Ατομική Διπλωματική Εργασία

Αυτόματη Κατασκευή Προγράμματος Μαθημάτων

Μάριος Βασιλάκης

Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Μάρτιος 2012

Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα μηχανικών Πληροφορικής και Τηλεπικοινωνιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αυτόματη Κατασκευή Προγράμματος Μαθημάτων

Μάριος Βασιλάκης

ΕΠΙΒΛΕΠΩΝ:
Κώστας Στεργίου

Η Ατομική αυτή Διπλωματική Εργασία υποβλήθηκε για την εκπλήρωση των απαιτήσεων απόκτησης του διπλώματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

ΚΟΖΑΝΗ (Μάρτιος, 2012)

Περίληψη

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.

Στόχος της διπλωματικής είναι η μοντελοποίηση του προβλήματος κατασκευής προγραμμάτων μαθημάτων ως προβλήματος ικανοποίησης περιορισμών και η επίλυση του χρησιμοποιώντας τεχνικές τοπικής αναζήτησης. 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. Η μέθοδος επίλυσης που θα χρησιμοποιηθεί είναι αυτή του ευρετικού μηχανισμού των ελαχίστων συγκρούσεων (minimum conflicts) και θα ελεγχθεί η αποδοτικότητά του. Για να επιτευχθεί αυτό θα ελεγχθεί ο αλγόριθμος των ελαχίστων συγκρούσεων και δύο παραλλαγές του, με δύο διαφορετικές αρχικοποιήσεις, σε τρία προβλήματα με διαφορετική δυσκολία. Επίσης θα δοθούν κάποια κριτήρια αξιολόγησης και των προβλημάτων, ώστε να γίνεται κατανοητή η δυσκολία τους, αλλά και της απόδοσης των αλγορίθμων για κάθε αποτέλεσμα εκτέλεσης του πάνω σε κάθε πρόβλημα. Έτσι θα δοθεί μια ξεκάθαρη εικόνα για το πόσο αποδοτική είναι η κάθε μέθοδος και σε ποια περίπτωση.

Για την υλοποίηση του προγράμματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού c++. Από τα εκτεταμένα αποτελέσματα και συμπεράσματα που λήφθηκαν δίνεται μια γενικότερη εικόνα για την αποδοτικότητα των αλγορίθμων και γίνεται κατανοητό γιατί θα μπορούσαν να είναι χρήσιμοι σε κάποια μελλοντική υλοποίηση μιας γραφική διεπαφής χρήστη.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Κώστα Στεργίου για την υποστήριξη και καθοδήγηση που μου πρόσφερε κατά την διάρκεια της υλοποίησης αυτής της εργασίας, αλλά και για την επίτευξη μιας άψογης συνεργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων – Πινάκων

Κεφάλαιο 1 : Εισαγωγή

Στην καθημερινή μας ζωή είναι πολύ συχνή η ανάγκη για την παραγωγή ωρολογίων προγραμμάτων για διάφορα ιδρύματα, εκπαιδευτικούς οργανισμούς, μέσα μαζικής μεταφοράς, εταιρίες κ.α. Ένα από τα πιο συνηθισμένα προβλήματα που αντιμετωπίζουν οι άνθρωποι είναι αυτό του χρονοπρογραμματισμού. Ακόμα και στην καθημερινή τους ζωή οι άνθρωποι καλούνται να επιλύσουν τέτοια προβλήματα. Για παράδειγμα ένα φοιτητής ο οποίος προετοιμάζεται για μία εξέταση πρέπει να οργανώσει ένα πρόγραμμα μελέτης, όπου αυτό το πρόγραμμα δεν είναι τίποτε άλλο από ένα χρονοπρογραμματισμό διαδικασιών όπως είναι η μάθηση, η εξάσκηση και η επανάληψη. Γενικότερα κάθε πρόβλημα το οποίο έχει στόχο του να ταξινομήσει ένα σύνολο δραστηριοτήτων σε ένα πρόγραμμα λέγεται πρόβλημα χρονοπρογραμματισμού (scheduling problem).

Ένα είδος του χρονοπρογραμματισμού είναι η κατασκευή ωρολογίων προγραμμάτων. Τέτοια προβλήματα αντιμετωπίζονται από διάφορα ιδρύματα, εκπαιδευτικούς οργανισμούς, μέσα μαζικής μεταφοράς, εταιρίες κ.α. Σε πολλές από αυτές τις περιπτώσεις η δημιουργία του ωρολογίου προγράμματος αναλαμβάνεται από άτομα, τα οποία πρέπει να επιλύσουν αυτό το δύσκολο πρόβλημα. Ο λόγος που αυτού του είδους τα προβλήματα είναι δύσκολα, αλλά και χρονοβόρα λόγω των τεχνικών με τις οποίες ακόμα και σήμερα λύνονται, είναι το μεγάλο πλήθος αλληλεπιδρώντων παραγόντων και περιορισμών.

