει η στιγμή να γίνει ανάθεση τιμών για την διάλεξη *L*, το σύστημα να μην μπορεί να βρει κατάλληλη τιμή με αποτέλεσμα να πραγματοποιεί άσκοπες οπισθοδρομήσεις. Εάν ο χρήστης για αυτές τις περιπτώσεις των διαλέξεων ορίσει χρονικές στιγμές έναρξης η οποίες ουσιαστικά να επιτρέπουν στις διαλέξεις να προηγηθούν στην διαδικασία ανάθεση τιμών έναντι των υπολοίπων, τότε ο χρόνος αναζήτησης μειώνεται δραστικά.

Εφόσον έχει γίνει η επιλογή της χρονικής μεταβλητής S στη συνέχεια πρέπει να επιλεχθεί η τιμή που θα της αναθέσουν. Η επιλογή της υποψήφιας τιμής γίνεται με βάση τον γραμμικό συνδυασμό *10*cost1 + cost2* των **κριτηρίων 1 και 2** της παραγράφου 3.5 όπου το **κριτήριο 1** εξασφαλίζει την ομοιόμορφη κατανομή των διδακτικών ωρών για κάθε ομάδα μαθημάτων κατά την διάρκεια της εβδομάδας και το **κριτήριο 2** εξασφαλίζει τον ελάχιστο αριθμό κενών μεταξύ των μαθημάτων της ίδιας ομάδας κατά την διάρκεια κάθε ημέρας.

Για παράδειγμα, ας θεωρήσουμε πάλι ότι έχουμε *5* διαλέξεις, τις L_1, L_2, L_3, L_4 και L_5 οι οποίες έχουν διάρκεια *2,3,4,1* και *2* αντίστοιχα. Το ωρολόγιο πρόγραμμα αποτελεί από *3* ημέρες με *5* ώρες διδασκαλίας έκαστος.

Οι διαλέξεις L_1και L_2 ανήκουν στο μάθημα C_1, οι L_3 και L_5 ανήκουν στο μάθημα C_2 και τέλος η L_4 ανήκει στο μάθημα C_3. Τα μαθήματα C_1 και C_2 διδάσκονται από τον καθηγητή T_1 και το μάθημα C_3 διδάσκεται από τον καθηγητή T_2. Τα μαθήματα C_1 και C_3 ανήκουν στην ομάδα μαθημάτων G_1 και τα μαθήματα C_2 και C_3 ανήκουν στη ομάδα μαθημάτων

G_1. Εφαρμόζοντας τώρα τον **περιορισμό 1** οι χρονικές μεταβλητές έναρξης των διαλέξεων πρέπει να έχουν πεδία S_1={1..4,6..9,11..14}, S_2={1..3,6..8,11..13}, S_3={1,2,6,7,11,12}, S_4={1..15}, S_5={1..4,6..9,11..14}. Ο αριθμός των περιορισμών στις οποίες μετέχουν οι χρονικές μεταβλητές έναρξης των διαλέξεων θα είναι αντίστοιχα 7,8,9,4 και 7.

	$S_1(7)$	$S_2(8)$	$S_3(9)$	$S_4(4)$	$S_5(7)$
Αρχικά πεδία	1..4,6..9,11..14	1..3,6..8,11..13	1,2,6,7,11,12	1..15	1..4,6..9,11..14
Μετά $S_3 = 1$	6..9,11..14	6..8,11..13	1	5..15	6..9,11..14
Μετά $S_2 = 6$	11..14	6	1	5,9..15	9,11..14
Μετά $S_1 = 11$	11	6	1	5,13..15	9,13..14
Μετά $S_5 = 9$	11	6	1	5,13..15	9
Μετά $S_4 = 13$	11	6	1	13	9

Σχήμα 4.2: Παράδειγμα 2 αναζήτηση λύσης με βάση την Χρονολογική Οπισθοδρόμηση

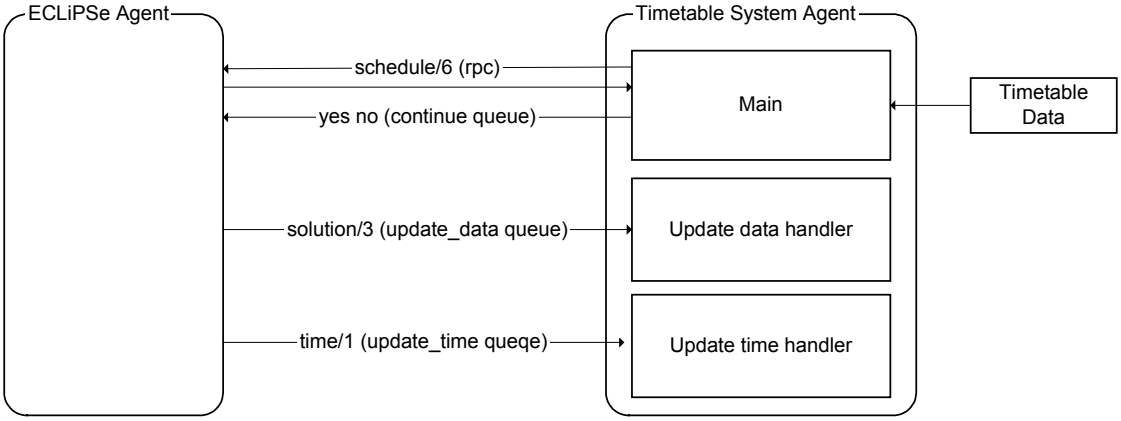
Η σειρά με την οποία θα γίνει ανάθεση τιμών στις χρονικές μεταβλητές έναρξης είναι η ίδια με αυτή τους σχήματος 4.2, σύμφωνα πάντα με ευριστική διάταξης μεταβλητών που έχουμε αναφέρει. Κάθε φορά που γίνεται ανάθεση τιμής σε μία μεταβλητή, από τα πεδία των υπολοίπων μεταβλητών αφαιρούνται τιμές σύμφωνα με τους περιορισμούς που ισχύουν. Στη περίπτωση τώρα που έχει γίνει ανάθεση τιμών στις μεταβλητές S_3, S_2, S_1 και S_5 και η μεταβλητή που επιλέγεται για ανάθεση είναι η S_4, θα πρέπει να επιλεγεί μία από τις τιμές του πεδίου {5,13..15}. Μέχρι στιγμής υπήρχε κατανομή των διαλέξεων, χωρίς κενά ανά ομάδα μαθημάτων και η συνάρτηση κόστους είναι ίση με 10*(4+3)+(0+0)=70. Αν τώρα υπολογίσουμε την συνάρτηση κόστους για κάθε μία από τις τιμές ξεχωριστά, προκύπτει ότι cost(5) = 70, cost(13) = 60, cost(14) = 61 και cost(15) = 62 και επομένως η τιμή που θα προτιμηθεί για την μεταβλητή S_4 είναι η 13.

Στη συνέχεια, εφόσον έχει πραγματοποιηθεί η ανάθεση τιμών για την χρονική μεταβλητή S θα πρέπει να επιλεγεί η υποψήφια τιμή για την αντίστοιχη μεταβλητή αίθουσας VR της διάλεξης. Η επιλογή της τιμής γίνεται με βάση το **κριτήριο 3** της παραγράφου 3.5 το οποίο εξασφαλίζει την κατανομή των αιθουσών ανάλογα με το μέγεθός τους στις διαλέξεις σε σχέση με τον αριθμό των φοιτητών κάθε διάλεξης. Στο κριτήριο θα μπορούσαν να είχαν χρησιμοποιηθεί και απόλυτοι αριθμοί όσο αφορά την χωρητικότητα κάθε αίθουσας και τον αριθμό των φοιτητών που παρακολουθούν κάθε διάλεξη, όπως αναφέρεται στα [4, 6, 12, 14, 18] ή θα μπορούσε να αποτελεί αυστηρό περιορισμό και όχι κριτήριο, όπως αναφέρεται στα [5, 6, 14]. Παρόλα αυτά ο λόγος για τον οποίο δεν επιλέξαμε κανένα από τους δύο τρόπους ή συνδυασμό αυτών είναι ότι συνήθως ο περιορισμός αυτός δεν ικανοποιείται και επομένως δεν μπορεί να αποτελεί αυστηρό περιορισμό αλλά και ως κριτήριο να μπορεί να εκφραστεί με όσο πιο φυσικό και απλό τρόπο γίνεται.

4.5 Επικοινωνία ECL^{i}PS^e – Συστήματος κατάρτισης ωρολογίου προγράμματος

Η επικοινωνία της ECL^{i}PS^e με το σύστημα κατάρτισης ωρολογίου προγράμματος φαίνεται στο σχήμα 4.3. Η δομή του προγράμματος αποτελείται από δύο πράκτορες, τον

πράκτορα της ECL^{i}PS^e και τον πράκτορα του συστήματος μας. Οι δύο πλευρές επικοινωνούν μέσω ERPC (ECL^{i}PS^e Remote Predicate Call ή και διαφορετικά RPC) και τριών ουρών. Ο πράκτορας του συστήματος μας δέχεται τα δεδομένα από τον χρήστη μέσω της διεπαφής που έχουμε αναπτύξει. Στη συνέχεια όταν ο χρήστης επιλέξει την έναρξη αναζήτηση λύσης, τότε μέσω του RPC το σύστημα της ECL^{i}PS^e καλείται να ικανοποιήσει τον σκοπό (goal) *schedule(Courses,Hours,Days,Teachers,Groups,Rooms)*. Η λίστα Courses περιέχει τα μαθήματα του ωρολογίου προγράμματος, η μεταβλητή *Hours* τις ώρες διδασκαλίας ανά ημέρα, η μεταβλητή *Days* τον αριθμό ημερών διδασκαλίας, η λίστα *Teachers* τους καθηγητές, η λίστα *Groups* τις ομάδες μαθημάτων και τέλος η λίστα *Rooms* τις αίθουσες του ωρολογίου προγράμματος. Σε αυτήν την φάση ο έλεγχος έχει περάσει από το σύστημα μας στο σύστημα της ECL^{i}PS^e και θα τον ξαναποκτήσουμε μόνο όταν ο σκοπός *schedule/6* εκτελεστεί. Για να μπορούμε όμως να έχουμε την εποπτεία της διαδικασίας γίνεται χρήση των παρακάτω ουρών:



Σχήμα 4.3 : Η επικοινωνία της ECL^{i}PS^e με το σύστημα κατάρτισης ωρολογίου προγράμματος

update_data : η ουρά αυτή διαβιβάζει από την ECL^{i}PS^e προς το σύστημά μας, κάθε φορά που γίνεται ανάθεση τιμών στις μεταβλητές αίθουσας και χρόνου μίας διάλεξης ή οπισθοδρόμηση κατά την διαδικασία αναζήτησης λύσης, του κατηγορήματος *solution(Ns,S,R)* όπου *Ns* το όνομα (κλειδί) της διάλεξη, *S* η τιμή της μεταβλητής χρόνου και *R* η τιμή της μεταβλητής αίθουσας. Σε περίπτωση οπισθοδρόμησης οι μεταβλητές *S* και *R* παίρνουν τις τιμές μηδέν έκαστος.

update_time: η ουρά αυτή διαβιβάζει από την ECL^{i}PS^e προς το σύστημά μας, κάθε φορά που βρίσκεται μία λύση, του κατηγορήματος *time(NTime)* όπου *NTime* ο χρόνος αναζήτησης της λύσης.

continue: η ουρά αυτή διαβιβάζει από το σύστημά μας προς την ECL^{i}PS^e, κάθε φορά που βρίσκεται μία λύση, του ατόμου (atom) *Continue* το οποίο μπορεί να πάρει τιμές *yes* ή *no* αναλόγως με το αν ο χρήστης επιθυμεί την αναζήτηση επόμενης λύσης ή τερματισμό της λειτουργίας.

Πρέπει να αναφέρουμε ότι για την αναζήτηση λύσης έχουμε δημιουργήσει διαφορετικό thread έτσι ώστε να μην περιορίζεται ο χρήστης σε σχέση με τις εργασίες που θέλει να εκτελέσει στην διεπαφή του συστήματος.

4.6 Περίληψη

Στο κεφάλαιο αυτό παρουσιάσαμε την αναπαράσταση των δεδομένων του προβλήματος όπως επίσης και την υλοποίηση των περιορισμών του. Κάναμε σύγκριση των μεθόδων υλοποίησης των περιορισμών, με βάση του πια μέθοδος παράγει το μικρότερο δέντρο παραγωγής της ECL^iPS^e. Τέλος παρουσιάσαμε τον αλγόριθμο χρονολογικής οπισθοδρόμησης με ευριστική αναζήτηση που σύμφωνα με το [1], ο οποίος δίνει τα καλύτερα αποτελέσματα όπως επίσης τον τρόπο επικοινωνίας της ECL^iPS^e με το σύστημα κατάρτισης ωρολογίου προγράμματος που αναπτύξαμε.

Κεφάλαιο 5

Η επικοινωνία χρήστη με το σύστημα

5.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται η επικοινωνία του χρήστη με το σύστημα μέσω της διεπαφής που έχουμε αναπτύξει. Γίνεται αναφορά στη λειτουργικότητα του συστήματος καθώς επίσης στις βασικές αρχές σχεδιασμού βάση των οποίων έγινε η υλοποίηση. Το σύστημα μας απευθύνεται σε αδαείς αλλά και έμπειρους χρήστες καθώς έγινε προσπάθεια το γραφικό περιβάλλον που παρέχεται να είναι απλό. Παρουσιάζονται επίσης τα πιο πιθανά σενάρια που μπορεί να υπάρξουν κατά την χρήση του συστήματος.

5.2 Βασικές Αρχές Σχεδιασμού

Η εφαρμογή υλοποιήθηκε σε Java αφενός επειδή είναι διαπλατφορμική και αφετέρου επειδή υπάρχουν έτοιμες βιβλιοθήκες για την επικοινωνία της Java με τον mysql server και το σύστημα της ECL^{i}PS^{e}. Κατά το σχεδιασμό του γραφικού περιβάλλοντος της εφαρμογής, έγινε προσπάθεια να εφαρμοστούν οι παρακάτω βασικές αρχές σχεδιασμού γραφικού περιβάλλοντος όπως αναφέρονται στο [29] :

- **Ορατότητα της Κατάστασης του Συστήματος (Visibility of System Status):** Η διεπαφή που υλοποιήθηκε ενημερώνει διαρκώς τον χρήστη σχετικά με το που βρίσκεται και τι κάνει. Αυτό επιτυγχάνεται με χρήση του status bar αλλά και με ορθών τίτλων στους διάλογους εφαρμογής.

- **Αντιστοίχιση μεταξύ Συστήματος και Πραγματικού Κόσμου (Match Between System and the Real World):** Κατά το σχεδιασμό του γραφικού περιβάλλοντος ελήφθην υπ' όψιν το κοινό στο οποίο αυτό απευθύνεται και έτσι χρησιμοποιήθηκαν όροι οικείοι και κατανοητοί για το σύνολο των ενδιαφερομένων.

- **Έλεγχος και Ελευθερία Χρήστη (User Control and Freedom):** Ο χρήστης στις περισσότερες περιπτώσεις έχει την ελευθερία να ενεργήσει με τον τρόπο και τη σειρά που θέλει ο ίδιος, χωρίς να είναι δεσμευμένος να ακολουθήσει μια ακριβή και συγκεκριμένη σειρά ενεργειών. Μειώθηκαν οι διάλογοι που ενημερώνουν τον χρήστη μόνο στους απαραίτητους, δηλαδή, σε εκείνους οπού εγκυμονεί κίνδυνος να απολεσθούν δεδομένα από κατά λάθος ενέργειες.

- **Συνέπεια και Πρότυπα (Consistency and Standards):** Σε όλη την εφαρμογή ακολουθείται η ίδια λογική σχεδιασμού (π.χ. ίδια χρώματα, κοινή ονομασία κουμπιών που εκτελούν την ίδια ενέργεια). Επίσης έγινε προσπάθεια ώστε η

εφαρμογή να ακολουθεί τα πρότυπα άλλων ευρέως χρησιμοποιούμενων εφαρμογών ώστε να είναι όσο το δυνατόν πιο οικεία στους χρήστες.

- **Αναγνώριση πάρα Μνήμη (Recognition rather than Recall):** Ο χρήστης δεν είναι υποχρεωμένος να θυμάται πληροφορία μεταξύ των ενεργειών που κάνει. Με αυτή την καθοδήγηση κατά νου, σε κάθε σημείο στην εφαρμογή υπάρχουν επαρκείς επεξηγήσεις ώστε ο χρήστης να καθοδηγείται στο πως θα κάνει αυτά που θέλει να κάνει. Οι επεξηγήσεις είναι σύντομα tool tips ή ακόμα και σύντομα labels που επεξηγούν την διαδικασία που μπορεί να ακολουθηθεί.

- **Αποτροπή Λαθών (Error Prevention):** Ο χρήστης εμποδίζεται από το να κάνει λάθη με το να του προσφέρονται όλες οι δυνατές επιλογές όπου χρειάζεται ώστε πότε να μην ενεργεί αυθαίρετα, στο όποιο κατά κύριο λόγω οφείλεται η πρόκληση λαθών.

- **Ευελιξία και Αποδοτικότητα Χρήσης (Flexibility and Efficiency of Use):** Το σύστημα απευθύνεται τόσο σε έμπειρους χρήστες όσο και σε λιγότερο έμπειρους ή ακόμα και αρχάριους επιτρέποντας τη ταυτόχρονη εξυπηρέτηση όλων των χρηστών, δίχως η εξυπηρέτηση των μεν να επιβαρύνει τους δε. Έχει γίνει αρκετή προσπάθεια να έχει η εφαρμογή καλή απόκριση. Οι όποιες καθυστερήσεις συμβαίνουν οφείλονται κατά κύριο λόγω στην τεράστια ποσότητα πληροφορίας που πρέπει να φορτωθεί από την Βάση. Για την καλύτερη απόδοση κατά την χρήση της εφαρμογής η πληροφορία, αυτή, φορτώνεται κατά την εκκίνηση της εφαρμογής. Τα περισσότερα δεδομένα του συστήματος κρατούνται στην μνήμη ώστε να αποφεύγεται η συχνή επικοινωνία με την βάση δεδομένων, η οποία είναι χρονοβόρα.

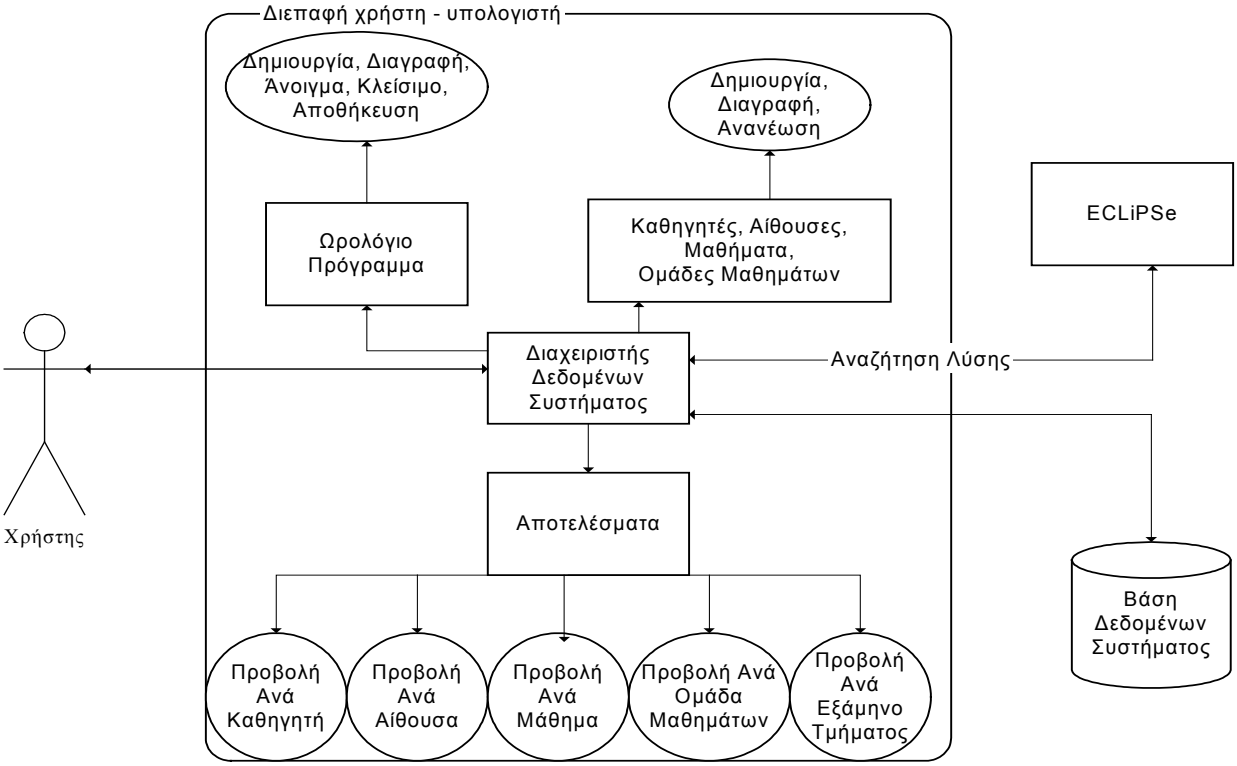
- **Αισθητική και Μινιμαλιστική Σχεδίαση (Aesthetic and Minimalistic Design):** Το φαίνεσθαι (Look & Feel) της εφαρμογής είναι το ίδιο πάντα το ίδιο με αυτό του λειτουργικού συστήματος, ώστε να είναι πιο κοντά στις αισθητικές προτιμήσεις του χρήστη. Εκτός αυτού, στο χρήστη δεν παρέχονται περιττές πληροφορίες και οι επεξηγήσεις παραμένουν σύντομες, διότι αυτό θα εμποδίσει τον χρήστη να εστιάσει την προσοχή του στην πληροφορία. Ακόμη και στην περίπτωση των λαθών, η επισήμανση αυτών γίνεται με τρόπο σύντομο ώστε να φαίνεται ξεκάθαρα το μήνυμα που θέλουν να δώσουν στο χρήστη.

- **Βοήθεια Χρηστών στην Αναγνώριση, Διάγνωση και Ανάνηψη των Λαθών (Help Users Recognize, Diagnose, and Recover from Errors):** Τα μηνύματα λάθους είναι λακωνικά, εύκολα κατανοητά και απόλυτα ακριβή έτσι ώστε να τα καταλαβαίνει αμέσως ο χρήστης και να μπορεί να διορθώσει εύκολα τα λάθη του.

5.3 Λειτουργικότητα

Σε αυτήν την παράγραφο θα κάνουμε λεπτομερή ανάλυση της λειτουργικότητας που παρέχεται από την εφαρμογή. Για τον σκοπό αυτό θα χρησιμοποιηθούν τόσο διαγράμματα όσο και Use Cases.

Το αντικείμενο της εφαρμογής είναι η κατάρτιση και επεξεργασία ωρολογίου προγράμματος. Πιο αναλυτικά ο χρήστης πρέπει να μπορεί να δημιουργήσει νέο ωρολόγιο πρόγραμμα ή να φορτώσει τα δεδομένα ενός προϋπάρχων από την βάση δεδομένων. Στη συνέχεια εφόσον το ωρολόγιο πρόγραμμα είναι ανοιχτό μπορεί να δημιουργεί, να προσθέτει, να επεξεργάζεται ή και να σβήνει πληροφορία που βρίσκεται στο ωρολόγιο πρόγραμμα του. Οι καθηγητές, οι αίθουσες, τα μαθήματα και οι ομάδες των μαθημάτων είναι κοινά για όλα τα ωρολόγια προγράμματα, δεδομένου ότι η εφαρμογή απευθύνεται στην γραμματεία ενός πανεπιστήμιου όπου τα δεδομένα είναι σταθερά και υπεισέρχονται μικρές αλλαγές κάθε φορά. Επομένως η επανάληψη πληροφορίας για κάθε ωρολόγιο πρόγραμμα ξεχωριστά – όπως ένας καθηγητής, μία αίθουσα κτλ.- η οποία ήδη υπάρχει είναι άσκοπη και χρονοβόρα. Ο χρήστης μπορεί να αναζητήσει λύση και στην περίπτωση που τα αποτελέσματα δεν τον ικανοποιούν, να ζητήσει από το σύστημα εύρεσης επόμενης λύσης. Τα αποτελέσματα των λύσεων παρέχονται στον χρήστη μέσω πέντε διαφορετικών τρόπων έτσι ώστε να έχει καλύτερη εποπτεία. Η όλη λειτουργικότητα του εργαλείου παρουσιάζεται γραφικά στο παρακάτω διάγραμμα.



Σχήμα 5.1 : Η λειτουργικότητα του εργαλείου

5.4 Use Cases

Τα Use Cases είναι ένας εύκολος και συνοπτικός τρόπος παρουσίασης της λειτουργικότητας της εφαρμογής. Με τα Use Cases μπορούν να παρουσιαστούν όλα τα

βήματα μίας διαδικασίας τόσο αν εκτελείτε σωστά όσο και στις περιπτώσεις που κάτι δεν εξελιχτεί όπως αναμενόταν.

5.4.1 USE CASE 1 : Δημιουργία νέου ωρολογίου προγράμματος

USE CASE 1	Δημιουργία νέου ωρολογίου προγράμματος	
Goal In Context	Ο χρήστης θέλει να δημιουργήσει νέο ωρολόγιο πρόγραμμα.	
Scope & Level	Σύστημα , Primary Task	
Precondition		
Success End Condition	Αποθηκεύεται το ωρολόγιο πρόγραμμα στην Βάση.	
Failed End Condition	Το ωρολόγιο πρόγραμμα δεν αποθηκεύεται (system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης ξεκινάει την εφαρμογή	
Steps	**1**	Ανάκτηση από την Βάση Δεδομένων των ονομάτων όλων των ωρολογίων προγραμμάτων.
	2	Ο χρήστης δηλώνει τα χαρακτηριστικά (όνομα, ημέρες, περίοδο διδασκαλίας κτλ.)που επιθυμεί να έχει το ωρολόγιο πρόγραμμα.
	3	Αποθήκευση του ωρολογίου προγράμματος στην Βάση Δεδομένων
Extensions	**1α**	Η σύνδεση με την Βάση δεν είναι δυνατή. **1α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**2α**	Υπάρχει ήδη ωρολόγιο πρόγραμμα με την ίδια ονομασία. Ο χρήστης ενημερώνεται **2α1**. επαναλαμβάνει το βήμα **2** **2α2**. ακυρώνει την διαδικασία δημιουργίας νέου ωρολογίου προγράμματος.
	2β	Ο χρήστης επιλέγει η ώρα έναρξης των μαθημάτων να είναι μεγαλύτερη από την ώρα λήξης. Ο χρήστης ενημερώνεται **2α1**. επαναλαμβάνει το βήμα **2** **2α2**. ακυρώνει την διαδικασία δημιουργίας νέου ωρολογίου προγράμματος.

Πίνακας 5.1 : USE CASE 1- Δημιουργία νέου ωρολογίου προγράμματος

5.4.2 USE CASE 2 : Επεξεργασία ωρολογίου προγράμματος

USE CASE 2	Επεξεργασία ωρολογίου προγράμματος
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί ένα ωρολόγιο πρόγραμμα.

Scope & Level	Σύστημα , Sub-function.	
Precondition	Να έχει ξεκινήσει η εφαρμογή. Να έχει επιλέξει κάποιο ωρολόγιο πρόγραμμα	
Success End Condition	Αποθηκεύεται το ωρολόγιο πρόγραμμα στην Βάση.	
Failed End Condition	Το ωρολόγιο πρόγραμμα δεν αποθηκεύεται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger		
Steps	**1**	Ανάκτηση από την Βάση Δεδομένων του επιλεχθέντος από τον χρήστη ωρολόγιο πρόγραμμα με βάση το όνομά του (Timetable Name).
	2	Επεξεργασία των καθηγητών. (**USE CASE 3**)
	3	Επεξεργασία των αιθουσών. (**USE CASE 4**)
	4	Επεξεργασία των μαθημάτων. (**USE CASE 5**)
	5	Επεξεργασία των ομάδων μαθημάτων. (**USE CASE 6**)
	6	Επιλογή των διαλέξεων που μετέχουν στο ωρολόγιο πρόγραμμα.
	7	Δήλωση των χρονικών στιγμών έναρξης για τις διαλέξεις που μετέχουν στην κατάρτιση ωρολογίου προγράμματος.
	8	Αναζήτηση λύσης.
	9	Προβολή των αποτελεσμάτων.
	10	Αποθήκευση του ωρολογίου προγράμματος στην Βάση Δεδομένων
Extensions	**1α**	Η σύνδεση με την Βάση δεν είναι δυνατή. **1α1**. Ο χρήστης ενημερώνεται.
	8α	Η σύνδεση με την ECL^{i}PS^{e} δεν είναι δυνατή. **8α1**. Ο χρήστης ενημερώνεται και το πρόγραμμα τερματίζει.
Sub-Variations	**2α**	Ο χρήστης επιλέγει να μην επεξεργαστεί τους καθηγητές.
	3α	Ο χρήστης επιλέγει να μην επεξεργαστεί τις αίθουσες.
	4α	Ο χρήστης επιλέγει να μην επεξεργαστεί τα μαθήματα.
	5α	Ο χρήστης επιλέγει να μην επεξεργαστεί τις ομάδες μαθημάτων.
	6α	Ο χρήστης επιλέγει να μην συμπεριλάβει διαλέξεις που να μετέχουν στο ωρολόγιο πρόγραμμα.

	7α	Ο χρήστης επιλέγει να μην δηλώσει χρονικές στιγμές έναρξης για τις διαλέξεις που μετέχουν στην κατάρτιση ωρολογίου προγράμματος. **7α1**. Το πρόγραμμα θεωρεί ως πιθανές χρονικές στιγμές έναρξης των διαλέξεων όλες τις χρονικές στιγμές του προβλήματος.
	8α	Ο χρήστης δεν έχει δηλώσει καθηγητές ή σε κάποια από τα μαθήματα δεν έχει κάνει ανάθεση καθηγητών. Ο χρήστης ενημερώνεται. **8α.1**. επαναλαμβάνει το βήμα **2.**
	8β	Ο χρήστης δεν έχει δηλώσει αίθουσες. Ο χρήστης ενημερώνεται. **8β.1**. επαναλαμβάνει το βήμα **3.**
	8γ	Ο χρήστης δεν έχει δηλώσει μαθήματα. Ο χρήστης ενημερώνεται. **8γ.1**. επαναλαμβάνει το βήμα **4.**
	8δ	Ο χρήστης δεν έχει συμπεριλάβει διαλέξεις που να μετέχουν στο ωρολόγιο πρόγραμμα. Ο χρήστης ενημερώνεται. **8δ.1**. επαναλαμβάνει το βήμα **6.**
	8ε	Δεν υπάρχει λύση για το πρόβλημα ή έχει παρέλθει η διάρκεια των 15’ λεπτών χωρίς εύρεση λύσης. Ο χρήστης ενημερώνεται. **8ε.1**. χαλαρώνει κάποιους από τους περιορισμούς.
	9α	Δεν υπάρχουν αποτελέσματα.
	10α	Ο χρήστης επιλέγει να μην αποθηκεύσει τα δεδομένα του τρέχοντος ωρολογίου προγράμματος. Επιλέγει έξοδο από το πρόγραμμα ή εκκίνηση νέου ωρολογίου προγράμματος. **10α1**. Το πρόγραμμα ενημερώνει τον χρήστη.

Πίνακας 5.2 : USE CASE 2 - Επεξεργασία ωρολογίου προγράμματος

5.4.3 USE CASE 3 : Επεξεργασία καθηγητών

USE CASE 3	Επεξεργασία καθηγητών.
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των καθηγητών.
Scope & Level	Σύστημα , Sub-function.
Precondition	Να έχει ξεκινήσει η εφαρμογή.
Success End Condition	Αποθηκεύονται οι αλλαγές των καθηγητών στην Βάση.

Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποιο καθηγητή.	
Steps	**1**	Ο χρήστης επιλέγει κάποιο όνομα για τον καθηγητή.
	2	Ο χρήστης επιλέγει ή αφαιρεί χρονικές στιγμές για τις οποίες ο καθηγητής δεν είναι διαθέσιμος.
	3	Ο χρήστης επιλέγει να διαγράψει τον καθηγητή.
	4	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	4α	Η σύνδεση με την Βάση δεν είναι δυνατή. **4α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**1α**	Υπάρχει ήδη καθηγητής με την ίδια ονομασία ή έχει δηλώσει κενό όνομα. Ο χρήστης ενημερώνεται **1α1**. επαναλαμβάνει το βήμα **1** **1α2**. ακυρώνει την όλη διαδικασία.
	2α	Ο χρήστης επιλέγει να μην κάνει αλλαγές στις χρονικές στιγμές για τις οποίες ο καθηγητής δεν είναι διαθέσιμος.
	3α	Ο χρήστης ενημερώνεται για την διαδικασία που θέλει να εκτελέσει. **3α1**. επιλέγει την εκτέλεση της διαδικασίας. **3α2**. ακυρώνει την όλη διαδικασία.
	4α	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.3 : USE CASE 3 - Επεξεργασία καθηγητών

5.4.4 USE CASE 4 : Επεξεργασία αιθουσών

USE CASE 4	Επεξεργασία αιθουσών.
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των αιθουσών.
Scope & Level	Σύστημα , Sub-function.
Precondition	Να έχει ξεκινήσει η εφαρμογή.
Success End Condition	Αποθηκεύονται οι αλλαγές των αιθουσών στην Βάση.
Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)
Primary, Secondary Actor	Χρήστης
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποια αίθουσα.

Steps	**1**	Ο χρήστης επιλέγει κάποιο όνομα για την αίθουσα.
	2	Ο χρήστης επιλέγει ή αφαιρεί χρονικές στιγμές για τις οποίες η αίθουσα δεν είναι διαθέσιμη.
	3	Ο χρήστης επιλέγει να διαγράψει την αίθουσα.
	4	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	4α	Η σύνδεση με την Βάση δεν είναι δυνατή. **4α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**1α**	Υπάρχει ήδη αίθουσα με την ίδια ονομασία ή έχει δηλώσει κενό όνομα. Ο χρήστης ενημερώνεται **1α1**. επαναλαμβάνει το βήμα **1** **1α2**. ακυρώνει την όλη διαδικασία.
	2α	Ο χρήστης επιλέγει να μην κάνει αλλαγές στις χρονικές στιγμές για τις οποίες η αίθουσα δεν είναι διαθέσιμη.
	3α	Ο χρήστης ενημερώνεται για την διαδικασία της διαγραφής που θέλει να εκτελέσει. **3α1**. επιλέγει την εκτέλεση της διαδικασίας. **3α2**. ακυρώνει την όλη διαδικασία.
	4α	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.4 : USE CASE 4 - Επεξεργασία αιθουσών

5.4.5 USE CASE 5 : Επεξεργασία μαθημάτων

USE CASE 5	Επεξεργασία μαθημάτων.	
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των μαθημάτων.	
Scope & Level	Σύστημα , Sub-function.	
Precondition	Να έχει ξεκινήσει η εφαρμογή.	
Success End Condition	Αποθηκεύονται οι αλλαγές των μαθημάτων στην Βάση.	
Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποιο μάθημα.	
Steps	**1**	Ο χρήστης επιλέγει κάποιο όνομα για το μάθημα.

	2	Ο χρήστης επιλέγει τον καθηγητή που διδάσκει το μάθημα.
	3	Ο χρήστης επεξεργάζεται τα στιγμιότυπα του μαθήματος. (**USE CASE 7**)
	4	Ο χρήστης εξεργάζεται τις διαλέξεις του μαθήματος. (**USE CASE 8**)
	5	Ο χρήστης εξεργάζεται τα υπόλοιπα δεδομένα του μαθήματος.
	6	Ο χρήστης επιλέγει να διαγράψει το μάθημα.
	7	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	7α	Η σύνδεση με την Βάση δεν είναι δυνατή. **7α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**1α**	Υπάρχει ήδη το μάθημα ή έχει δηλώσει κενό όνομα. Ο χρήστης ενημερώνεται **1α1**. επαναλαμβάνει το βήμα **1** ή το βήμα **3** **1α2**. ακυρώνει την όλη διαδικασία.
	2α	Ο χρήστης επιλέγει να μην αναθέσει καθηγητή στο μάθημα.
	3α	Ο χρήστης επιλέγει να μην δημιουργήσει στιγμιότυπα του μαθήματος.
	4α	Ο χρήστης επιλέγει να μην δημιουργήσει διαλέξεις του μαθήματος.
	6α	Ο χρήστης ενημερώνεται για την διαδικασία της διαγραφής που θέλει να εκτελέσει. **6α1**. επιλέγει την εκτέλεση της διαδικασίας. **6α2**. ακυρώνει την όλη διαδικασία.
	7α	Ο χρήστης δεν έχει δημιουργήσει στιγμιότυπα του μαθήματος. Ο χρήστης ενημερώνεται **7α1**. επαναλαμβάνει το βήμα **3** **7α2**. ακυρώνει την όλη διαδικασία.
	7β	Ο χρήστης δεν έχει δημιουργήσει διαλέξεις του μαθήματος. Ο χρήστης ενημερώνεται **7β1**. επαναλαμβάνει το βήμα **4** **7β2**. ακυρώνει την όλη διαδικασία.
	7γ	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.5 : USE CASE 5 - Επεξεργασία μαθημάτων

5.4.6 USE CASE 6 : Επεξεργασία ομάδων μαθημάτων

USE CASE 6	Επεξεργασία ομάδων μαθημάτων.
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των ομάδων μαθημάτων.

Scope & Level	Σύστημα , Sub-function.	
Precondition	Να έχει ξεκινήσει η εφαρμογή.	
Success End Condition	Αποθηκεύονται οι αλλαγές των ομάδων μαθημάτων στην Βάση.	
Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποια ομάδα μαθημάτων.	
Steps	**1**	Ο χρήστης επιλέγει κάποιο όνομα για την ομάδα μαθημάτων.
	2	Ο χρήστης προσθέτει ή αφαιρεί κάποια μαθήματα που μετέχουν στην ομάδα.
	3	Ο χρήστης επιλέγει να διαγράψει την ομάδα μαθημάτων.
	4	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	4α	Η σύνδεση με την Βάση δεν είναι δυνατή. **4α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**1α**	Υπάρχει ήδη ομάδα μαθημάτων με την ίδια ονομασία ή έχει δηλώσει κενό όνομα. Ο χρήστης ενημερώνεται **1α1**. επαναλαμβάνει το βήμα **1** **1α2**. ακυρώνει την όλη διαδικασία.
	2α	Ο χρήστης επιλέγει να μην κάνει αλλαγές στο σύνολο των μαθημάτων που μετέχουν στην ομάδα.
	3α	Ο χρήστης ενημερώνεται για την διαδικασία που θέλει να εκτελέσει. **3α1**. επιλέγει την εκτέλεση της διαδικασίας. **3α2**. ακυρώνει την όλη διαδικασία.
	4α	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.6 : USE CASE 6 - Επεξεργασία ομάδων μαθημάτων

5.4.7 USE CASE 7 : Επεξεργασία των στιγμιότυπων ενός μαθήματος

USE CASE 7	Επεξεργασία στιγμιότυπων ενός μαθήματος.
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των στιγμιότυπων.
Scope & Level	Σύστημα , Sub-function.

Precondition	Να έχει ξεκινήσει η εφαρμογή. Να έχει επιλεγεί κάποιο μάθημα.	
Success End Condition	Αποθηκεύονται οι αλλαγές στην Βάση.	
Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποιο στιγμιότυπο μαθήματος.	
Steps	**1**	Ο χρήστης επιλέγει το τμήμα στο οποίο διδάσκεται το μάθημα.
	2	Ο χρήστης επιλέγει το τομέα στον οποίο ανήκει το μάθημα.
	3	Ο χρήστης επεξεργάζεται τα υπόλοιπα δεδομένα.
	4	Ο χρήστης επιλέγει να διαγράψει το στιγμιότυπο του μαθήματος..
	5	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	4α	Η σύνδεση με την Βάση δεν είναι δυνατή. **4α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**1α**	Υπάρχει ήδη το στιγμιότυπο του μαθήματος ή έχει δηλώσει κενό όνομα. Ο χρήστης ενημερώνεται **1α1**. επαναλαμβάνει το βήμα **1** **1α2**. ακυρώνει την όλη διαδικασία.
	2α	Ο χρήστης δεν δηλώνει το όνομα του τομές στον οποίο ανήκει το μάθημα. Ο χρήστης ενημερώνεται **2α1**. επαναλαμβάνει το βήμα **2** **2α2**. ακυρώνει την όλη διαδικασία.
	4α	Ο χρήστης ενημερώνεται για την διαδικασία που θέλει να εκτελέσει. **4α1**. επιλέγει την εκτέλεση της διαδικασίας. **4α2**. ακυρώνει την όλη διαδικασία.
	5α	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.7 : USE CASE 7 - Επεξεργασία των στιγμιότυπων ενός μαθήματος

5.4.8 USE CASE 8 : Επεξεργασία των διαλέξεων ενός μαθήματος

USE CASE 8	Επεξεργασία διαλέξεων ενός μαθήματος.

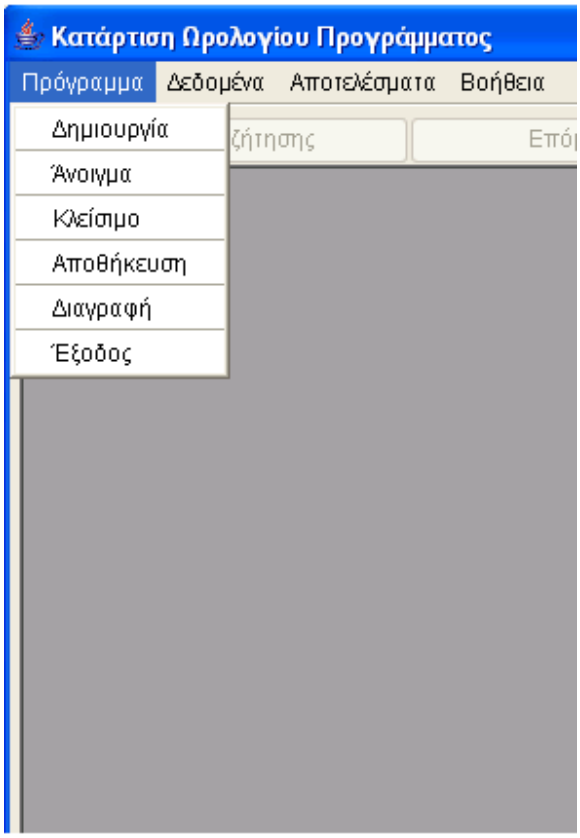
Goal In Context	Ο χρήστης θέλει να επεξεργαστεί τα δεδομένα των διαλέξεων.	
Scope & Level	Σύστημα , Sub-function.	
Precondition	Να έχει ξεκινήσει η εφαρμογή. Να έχει επιλεγεί κάποιο μάθημα.	
Success End Condition	Αποθηκεύονται οι αλλαγές στην Βάση.	
Failed End Condition	Οι αλλαγές δεν αποθηκεύονται (ο χρήστης δεν θέλει ή system crash)	
Primary, Secondary Actor	Χρήστης	
Trigger	Ο χρήστης επιλέγει ή δημιουργεί κάποια διάλεξη μαθήματος.	
Steps	**1**	Ο χρήστης επιλέγει το είδος της διάλεξης και τη διάρκεια της. Αναλόγως με το είδος του εμφανίζονται και οι αντίστοιχες αίθουσες.
	2	Ο χρήστης επιλέγει τις αίθουσες στις οποίες μπορεί να διδαχθεί η διάλεξη.
	3	Ο χρήστης επιλέγει να διαγράψει την διάλεξη του μαθήματος..
	4	Αποθήκευση των αλλαγών στην Βάση Δεδομένων.
Extensions	**1α**	Ο χρήστης ακυρώνει την όλη διαδικασία.
	4α	Η σύνδεση με την Βάση δεν είναι δυνατή. **4α1**. Ο χρήστης ενημερώνεται.
Sub-Variations	**2α**	Ο χρήστης επιλέγει να μην δηλώσει αίθουσες στις οποίες μπορεί να διδαχθεί η διάλεξη. **2α1**. το σύστημα θεωρεί ότι όλες οι αίθουσες που αντιστοιχούν στην διάλεξη είναι υποψήφιες.
	3α	Ο χρήστης ενημερώνεται για την διαδικασία που θέλει να εκτελέσει. **3α1**. επιλέγει την εκτέλεση της διαδικασίας. **3α2**. ακυρώνει την όλη διαδικασία.
	5α	Ο χρήστης ακυρώνει την όλη διαδικασία.

Πίνακας 5.8 : USE CASE 8 - Επεξεργασία των διαλέξεων ενός μαθήματος

Παρακάτω, αναλύονται όλες οι ενέργειες που αναφέρονται στο σχήμα 5.1 και στα Use Cases και αντιπροσωπεύουν την λειτουργικότητα του εργαλείου, από την κατάρτιση ενός νέου ωρολογίου προγράμματος έως και την αποθήκευσή του. Βεβαίως, τα ενδιάμεσα βήματα μπορούν να επαναληφθούν όσες φορές είναι απαραίτητο και με οποιαδήποτε σειρά.

5.5 Το μενού

Κατά την εκκίνηση της εφαρμογής εμφανίζεται στο χρήστη το κύριο πλαίσιο εργασίας, στο οποίο υπάρχει το μενού επιλογών όπως φαίνεται στο σχήμα.5.2. Το βασικό μενού παρέχει στον χρήστη επιλογές και λειτουργίες όπως για παράδειγμα η δημιουργία νέου ωρολογίου προγράμματος, η επεξεργασία μαθημάτων και άλλες επιλογές.



Σχήμα 5.2 : Το μενού

Ο πίνακας 5.9 παρουσιάζει το μενού επιγραμματικά με όλες τις λειτουργίες που παρέχει.

Όνομα μενού	Λειτουργία
Πρόγραμμα	
Δημιουργία	Δημιουργεί ένα νέο ωρολόγιο πρόγραμμα.
Άνοιγμα	Φορτώνει ένα ωρολόγιο πρόγραμμα από την βάση.
Κλείσιμο	Κλείνει το ωρολόγιο πρόγραμμα.
Αποθήκευση	Αποθηκεύει το ωρολόγιο πρόγραμμα στη βάση.
Διαγραφή	Διαγράφει το ωρολόγιο πρόγραμμα από την βάση.
Έξοδος	Τερματίζει την εφαρμογή.

Όνομα μενού	Λειτουργία
Δεδομένα	
Καθηγητές	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να επεξεργαστεί τους καθηγητές.
Αίθουσες	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να επεξεργαστεί τις αίθουσες.
Μαθήματα	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να επεξεργαστεί τα μαθήματα.
Ομάδες μαθημάτων	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να επεξεργαστεί τις ομάδες μαθημάτων.
Αποτελέσματα	
Ανά μάθημα	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά μάθημα.
Ανά καθηγητή	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά καθηγητή.
Ανά αίθουσα	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά αίθουσα.
Ανά ομάδα μαθημάτων	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά ομάδα μαθημάτων.
Ανά εξάμηνο τμήματος	Ανοίγει τον διάλογο από τον οποίο ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης λύσης για το ωρολόγιο πρόγραμμα, ανά εξάμηνο τμήματος.
Βοήθεια	
Σχετικά	Ανοίγει ένα μήνυμα που αφορά την εφαρμογή

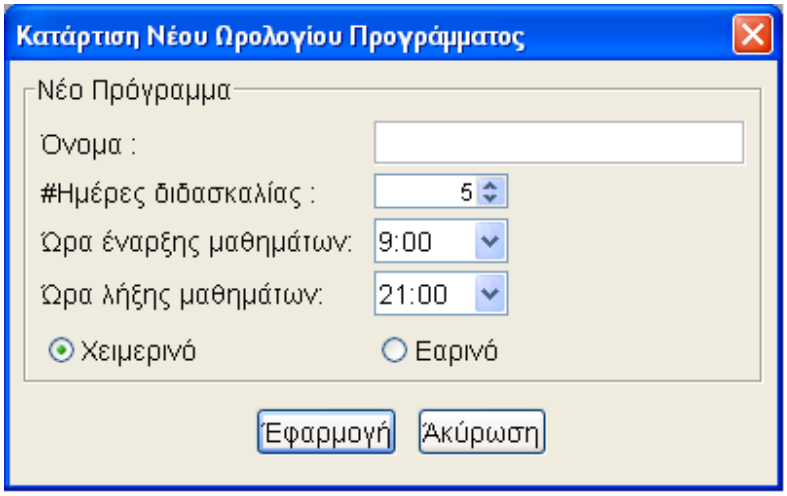
Πίνακας 5.9 : Το μενού επιγραμματικά με όλες τις λειτουργίες που παρέχει

5.6 Το ωρολόγιο πρόγραμμα μαθημάτων

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία ενός ωρολογίου προγράμματος.

5.6.1 Δημιουργία νέου ωρολογίου προγράμματος

Επιλέγοντας μέσω του μενού την επιλογή Πρόγραμμα → Δημιουργία, εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.3. Στο διάλογο παρατηρούμε ότι ο χρήστης καλείται να συμπληρώσει το όνομα του νέου προγράμματος, τον αριθμό των ημερών διδασκαλίας, τις ώρες έναρξης και λήξης των μαθημάτων και τέλος να επιλέξει την περίοδο, χειμερινή ή εαρινή, του προγράμματος. Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.

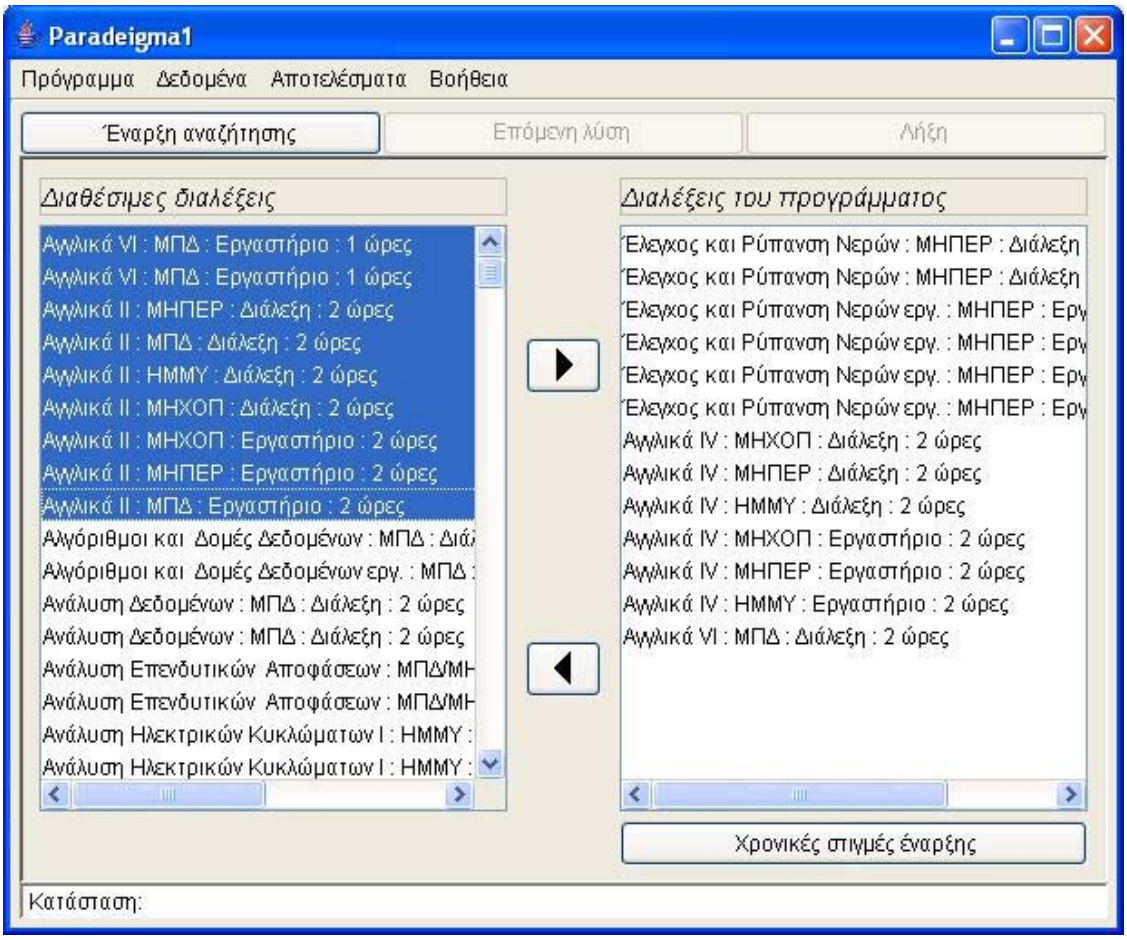


Σχήμα 5.3 : Ο διάλογος για την δημιουργία νέου ωρολογίου προγράμματος

5.6.2 Επεξεργασία ωρολογίου προγράμματος

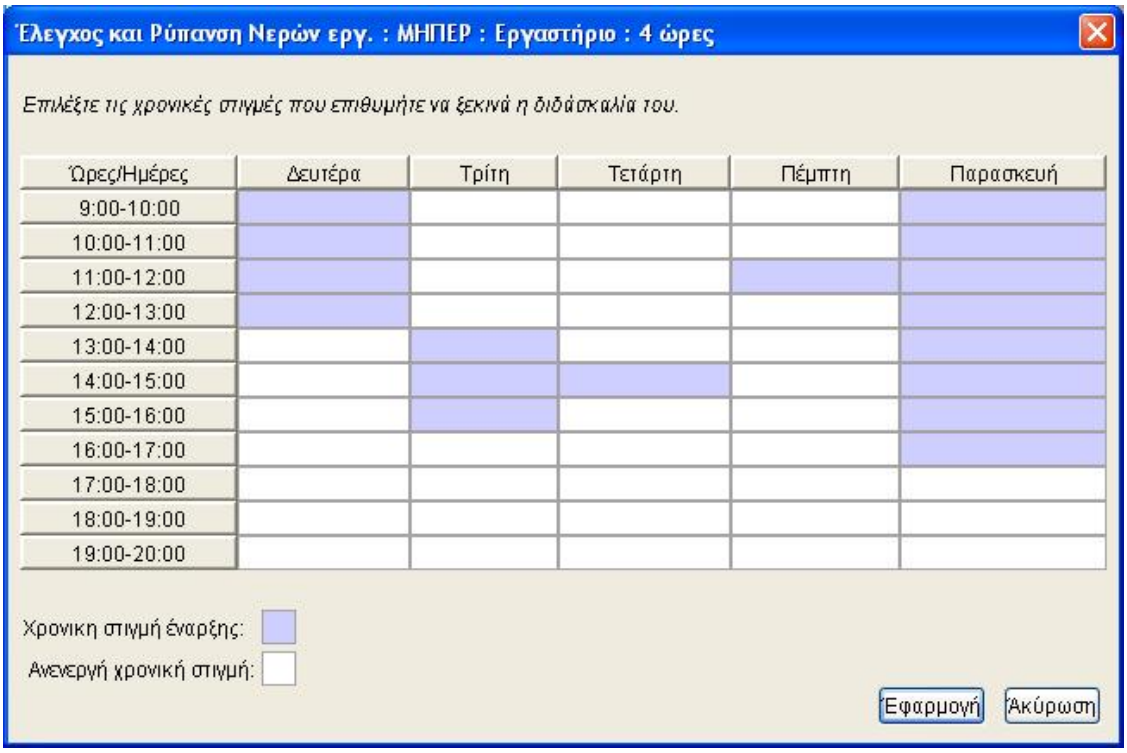
Η επεξεργασία ενός ωρολογίου προγράμματος πραγματοποιείται στο κύριο πλαίσιο εργασίας όπως φαίνεται στο σχήμα 5.4. Στο σχήμα φαίνεται το μενού επιλογών, η μπάρα εργαλείων (Tool bar) η οποία περιέχει την ομάδα κουμπιών "Έναρξη αναζήτησης", "Επόμενη λύση" και "Λήξη" που αφορούν την αναζήτηση λύσης για την οποία θα μιλήσουμε αργότερα, τις δύο λίστες των διαλέξεων, δύο κουμπιά τα οποία επιτρέπουν την μεταφορά διαλέξεων από την μία λίστα στην άλλη και τέλος το κουμπί "Χρονικές στιγμές έναρξης" που επιτρέπει την δήλωση των χρονικών στιγμών έναρξης για κάθε διάλεξη της δεξιάς λίστας, ξεχωριστά. Αναλόγως με το εάν το ωρολόγιο πρόγραμμα ανήκει στη εαρινή ή χειμερινή περίοδο εμφανίζονται στον χρήστη οι διαλέξεις των μαθημάτων της αντίστοιχης περιόδου. Για παράδειγμα εάν το ωρολόγιο πρόγραμμα ανήκει στην εαρινή περίοδο τότε του εμφανίζονται μόνο οι διαλέξεις των μαθημάτων που ανήκουν στα ζυγά εξάμηνα. Σε περίπτωση που ο χρήστης δεν έχει δηλώσει μαθήματα για την συγκεκριμένη περίοδο τότε οι λίστες είναι κενές. Στην αριστερή λίστα εμφανίζονται οι διαλέξεις των μαθημάτων που είναι διαθέσιμες αλλά δεν μετέχουν στο ωρολόγιο πρόγραμμα. Ο χρήστης έχει την δυνατότητα να μεταφέρει στην δεξιά λίστα, η

οποία περιέχει τις διαλέξεις που μετέχουν στο ωρολόγιο πρόγραμμα, τις διαλέξεις που επιθυμεί να μετέχουν στο ωρολόγιο πρόγραμμα και αντιστρόφως. Οι διαλέξεις των μαθημάτων εμφανίζονται σε κάθε λίστα σε αλφαβητική σειρά με βάση το όνομα του μαθήματος που ανήκουν, το τμήμα, το είδος (ασκήσεις, διάλεξη ή εργαστήριο) και τέλος την διάρκεια.



Σχήμα 5.4 : Βασικό πλαίσιο επεξεργασία ωρολογίου προγράμματος

Η δεξιά λίστα επιτρέπει την επιλογή μίας διάλεξης κάθε φορά. Επιλέγοντας διάλεξη της δεξιά λίστα και πατώντας το κουμπί “Χρονικές στιγμές έναρξης”, εμφανίζεται ο διάλογος του σχήματος 5.5. Ο διάλογος επιτρέπει στον χρήστη να δηλώσει τις χρονικές στιγμές έναρξης της αντίστοιχης διάλεξης. Με μοβ χρώμα περιέχονται οι επιλεγμένες χρονικές στιγμές στις οποίες ο χρήστης επιθυμεί να ξεκινά η διδασκαλία της διάλεξης. Σε περίπτωση που χρήστης δεν επιθυμεί να δηλώσει συγκεκριμένες χρονικές στιγμές, τότε το σύστημα αυτομάτως θεωρεί ως υποψήφιες όλες οι χρονικές στιγμές που παρουσιάζονται στο σχήμα κάθε φορά.



Σχήμα 5.5 : Ο διάλογος για δήλωση των χρονικών στιγμών έναρξης της αντίστοιχης διάλεξης

5.6.3 Διαγραφή ωρολογίου προγράμματος

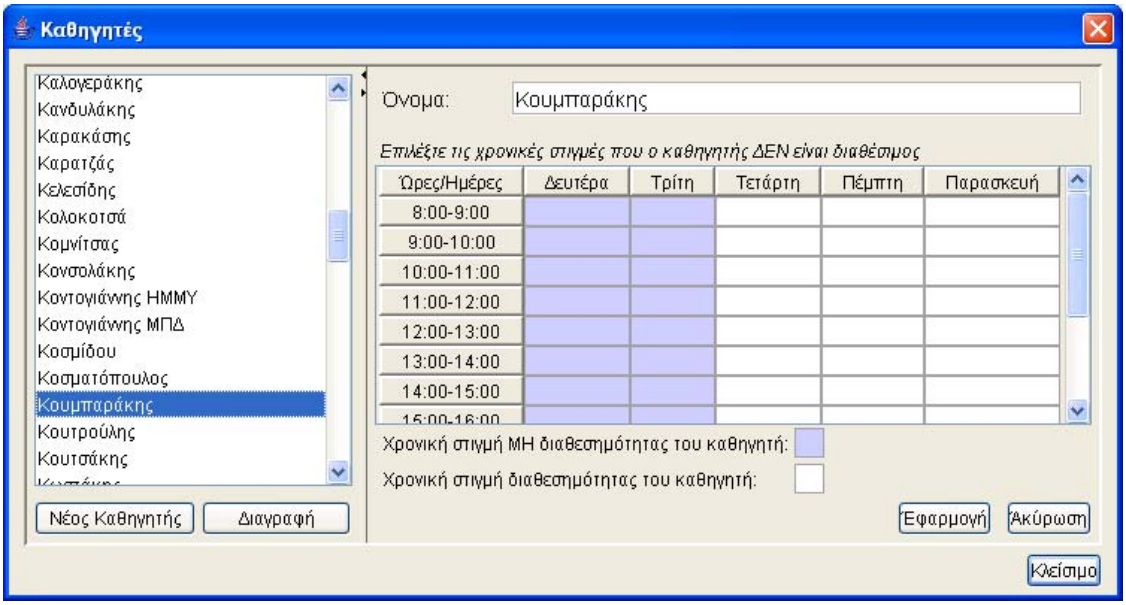
Η διαγραφή του ωρολογίου προγράμματος από την βάση δεδομένων πραγματοποιείται επιλέγοντας μέσω του μενού την επιλογή Πρόγραμμα → Διαγραφή, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

5.7 Οι καθηγητές

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία ενός καθηγητή.

5.7.1 Επεξεργασία - δημιουργία καθηγητή

Επιλέγοντας μέσω του μενού την επιλογή Δεδομένα → Καθηγητές εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.6. Η λίστα εμφανίζει όλους τους καθηγητές σε αλφαβητική σειρά, ενώ με την ομάδα κουμπιών "Νέος καθηγητής" και "Διαγραφή" δίνεται η δυνατότητα στον χρήστη να δημιουργήσει νέο καθηγητή ή να διαγράψει τον επιλεγμένο καθηγητή της λίστας. Τα χαρακτηριστικά του καθηγητή είναι το όνομα του και οι χρονικές στιγμές για τις οποίες ο καθηγητής δεν είναι διαθέσιμος. Οι επιλεγμένες χρονικές στιγμές για τις οποίες ο καθηγητής δεν είναι διαθέσιμος απεικονίζονται με μοβ χρώμα. Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.



Σχήμα 5.6 : Ο διάλογος επεξεργασίας - δημιουργίας καθηγητή

5.7.2 Διαγραφή καθηγητή

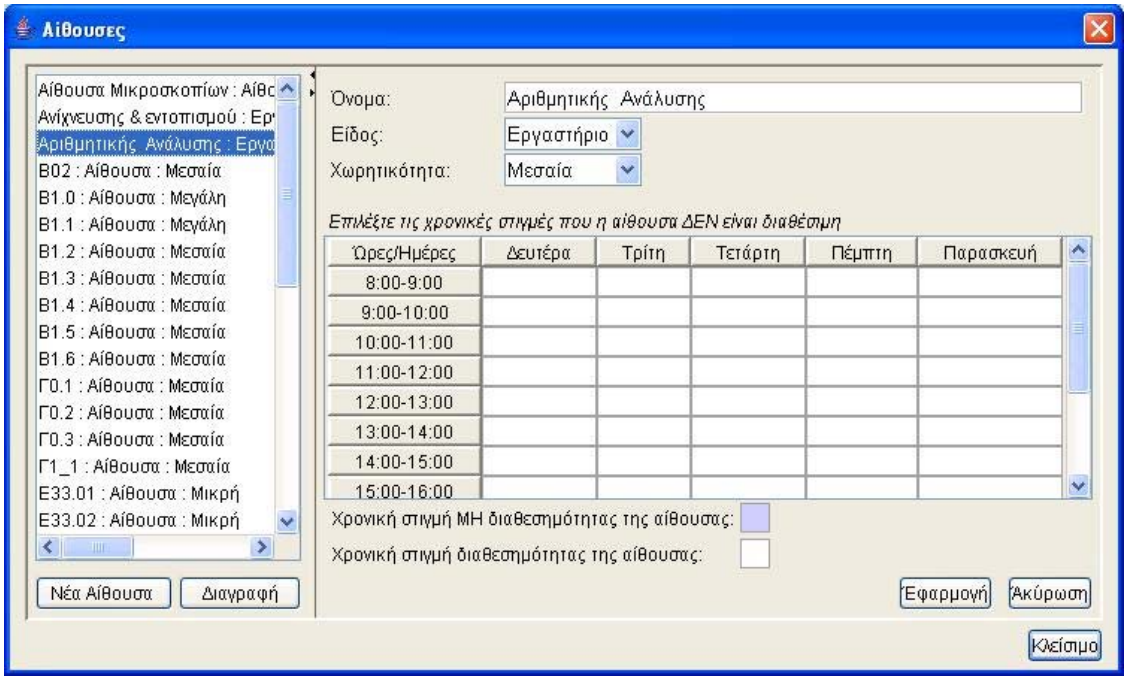
Η διαγραφή ενός καθηγητή από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

5.8 Οι αίθουσες

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία μίας αίθουσας.

5.8.1 Επεξεργασία - δημιουργία αίθουσας

Επιλέγοντας μέσω του μενού την επιλογή Δεδομένα → Αίθουσες εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.7. Η λίστα εμφανίζει όλες τις αίθουσες σε αλφαβητική σειρά, ενώ με την ομάδα κουμπιών "Νέα αίθουσα" και "Διαγραφή" δίνεται η δυνατότητα στον χρήστη να δημιουργήσει νέα αίθουσα ή να διαγράψει την επιλεγμένη αίθουσα της λίστας. Τα χαρακτηριστικά της αίθουσας είναι το όνομα της, το είδος (εάν δηλαδή είναι αίθουσα διδασκαλίας ή εργαστηριακή), η χωρητικότητα της (μικρή, μεσαία ή μεγάλη) και οι χρονικές στιγμές για τις οποίες η αίθουσα δεν είναι διαθέσιμη. Οι επιλεγμένες χρονικές στιγμές για τις οποίες η αίθουσα δεν είναι διαθέσιμη απεικονίζονται με μοβ χρώμα. Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.



Σχήμα 5.7 : Ο διάλογος επεξεργασίας - δημιουργίας αίθουσας

5.8.2 Διαγραφή αίθουσας

Η διαγραφή μίας αίθουσας από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

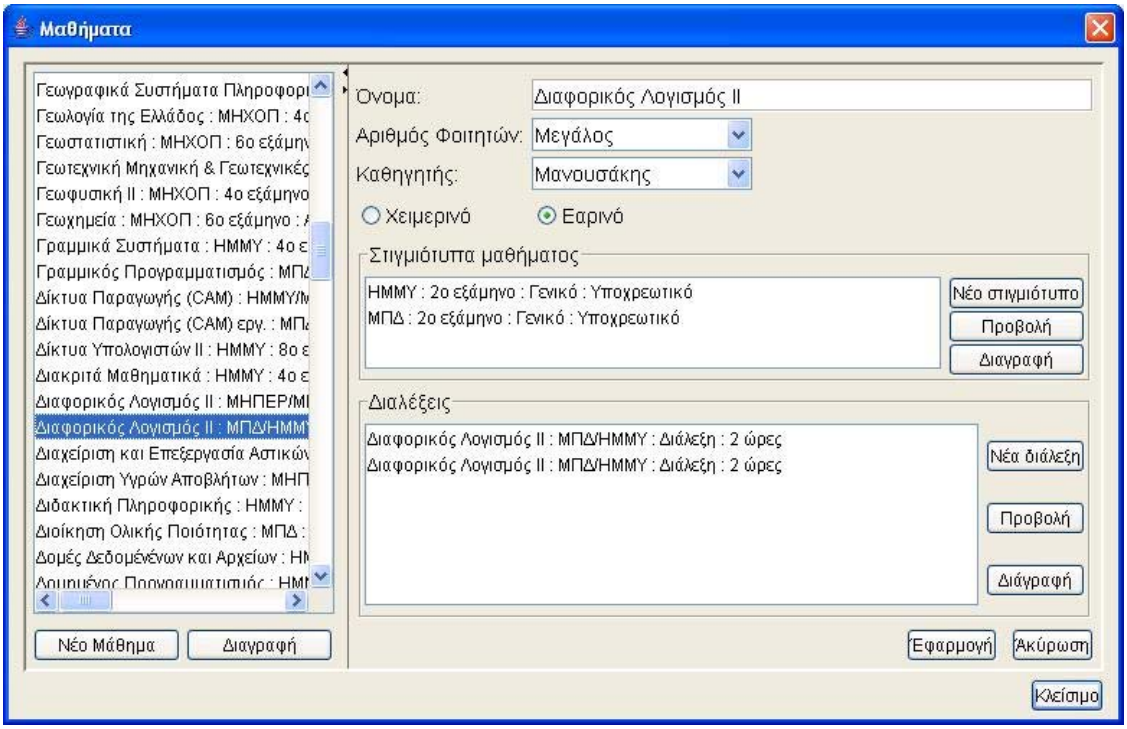
5.9 Τα μαθήματα, οι διαλέξεις τους και τα στιγμιότυπα των μαθημάτων

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία ενός μαθήματος, των στιγμιότυπων και των διαλέξεων του.

5.9.1 Επεξεργασία - δημιουργία μαθήματος

Επιλέγοντας μέσω του μενού την επιλογή Δεδομένα → Μαθήματα εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.8. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλα τα μαθήματα σε αλφαβητική σειρά με βάση το όνομα, το τμήμα και το εξάμηνο στο οποίο διδάσκεται, ενώ με την ομάδα κουμπιών "Νέο μάθημα" και "Διαγραφή" δίνεται η δυνατότητα στον χρήστη να δημιουργήσει νέο μάθημα ή να διαγράψει το επιλεγμένο μάθημα της λίστας. Τα χαρακτηριστικά του μαθήματος είναι το όνομα του, ο αριθμός των φοιτητών που το παρακολουθούν (εάν δηλαδή είναι μικρός, μεσαίος ή μεγάλος), ο καθηγητής που διδάσκει το μάθημα ο οποίος μπορεί να είναι και κενός, σε πια περίοδο (Χειμερινή ή Εαρινή) διδάσκεται το μάθημα, τα στιγμιότυπα του μαθήματος και οι διαλέξεις του. Οι καθηγητές εμφανίζονται στο Combobox με αλφαβητική σειρά. Η λογική των στιγμιότυπων ενός μαθήματος βασίζεται στο γεγονός ότι ένα μάθημα μπορεί να παρακολουθείται από ένα ή και περισσότερα τμήματα. Σε

αυτήν την περίπτωση αντί να δημιουργούμε περισσότερα από ένα μαθήματα για κάθε τμήμα ξεχωριστά, που είναι λάθος καθώς το μάθημα είναι ουσιαστικά ένα, δημιουργούμε ένα μάθημα και στιγμιότυπα αυτού διατηρώντας έτσι την πληροφορία ότι το μάθημα διδάσκεται σε περισσότερα από ένα τμήματα. Μέσω της ομάδας κουμπιών "Νέο στιγμιότυπο", "Προβολή" και " Διαγραφή " δίνεται η δυνατότητα στον χρήστη να δημιουργήσει, να προβάλει ή να διαγράψει ένα στιγμιότυπο της λίστας. Όπως έχουμε αναφέρει ένα μάθημα αποτελείται από διαλέξεις τις οποίες περιέχονται στην λίστα των διαλέξεων. Μέσω της ομάδας κουμπιών "Νέα διάλεξη", "Προβολή" και " Διαγραφή " δίνεται η δυνατότητα στον χρήστη να δημιουργήσει, να προβάλει ή να διαγράψει μία διάλεξη της λίστας. Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.



Σχήμα 5.8 : Ο διάλογος επεξεργασίας - δημιουργίας μαθήματος

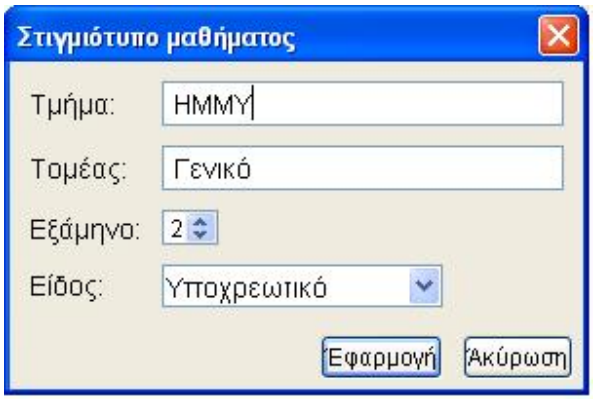
5.9.2 Διαγραφή μαθήματος

Η διαγραφή ενός μαθήματος από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

5.9.3 Επεξεργασία - δημιουργία στιγμιότυπου μαθήματος

Επιλέγοντας όπως προαναφέραμε ένα στιγμιότυπο του μαθήματος από την λίστα και πατώντας το κουμπί "Προβολή" ή "Νέο στιγμιότυπο" εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.9. Ο χρήστης καλείται να δηλώσει το τμήμα στο οποίο διδάσκεται, τον τομέα του τμήματος στον οποίο ανήκει, το εξάμηνο διδασκαλίας και το

είδος του μαθήματος (εάν δηλαδή είναι υποχρεωτικό ή επιλογής). Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.



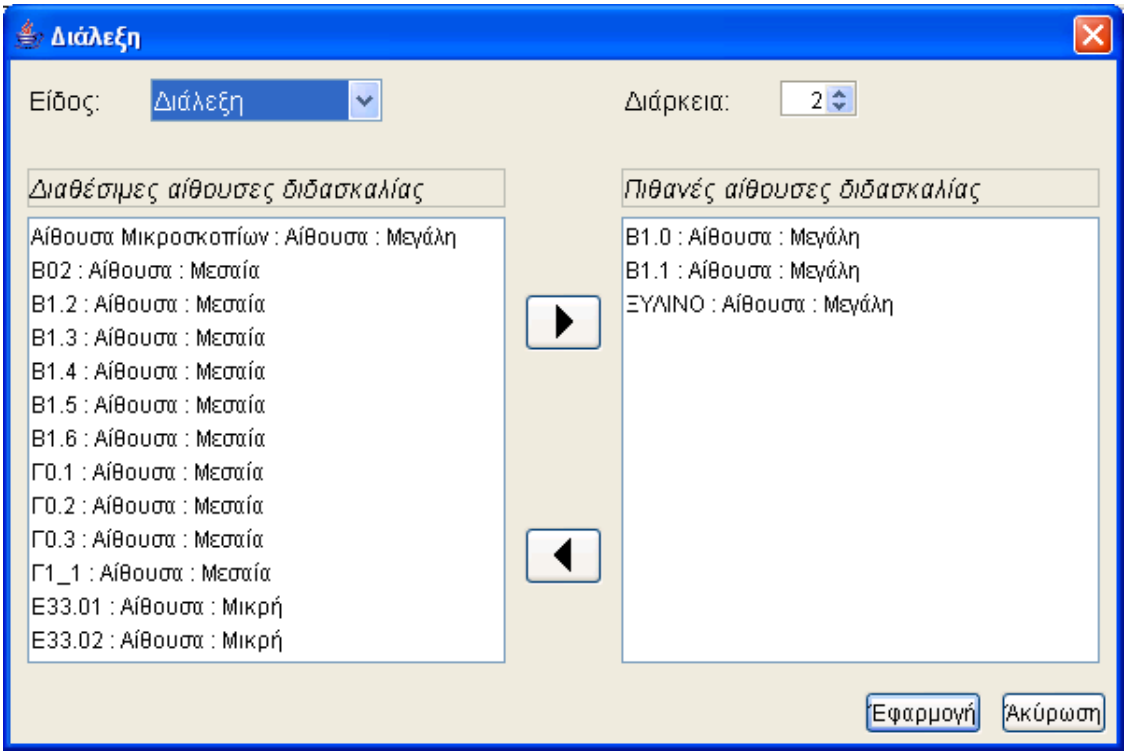
Σχήμα 5.9 : Ο διάλογος επεξεργασίας - δημιουργίας στιγμιότυπου μαθήματος

5.9.4 Διαγραφή στιγμιότυπου μαθήματος

Η διαγραφή ενός στιγμιότυπο μαθήματος από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

5.9.5 Επεξεργασία - δημιουργία νέας διάλεξης

Επιλέγοντας όπως προαναφέραμε μία διάλεξη του μαθήματος από την λίστα και πατώντας το κουμπί "Προβολή" ή "Νέα διάλεξη" εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.10. Ο χρήστης καλείται να δηλώσει το είδος της διάλεξης (εάν δηλαδή είναι διάλεξη, εργαστήριο ή ασκήσεις), την διάρκεια διδασκαλίας και τις αίθουσες στις οποίες μπορεί να διδαχθεί. Η αριστερή λίστα περιέχει τις αίθουσες οι οποίες είναι διαθέσιμες αλλά δεν διδάσκεται η διάλεξη και η δεξιά περιέχει τις αίθουσες στις οποίες μπορεί να διδαχθεί η διάλεξη. Εάν τώρα ο χρήστης δεν επιλέξει αίθουσες διδασκαλίας για την διάλεξη, τότε το σύστημα θεωρεί ως υποψήφιες αίθουσες όλες τις αίθουσες που αντιστοιχούν στην διάλεξη. Αναλόγως με το αν ο χρήστης επιλέγει μία διάλεξη να είναι εργαστήριο, ασκήσεις ή διάλεξη, το σύστημα θα του εμφανίζει και τις ανάλογες αίθουσες στις οποίες μπορεί να διδακτή. Πιο συγκεκριμένα, αν μία διάλεξη είναι εργαστηριακή τότε θα εμφανιστούν στο χρήστη μόνο εργαστηριακές αίθουσες για επιλογή και αντιστρόφως. Στη περίπτωση τώρα που στην βάση δεδομένων δεν υπάρχουν καθόλου δηλωμένες αίθουσες και οι δύο λίστες θα είναι κενές. Οι λίστες εμφανίζουν τις αίθουσες με αλφαβητική σειρά. Σε περίπτωση που τα δεδομένα είναι εσφαλμένα εμφανίζονται στον χρήστη τα αντίστοιχα μηνύματα λάθους.



Σχήμα 5.10 : Ο διάλογος επεξεργασίας - δημιουργίας νέας διάλεξης

5.9.6 Διαγραφή διάλεξης

Η διαγραφή μίας διάλεξης ενός μαθήματος από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

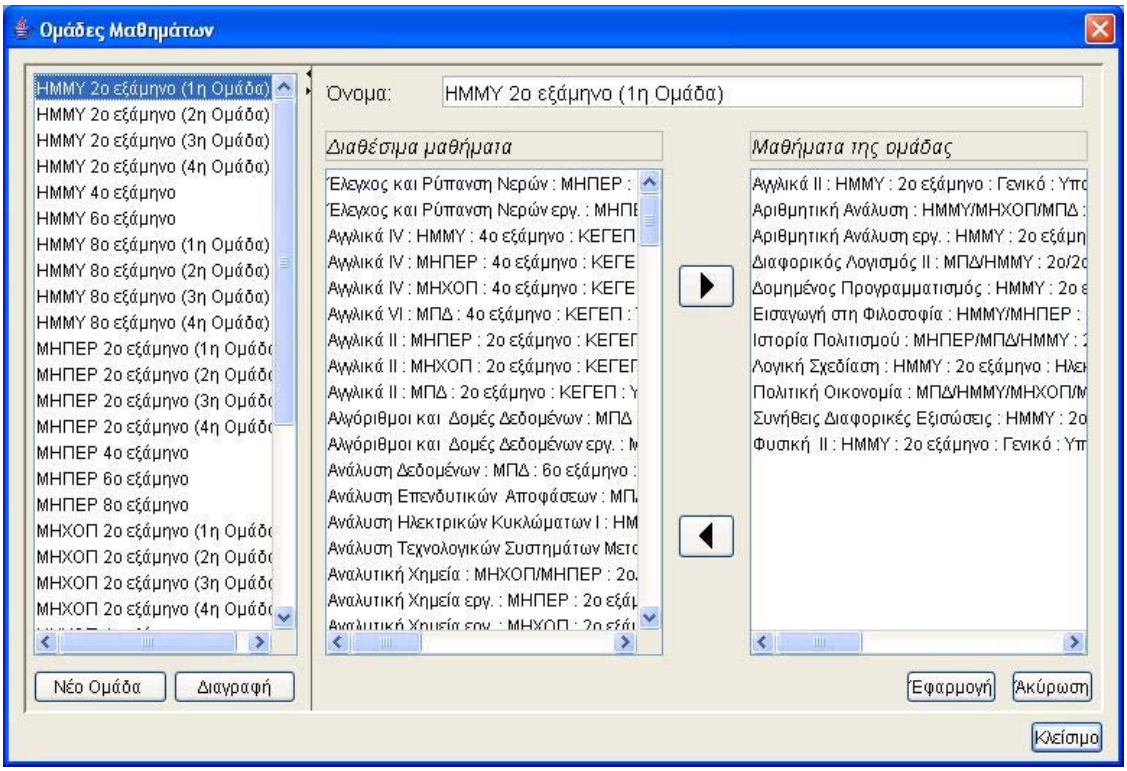
5.10 Οι ομάδες μαθημάτων

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία μίας ομάδας μαθημάτων.

5.10.1 Επεξεργασία - δημιουργία ομάδας μαθημάτων

Επιλέγοντας μέσω του μενού την επιλογή Δεδομένα → Ομάδες μαθημάτων εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.11. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλες τις ομάδες μαθημάτων σε αλφαβητική σειρά, ενώ με τα ομάδα κουμπιών "Νέα ομάδα" και "Διαγραφή" δίνεται η δυνατότητα στον χρήστη να δημιουργήσει νέα ομάδα μαθημάτων ή να διαγράψει την επιλεγμένη ομάδα της λίστας. Τα χαρακτηριστικά της ομάδας μαθημάτων είναι το όνομα της και τα μαθήματα που μετέχουν σε αυτή. Παρατηρούμε ότι ο διάλογος περιέχει άλλες δύο λίστες. Η αριστερή λίστα περιέχει τα μαθήματα που είναι διαθέσιμα ενώ η δεξιά λίστα περιέχει τα μαθήματα της ομάδας. Η εμφάνιση των μαθημάτων και στις δύο λίστες γίνεται με αλφαβητική σειρά. Σε περίπτωση όπου στην βάση δεδομένων δεν υπάρχουν μαθήματα, τότε και οι

δύο λίστες είναι κενές. Σε περίπτωση λανθασμένων δεδομένων ο χρήστης ειδοποιείται με μήνυμα από το σύστημα.



Σχήμα 5.11 : Ο διάλογος επεξεργασίας - δημιουργίας ομάδας μαθημάτων

5.10.2 Διαγραφή ομάδας μαθημάτων

Η διαγραφή μίας ομάδας μαθημάτων από την βάση δεδομένων πραγματοποιείται μέσω του κουμπιού "Διαγραφή" όπως προαναφέραμε, όπου ο χρήστης ενημερώνεται για την για την λειτουργία που θα πραγματοποιηθεί μέσω μηνύματος.

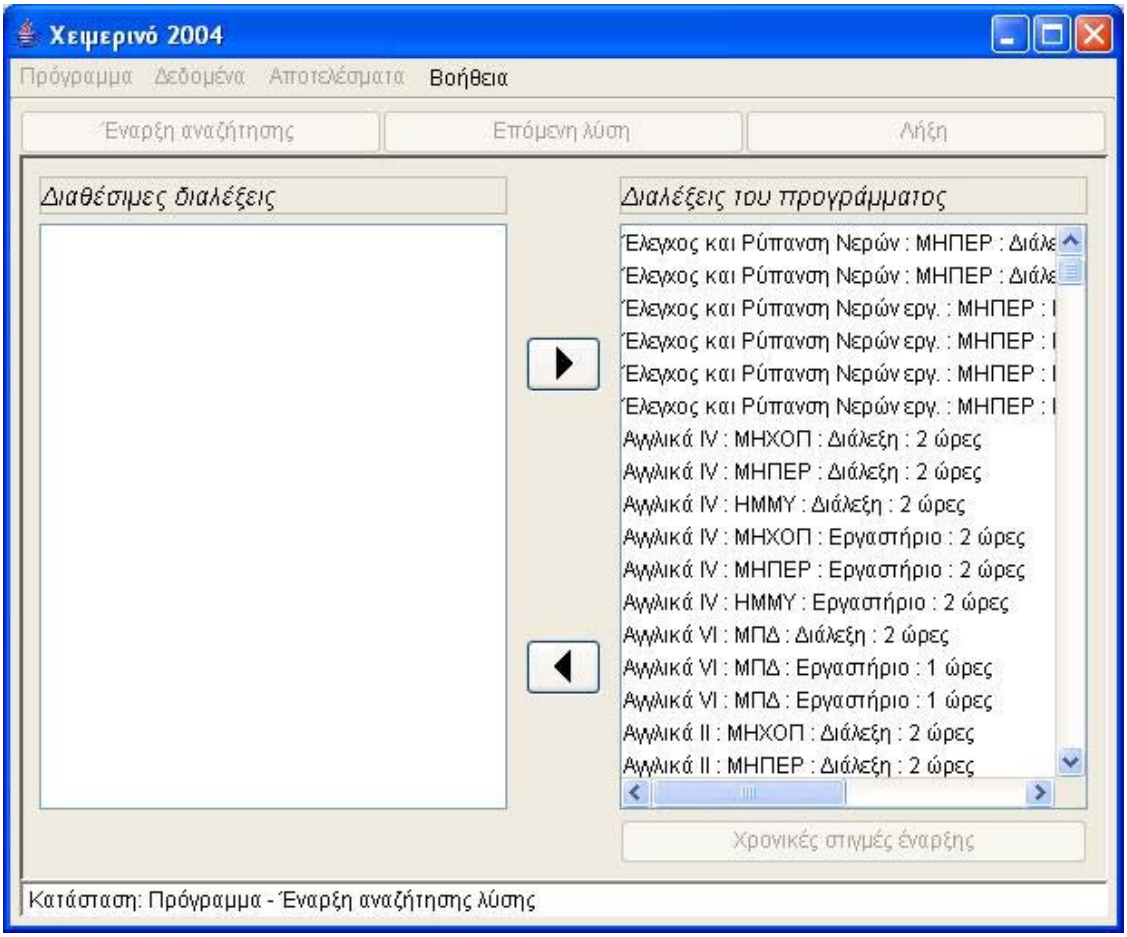
5.11 Αναζήτηση λύσης

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την αναζήτηση λύσης, αναζήτησης επόμενης καλύτερης ή ισότιμης λύσης με βάση την προηγούμενη και την λήξη της αναζήτησης ενός ωρολογίου προγράμματος. Όπως φαίνεται και στο σχήμα 5.4, η μπάρα εργαλείων (Tool bar) η οποία περιέχει τα τρία κουμπιά: "Έναρξη αναζήτησης", "Επόμενη λύση" και "Λήξη αναζήτησης".

5.11.1 Έναρξη αναζήτησης

Κατά την έναρξη αναζήτησης λύσης το σύστημα μέσω του status bar ενημερώνει τον χρήστη όπως φαίνεται στο σχήμα 5.12 για την λειτουργία που πραγματοποιείται. Κατά την διάρκεια την αναζήτησης ο χρήστης δεν μπορεί να εκτελέσει καμία άλλη λειτουργία

καθώς αυτή είναι απενεργοποιημένη. Σε περίπτωση μη εύρεσης λύσης μετά την έλευση 15’ το σύστημα ενημερώνει τον χρήστη μέσω ειδικού μηνύματος και τερματίζει την διαδικασία αναζήτησης. Στη περίπτωση που δεν υπάρχει λύση για το πρόγραμμα και αυτό μπορεί να διαπιστωθεί από το σύστημα κατά την διάρκεια της επιβολής των περιορισμών τότε ο χρήστης ενημερώνεται με κατάλληλο μήνυμα. Σε περίπτωση λανθασμένων δεδομένων ο χρήστης ειδοποιείται με μήνυμα από το σύστημα και καλείται να προβεί στις ανάλογες ενέργειες. Έστω ότι το σύστημα μας επιστρέφει λύση για το πρόγραμμα, ο χρήστης ενημερώνεται μέσω του status bar και του παρέχεται η δυνατότητα να δει τα αποτελέσματα της αναζήτησης μέσω της επιλογής “Αποτελέσματα” του μενού και να αποφασίσει εάν επιθυμεί αναζήτηση επόμενης λύσης η οποία θα είναι ισότιμη ή και καλύτερη της προηγούμενης ή εάν επιθυμεί λήξη της διαδικασίας αναζήτησης.



Σχήμα 5.12 : Το βασικό πλαίσιο εργασίας -Έναρξη αναζήτησης

5.11.2 Αναζήτηση επόμενης λύσης

Στη περίπτωση που ο χρήστης έχει επιλέξει την αναζήτηση επόμενης λύσης, τότε το σύστημα τον ενημερώνει μέσω του status bar με το αντίστοιχο μήνυμα. Σε περίπτωση μη εύρεσης λύσης μετά την έλευση 15’ το σύστημα ενημερώνει τον χρήστη μέσω ειδικού μηνύματος και τερματίζει την διαδικασία αναζήτησης. Στη περίπτωση που δεν υπάρχει

επόμενη καλύτερη λύση για το πρόγραμμα ενημερώνει τον χρήστη και τερματίζει την διαδικασία.

5.11.3 Λήξη αναζήτησης

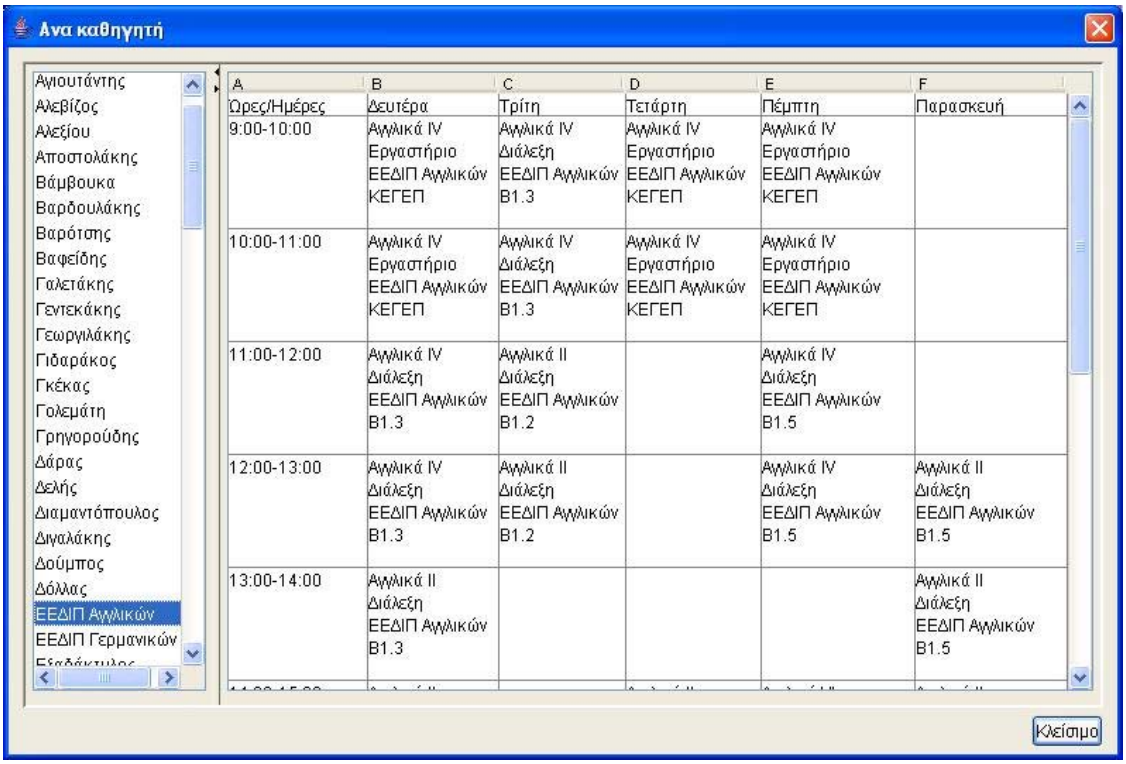
Στην περίπτωση που ο χρήστης έχει επιλέξει την λήξη αναζήτησης τότε το πρόγραμμα τερματίζει την διαδικασία αναζήτησης και θεωρεί ως λύση του συστήματος την πιο πρόσφατη λύση.

5.12 Προβολή Αποτελεσμάτων

Παρακάτω περιγράφονται οι λειτουργίες που αφορούν την προβολή των αποτελεσμάτων μίας λύσης. Στον χρήστη παρέχονται πέντε διαφορετικές περιπτώσεις προβολής των αποτελεσμάτων ανά καθηγητή, ανά αίθουσα, ανά μάθημα, ανά ομάδα μαθημάτων και τέλος ανά εξάμηνο τμήματος.

5.12.1 Ανά Καθηγητή

Επιλέγοντας μέσω του μενού την επιλογή Αποτελέσματα → Ανά καθηγητή εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.13. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλους τους καθηγητές σε αλφαβητική σειρά. Επιλέγοντας κάποιον καθηγητή εμφανίζεται στα δεξιά ο πίνακας με τα αποτελέσματα των διαλέξεων που διδάσκει ο καθηγητής και οι οποίες μετέχουν στην κατάρτιση ωρολογίου προγράμματος.

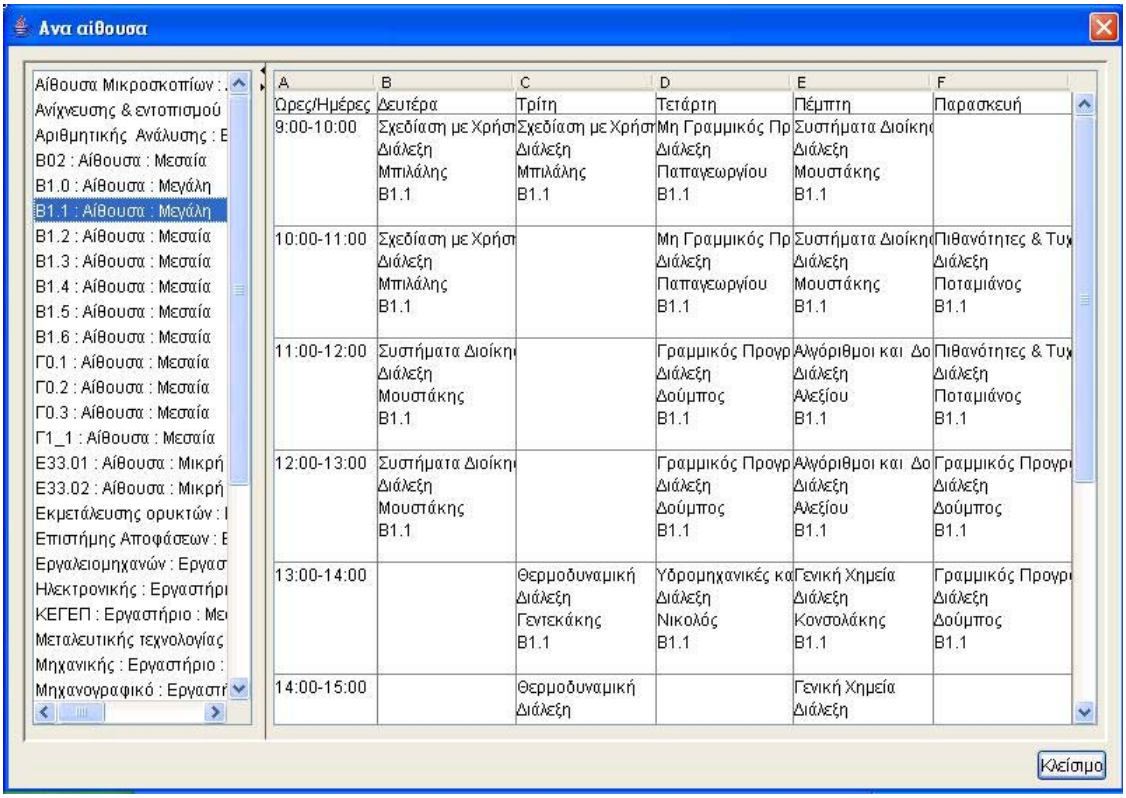


Σχήμα 5.13 : Ο διάλογος προβολής αποτελεσμάτων ανά καθηγητή

Στην περίπτωση που ο καθηγητής δεν διδάσκει μαθήματα ή οι διαλέξεις των μαθημάτων που διδάσκει δεν μετέχουν στο ωρολόγιο πρόγραμμα, τότε ο πίνακας είναι κενός.

5.12.2 Ανά Αίθουσα

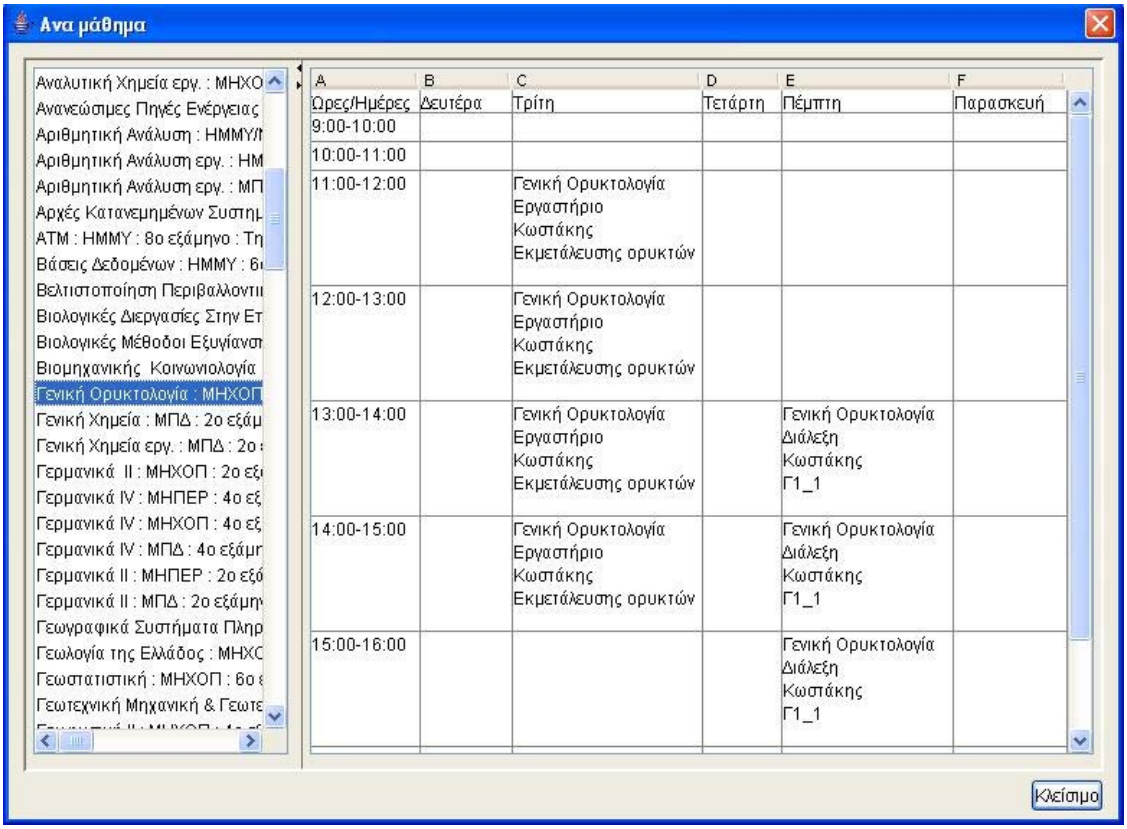
Επιλέγοντας μέσω του μενού την επιλογή Αποτελέσματα → Ανά αίθουσα εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.14. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλες τις αίθουσες σε αλφαβητική σειρά. Επιλέγοντας κάποια αίθουσα εμφανίζεται στα δεξιά ο πίνακας με τα αποτελέσματα των διαλέξεων που διδάσκονται στην αίθουσα και οι οποίες μετέχουν στην κατάρτιση ωρολογίου προγράμματος. Στην περίπτωση που στην αίθουσα δεν διδάσκεται κάποιο μάθημα που να μετέχει στο ωρολόγιο πρόγραμμα, τότε ο πίνακας είναι κενός.



Σχήμα 5.14 : Ο διάλογος προβολής αποτελεσμάτων ανά αίθουσα

5.12.3 Ανά Μάθημα

Επιλέγοντας μέσω του μενού την επιλογή Αποτελέσματα → Ανά μάθημα εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.15. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλα τα μαθήματα σε αλφαβητική σειρά. Επιλέγοντας κάποιο μάθημα εμφανίζεται στα δεξιά ο πίνακας με τα αποτελέσματα των διαλέξεων του μαθήματος, οι οποίες μετέχουν στην κατάρτιση ωρολογίου προγράμματος.

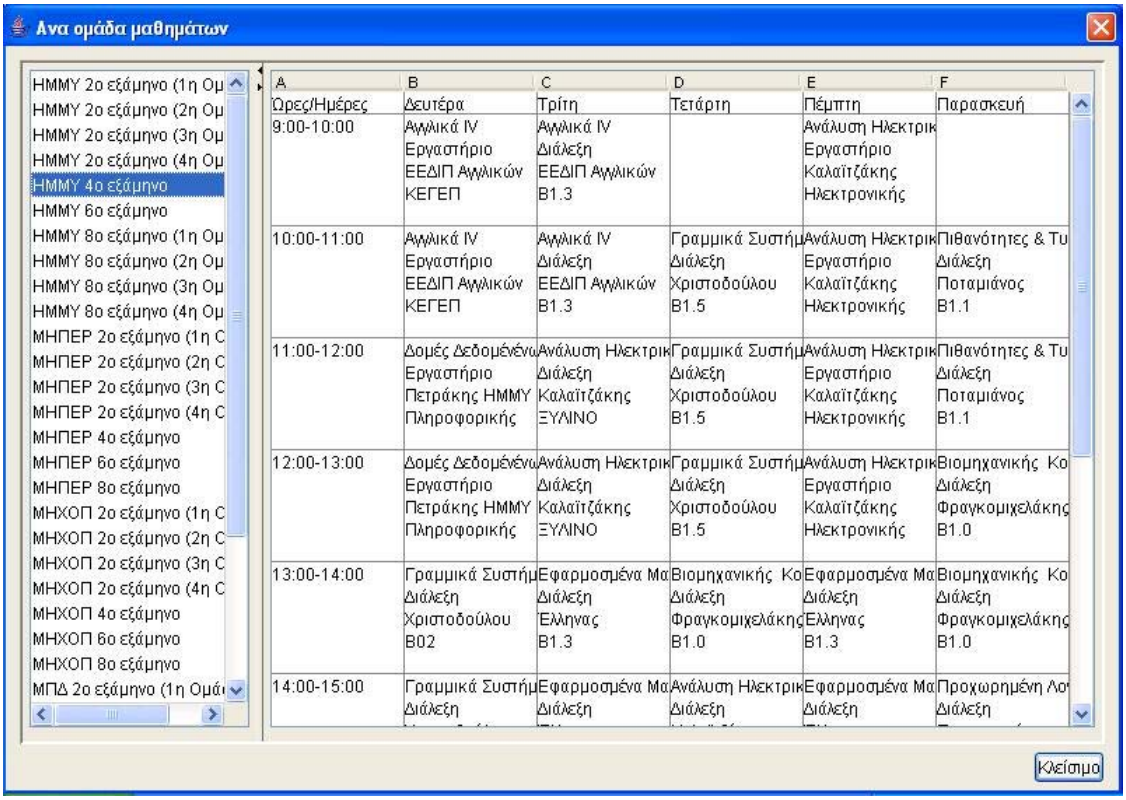


Σχήμα 5.15 : Ο διάλογος προβολής αποτελεσμάτων ανά μάθημα

5.12.4 Ανά Ομάδα Μαθημάτων

Επιλέγοντας μέσω του μενού την επιλογή Αποτελέσματα → Ανά ομάδα μαθημάτων εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.16. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλες τις ομάδες μαθημάτων σε αλφαβητική σειρά. Επιλέγοντας κάποια ομάδα εμφανίζεται στα δεξιά ο πίνακας με τα αποτελέσματα των διαλέξεων που ανήκουν στην ομάδα και οι οποίες μετέχουν στην κατάρτιση ωρολογίου προγράμματος.

Στην περίπτωση που η ομάδα μαθημάτων είναι κενή ή τα μαθήματα τα οποία συμμετέχουν στην ομάδα δεν μετέχουν στο ωρολόγιο πρόγραμμα, τότε ο πίνακας είναι κενός.

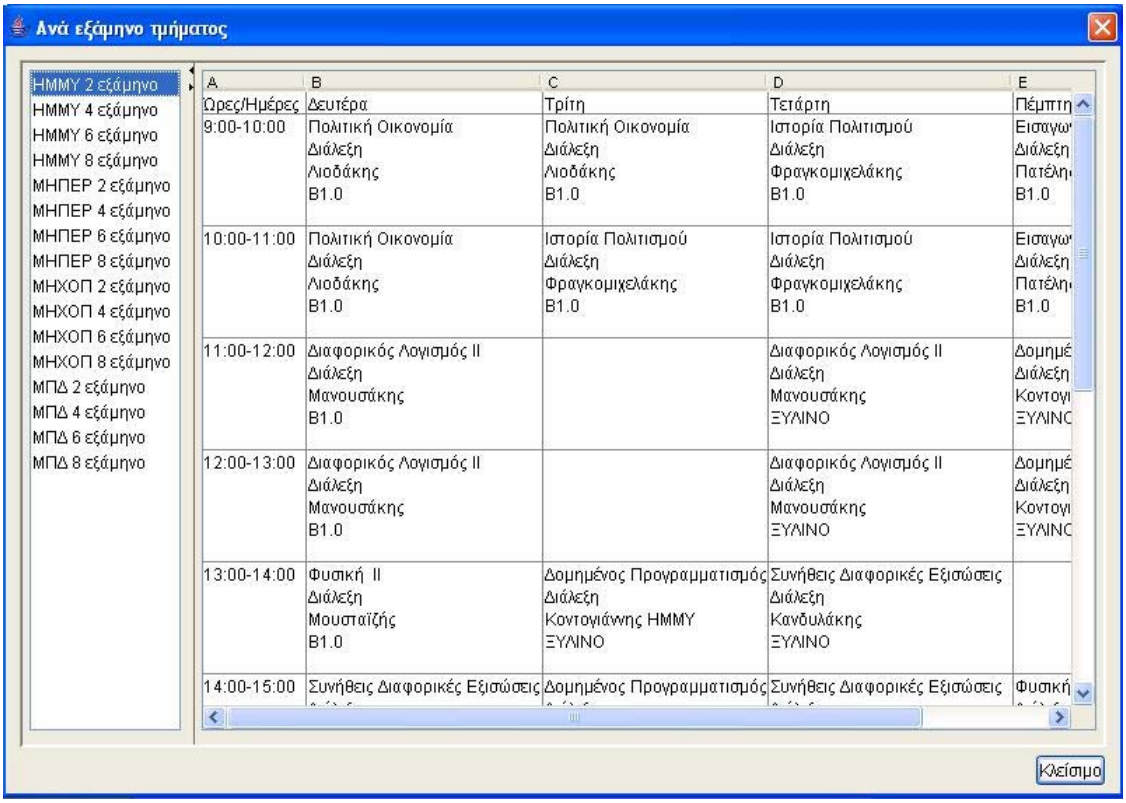


Σχήμα 5.16 : Ο διάλογος προβολής αποτελεσμάτων ανά ομάδα μαθημάτων

5.12.5 Ανά Εξάμηνο Τμήματος

Επιλέγοντας μέσω του μενού την επιλογή Αποτελέσματα → Ανά εξάμηνο τμήματος εμφανίζεται στον χρήστη ο διάλογος του σχήματος 5.17. Η βασική λίστα στα αριστερά του διαλόγου εμφανίζει όλα τα εξάμηνα των τμημάτων σε αλφαβητική σειρά με βάση το τμήμα και το εξάμηνο. Επιλέγοντας κάποιο εξάμηνο τμήματος εμφανίζεται στα δεξιά ο πίνακας με τα αποτελέσματα των διαλέξεων που ανήκουν στην εξάμηνο του τμήματος και οι οποίες μετέχουν στην κατάρτιση ωρολογίου προγράμματος.

Στην περίπτωση που το εξάμηνο του τμήματος δεν είναι της ίδιας περιόδου με αυτή ωρολογίου προγράμματος, τότε ο πίνακας είναι κενός.



Σχήμα 5.17 : Ο διάλογος προβολής αποτελεσμάτων ανά εξάμηνο τμήματος

5.13 Περίληψη

Στο κεφάλαιο αυτό παρουσιάστηκε η επικοινωνία του χρήστη με το σύστημα μέσω του γραφικού περιβάλλοντος που αναπτύξαμε. Έγινε αναφορά στη λειτουργικότητα του συστήματος καθώς επίσης στις βασικές αρχές σχεδιασμού βάση των οποίων έγινε η υλοποίηση. Το σύστημα μας απευθύνεται σε αδαείς αλλά και έμπειρους χρήστες καθώς έγινε προσπάθεια το γραφικό περιβάλλον που παρέχεται να είναι απλό και λειτουργικό βασιζόμενο στα πρότυπα άλλων ευρέως χρησιμοποιούμενων εφαρμογών ώστε να είναι όσο το δυνατόν πιο οικεία. Παρουσιάζονται επίσης τα πιο πιθανά σενάρια που μπορεί να υπάρξουν κατά την χρήση του συστήματος.

Κεφάλαιο 6

Ανακεφαλαίωση και μελλοντικές επεκτάσεις.

6.1 Συμπεράσματα – Ανακεφαλαίωση

Στην παρούσα διπλωματική διατριβή, αναπτύχθηκε ένα σύστημα κατάρτισης ωρολογίου προγράμματος. Ο σχεδιασμός του μοντέλου πραγματοποιήθηκε με βάση τη θεωρία των προβλημάτων ικανοποίησης περιορισμών, δίνοντας έτσι στον χρήστη την δυνατότητα να εκφράσει τους περιορισμούς που ισχύουν στο εκάστοτε πρόβλημα σύμφωνα με τις προτιμήσεις του. Η αυτόματη κατάρτιση ωρολογίου προγράμματος αποτελεί ένα πολύ δύσκολο πρόβλημα καθώς ανήκει στις τάξεις των NP-complete προβλημάτων όπου ο χρόνος αναζήτησης λύσης αυξάνει εκθετικά σε σχέση με τον αριθμό των δεδομένων.

Το βασικό πλαίσιο εργασίας μας ήταν το σύστημα της ECL^iPS^e, το οποίο αποτελεί μία γλώσσα λογικού προγραμματισμού με περιορισμούς. Είναι ένα σύστημα βασιζόμενο στην Prolog με κάποιες επιπλέον ανεπτυγμένες βιβλιοθήκες. Τα κυριότερα οφέλη που αποκομίσαμε από την χρήση του συστήματος της ECL^iPS^e είναι ότι μας δόθηκε η δυνατότητα να επικεντρωθούμε περισσότερο στην ανάπτυξη της εφαρμογή μας καθώς το σύστημα της ECL^iPS^e μας παρείχε έναν ισχυρό εσωτερικό μηχανισμό υπολογισμού (evaluation). Επιπλέον μας παρείχε κάποιες επιπλέον βιβλιοθήκες με υλοποιημένες τεχνικές γενικευμένης συνέπειας, οι οποίες έχουν αναπτυχθεί ειδικά για προβλήματα χρονικού προγραμματισμού όπως αυτό της κατάρτισης ωρολογίου προγράμματος.

Όπως έχουμε αναφέρει είναι πολύ σημαντικό τόσο το μοντέλο που θα αναπτυχθεί για το πρόβλημα όσο και οι αλγόριθμοι που χρησιμοποιούνται στην αποδοτικότητα του συστήματος. Ο συνδυασμός των αλγορίθμων της διάδοσης περιορισμών και των τεχνικών συνέπειας καθώς και οι μέθοδοι που επιλέγονται στην υλοποίηση των περιορισμών μπορεί να μειώσουν δραστικά το χρόνο αναζήτησης καθώς ο σωστός συνδυασμός των παραπάνω μπορεί να περιορίσει το χώρο αναζήτησης του προβλήματος. Ο αλγόριθμος αναζήτησης παίζει εξίσου σημαντικό ρόλο όπως και οι ευριστικές που μπορεί να χρησιμοποιηθούν έτσι ώστε το σύστημα να προσεγγίζει ευκολότερα την λύση, στην περίπτωση που αυτή υπάρχει.

Ασχοληθήκαμε με προβλήματα ικανοποίησης περιορισμών ενώ μελετήθηκαν τα είδη περιορισμών που υπάρχουν καθώς επίσης και αλγόριθμοι διάδοσης περιορισμών και των τεχνικών συνέπειας. Είδαμε πως η μέθοδος της συνέπειας ορίων υπερτερεί των άλλων μεθόδων σε περιπτώσεις γραμμικών περιορισμών για μεταβλητές πεπερασμένων πεδίων και επιπλέον πως μπορεί να αυξηθεί η απόδοση του συστήματος χρησιμοποιώντας μεθόδους γενικευμένης συνέπειας, κάνοντας χρήση "σύνθετων" περιορισμών όπως του cumulative, alldifferent και άλλων, σε ειδικές περιπτώσεις. Μελετήθηκαν οι ευριστικές διάταξης μεταβλητών και τιμών που μπορούν να χρησιμοποιηθούν σε αλγορίθμους *κατά βάθος* αναζήτησης λύσης ή ακόμα και συνδυασμό των παραπάνω.

Στη πορεία παρουσιάστηκε το μοντέλο του προβλήματος βασιζόμενο στα προβλήματα ικανοποίησης περιορισμών για πεπερασμένα πεδία καθώς επίσης και των εύκαμπτων και αυστηρών περιορισμών διέπουν το πρόβλημά μας. Η ποιότητα του ωρολογίου

προγράμματος μας καθορίστηκε βάση τριών κριτηρίων τα οποία συνδυάστηκαν γραμμικά σε σχέση με κάποια καθορισμένα βάρη, παράγοντας την αντικειμενική συνάρτηση, η οποία πρέπει να έχει όσο το δυνατόν μικρότερη τιμή.

Στη συνέχεια παρουσιάσαμε την αναπαράσταση των δεδομένων του προβλήματος όπως επίσης και την υλοποίηση των περιορισμών του. Κάναμε σύγκριση των μεθόδων υλοποίησης των περιορισμών, με βάση του πια μέθοδος παράγει το μικρότερο δέντρο παραγωγής της ECL^{i}PS^e.

Τέλος παρουσιάστηκε η επικοινωνία του χρήστη με το σύστημα μέσω του γραφικού περιβάλλοντος που αναπτύξαμε. Έγινε αναφορά στη λειτουργικότητα του συστήματος καθώς επίσης στις βασικές αρχές σχεδιασμού βάση των οποίων έγινε η υλοποίηση. Το σύστημα μας απευθύνεται σε αδαείς αλλά και έμπειρους χρήστες καθώς έγινε προσπάθεια το γραφικό περιβάλλον που παρέχεται να είναι απλό. Παρουσιάζονται επίσης τα πιο πιθανά σενάρια που μπορεί να υπάρξουν κατά την χρήση του συστήματος.

Η υλοποίηση των παραπάνω, έγινε σε ECL^{i}PS^e, MySQL και Java.

6.2 Μελλοντικές επεκτάσεις

Η παρούσα διατριβή μπορεί να επεκταθεί αξιοποιώντας την ήδη υπάρχουσα εργασία.

6.2.1 Επεκτάσεις αναφορικά με τις μεθόδους αναζήτησης λύσης

Όσο αφορά τις μεθόδους αναζήτησης, θα μπορούσε να προστεθεί και να αναπτυχθεί ένας επιπλέον αλγόριθμος αναζήτησης ο οποίος να επιτρέπει την εύρεση λύση με βάση κάποια είδη υπάρχουσα. Πιο αναλυτικά, στην περίπτωση που ο χρήστης δεν είναι απόλυτα ικανοποιημένος με την λύση που του πρόσφερε το σύστημα και θα επιθυμούσε να πραγματοποιήσει κάποιες αλλαγές, τότε θα μπορούσε το σύστημα να του προσφέρει αυτή την δυνατότητα, επιστρέφοντάς του μία εφικτή λύση η οποία θα περιλαμβάνει τις μικρότερες αλλαγές με βάση την προηγούμενη. Ουσιαστικά αναφερόμαστε στην απαίτηση που μπορεί να προβάλει ο χρήστης για επαναπρογραμματισμό του ωρολογίου προγράμματος.

Επιπλέον, η προεπεξεργασία των δεδομένων από τον χρήστη έτσι ώστε το σύστημα να του επιστρέφει μία εφικτή λύση, είναι πολύ δύσκολη, ειδικά στις περιπτώσεις μεγάλου αριθμού δεδομένων. Στην περίπτωση τώρα που για το πρόβλημα δεν υπάρχει εφικτή λύση, ο χρήστης δυσκολεύεται να κατανοήσει τους λόγους για τους οποίους το σύστημα δεν μπορεί να την βρει. Για τον λόγο αυτό, θα μπορούσε να αναπτυχθεί ένας μηχανισμός ο οποίος να εξάγει συμπεράσματα και να παρέχει στον χρήστη υποστήριξη σχετικά με τα δεδομένα και τους περιορισμούς που τους έχουν επιβληθεί, έτσι ώστε ο χρήστης στην πορεία να πραγματοποιήσει τις κατάλληλες αλλαγές.

6.2.2 Επεκτάσεις αναφορικά με το Γραφικό περιβάλλον

Όσο αφορά το γραφικό περιβάλλον, κρίνεται απαραίτητο να γίνουν δοκιμές ευχρηστίας (Usability Test). Με αυτές τις δοκιμές θα βρεθούν λάθη ή παραλήψεις που δυσχεραίνουν τη δουλεία των χρηστών, ενώ θα μπορούσαν να γίνουν και προτάσεις για καινούργια εργαλεία ή δυνατότητες που θα κάνουν ακόμα πιο εύκολη την πλοήγηση της εφαρμογής. Επιπρόσθετα θα μπορούσαν να γίνουν επεκτάσεις με βάση τα νέα εργαλεία και λειτουργίες που θα προστεθούν στο σύστημα.

Βιβλιογραφία

[1] Kyriakos Zervoudakis and Panagiotis Stamatopoulos. A Generic Object-Oriented Constraint-Based Model for University Course Timetabling. *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling PATAT 2000, LNCS 2079, pp. 28-47, Konstanz, 2001.*

[2] Panagiotis Stamatopoulos,Efstratios Viglas and Serafeim Karaboyas. Nearly Optimun Timetable Construction Through CLP and Intelligent Search. *International Journal on Artificial Intelligence Tools Vol. 7, No. 4 (1998) 415-442.*

[3] Harikleia Frangouli, Vassilis Harmadas and Panagiotis Stamatopoulos. UTSE : Construction of Optimum Timetables for University Courses – A CLP Based Approach. *Proceedings of the 3rd International Conference on the Practical Applications of Prolog PAP '95, pp. 225-243, Paris, 1995.*

[4] George M. White and Junhan Zhang. Generating Complete University Timetables by Combining Tabu Search with Constraint Logic. *Lecture Notes in Computer Science, vol.1408, pp.187-198, (1998).*

[5] Martin Henz and Jörg Würtz. Using Oz for College Timetabling. *Proceedings of the 1995 International Conference on the Practice and Theory of Atomated Timetabling, Edinburgh, Scotland, 20 August-1 September.*

[6] Christelle Guéret, Narendra Jussien, Patrice Boizumault and Christian Prins. Building University timetables using Constraint Logic Programming. *The Practice and Theory of Automated Timetabling, edited by Edmund Burke and Peter Ross. Springer-Verlag, pages 130–145, (1996).*

[7] Hans-Joachim Goltz. Combined Automatic and Interactive Timetabling Using Constraint Logic Programming. *PATAT 2000,Konstanz, August 2000, pp.78-95.*

[8] Hans-Joachim Goltz, Georg Küchler and Dirk Matzle. Constraint-Based Timetabling for Universities. *INAP'98, 11th International Conference on Applications of Prolog, Tokyo, September 1998, pp. 75-80.*

[9] Hans-Joachim Goltz. On Methods of Constraint-Based Timetabling. *PACLP'2000, Manchester, April 2000, pp. 167-177.*

[10] Hans-Joachim Goltz. Reducing Domains for Search in CLP(FD) and Its Application to Job-Shop Scheduling. *Proceedings of International Conference on Principles and Practice of Constraint Programming, Cassis, France, September 1995, Springer, LNCS 976, pp. 549-562.*

[11] Hans-Joachim Goltz and Ulrich John. Methods for Solving Practical Problems of Jop-Shop Scheduling Modeled in CPL(FD). *In Proceedings of PACT'96, Conference on Practical Application of Constraint Technology, London, April 1996.*

[12] P.Boizumault, Y. Delon and L.Peridy. Constraint Logic Programming for Examination Timetabling. *Journal of Logic Programming*, 1995, 1-17.

[13] Dons Banks, Peter Van Beek and Amnon Meisels. A Heuristic Incremental Modeling Approach to Course Timetabling. *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence*, *Vancouver, British Columbia, 16-29, June, 1998.*

[14] F.P.M. Dignum, W.P.M. Nuijten and L.M.A. Janssen. Solving a Time Tabling Problem by Constraint Satisfaction. *Technical report, Eindhoven University of Technology, 1995.*

[15] Peter Wilke, Matthias Gröbner and Norbert Oster. A Hybrid Genetic Algorithm for School Timetabling. *Published in "AI 2002: Advances in Artificial Intelligence", AI 02, December 2002, Springer Lecture Notes in Artificial Intelligence, Vol. 2557, B. McKay and J. Slaney (Eds.), pp.455-464.*

[16] Slim Abdennadher and Michael Marte. University Timetabling using Constraint Handling Rules. *In Journal of Applied Artifiacial Intelligence, Special Issue on Constraint Handling Rules, 14(4):311-326, 2000.*

[17] Luìs Paulo Reis and Eugénio Oliveira. A Language for Specifying Complete Timetabling Problems. *In PATAT2000 - Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling*, *pp.456-471, Konstanz, Germany, August, 2000.*

[18] Maria Kambi and David Gilbert. Timetabling in Constraint Logic Programming. *In Proceedings of INAP-96: Symposium and Exhibition on Industrial Applications of Prolog, pages 79--88, Tokyo, Japan, Oct 1996.*

[19] Claude Le Pape. Three mechanisms for managing resource constraints in a library for constraint-based schedule. *In Proceedings of the INRIA/IEEE Conference on Emerging Technologies and Factory Automation, Paris, France, 1995.*

[20] Rina Dechter and Itay Meiri. Experimental Evaluation of Preprocessing Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence, Volume 68 , Issue 2 (August 1994), Pages: 211 – 241,Year of Publication: 1994, ISSN:0004-3702 .*

[21] Kish Shen, Joachim Schimpf and Stefano Novello. A High-Level Generic Interface to External Programming Language for ECLiPSe. *Published in Practical Aspects of Declarative Languages, 4th International Symposium PADL 2002, LNCS 2257, Springer-Verlag.*

[22] Mark Wallace, Stefano Novello and Joachim Schimpf. ECLiPSe: A Platform for Constraint Logic Programming. *Technical report, August 1997.*

[23] Micha Meier. Event Handling in Prolog. *Technical report ECRC-ECRC-95-09.*

[24] IC-Park Home Page. http://www.icparc.ic.ac.uk.

[25] ECLiPSe 5.6: *User Manual, ECRC (2003).*

[26] Stuart Russell and Peter Norvig. Artificial Intelligence A Modern Approach. *Second edition, Section 5, pages 138-159, Prentice Hall Series in Artificial Intelligence.*

[27] Ivan Bratko. Prolog, Programming for Artificial Intelligence. *Third edition, Addison Wesley.*

[28] Kim Marriott and Peter J. Stuckey. Programming with Constraints, An Introduction. *The MIT Press, Cambridge, Massachusetts, London, England (1998).*

[29] Deborah J. Mahew - Principles and Guidelines in Software User Interface Design, *Prentice Hall PTR-Englewood, New Jersey 07632, 1992.*