Τα χαρακτηριστικά που κάνουν την κατασκευή του ωρολογίου προγράμματος δύσκολη, κάνουν εύκολη σε αντίθεση την κατηγοριοποίηση του εντάσσοντας τα στην κατηγορία των Προβλημάτων Ικανοποίησης Περιορισμών (Constraint Satisfaction Problems – CSP). Πιο συγκεκριμένα η κατασκευή ωρολογίου προγράμματος για το ακαδημαϊκό εξάμηνο ενός πανεπιστημίου γίνεται ιδιαίτερα περίπλοκη από ένα μεγάλο σύνολο περιορισμών, όπως είναι η διαθεσιμότητα των αιθουσών, η αλληλοεπικάλυψη μαθημάτων, το πλήθος των μαθημάτων ανά ημέρα κ.α. Αυτού του είδους οι περιορισμοί ονομάζονται γενικοί καθώς είναι κοινοί για κάθε πανεπιστήμιο.

Υπάρχουν επίσης οι ειδικοί περιορισμοί οι οποίοι έχουν να κάνουν με το συγκεκριμένο πανεπιστήμιο όπως είναι οι προτιμήσεις των καθηγητών για την μέρα που θα διδάξουν. Επιπλέον το γεγονός πως σε ένα ακαδημαϊκό πρόγραμμα υπάρχει μεγαλύτερη επιρροή της ανθρώπινης κρίσης αντίθετα με το σχολικό πρόγραμμα, επειδή στο ακαδημαϊκό πρόγραμμα δεν είναι υποχρεωτικό να γίνεται κάθε ώρα μάθημα αντίθετα με το σχολικό, με αποτέλεσμα οι ειδικοί περιορισμοί να έχουν μεγάλο μέρος στο πρόγραμμα.

Στον τομέα της Τεχνητής Νοημοσύνης έχουν γίνει πολλές έρευνες πάνω στην επίλυση προβλημάτων ικανοποίησης περιορισμών κυρίως με χρήση αλγορίθμων τοπικής αναζήτησης, γενετικών αλγορίθμων, προσομοιωμένης ανόπτησης κ.α. Αντίστοιχα και στην κατασκευή ωρολογίου προγράμματος έχουν γίνει εφαρμογές των αλγορίθμων με πολύ καλά αποτελέσματα. Στην συγκεκριμένη εργασία θα γίνει χρήση τοπικής αναζήτησης και πιο συγκεκριμένα του ευρετικού μηχανισμού των ελάχιστων συγκρούσεων (min-conflicts heuristic).

Για να βρεθεί μια τελική λύση στο πρόβλημα του ωρολογίου προγράμματος πρέπει να ικανοποιηθούν όλοι οι περιορισμοί, να μην παραβιάζεται κανένας. Μια υπολογιστική λύση όμως μπορεί να μην είναι απόλυτα λειτουργική, δηλαδή να ικανοποιεί όλους τους περιορισμούς αλλά να μην είναι και ποιοτική (π.χ. μια άνιση

ημερήσια κατανομή των μαθημάτων) κάνοντας το ωρολόγιο πρόγραμμα μη λειτουργικό.

Επίσης η πολυπλοκότητα του προβλήματος μπορεί να οδηγήσει σε αδυναμία επίλυσης. Λόγω όλων των περιορισμών και του μεγάλου χώρου αναζήτησης είναι πιθανό να μην υπάρχει λύση, δηλαδή να μην είναι δυνατόν να ικανοποιηθούν όλοι οι περιορισμοί, κυρίως οι ειδικοί (π.χ. πολλοί καθηγητές προτιμούν την ίδια μέρα).

Το μεγαλύτερο πλεονέκτημα της επίλυσης αυτού του προβλήματος με χρήση αλγορίθμων τεχνητής νοημοσύνης είναι η μεγάλη ταχύτητα επίλυσης. Ακόμα και όταν η τελική λύση δεν είναι απόλυτα λειτουργική μπορεί με κάποιες απλές αλλαγές από την ανθρώπινη πλευρά να φτάσει στην ποιοτική λύση η οποία επιδιώκεται.

Κεφάλαιο 2 : Χρονοπρογραμματισμός (Timetabling)

2.1 Τι είναι ο χρονοπρογραμματισμός

«Ο χρονοπρογραμματισμός είναι η κατανομή δεδομένων γεγονότων σε υποδοχείς χρόνου και χώρου, υπό ορισμένους περιορισμούς, με τέτοιο τρόπο ώστε να ικανοποιηθούν όσο το δυνατόν περισσότεροι στόχοι».

Μερικά παραδείγματα προβλημάτων χρονοπρογραμματισμού είναι ο εκπαιδευτικός προγραμματισμός (χρονοδιάγραμμα διαλέξεων ή εξετάσεων), ο προγραμματισμός στο χώρο εργασίας, ο προγραμματισμός αθλητικών γεγονότων, ο προγραμματισμός της λειτουργίας των μεταφορικών μέσων και ούτω καθεξής.

Τα προβλήματα χρονοπρογραμματισμού ορίζουν μια κλάση προβλημάτων βελτιστοποίησης με περιορισμούς, που είναι δύσκολο να λυθούν. Τέτοια προβλήματα είναι κυρίως ταξινομημένα ως προβλήματα ικανοποίησης περιορισμού, όπου ο κύριος σκοπός είναι να ικανοποιηθούν όλοι οι περιορισμοί του προβλήματος, παρά η βελτιστοποίηση των στόχων.

Αυτή τη στιγμή, η επιστήμη δεν έχει καμία αναλυτική μέθοδο για την επίλυση όλων των περιπτώσεων προβλημάτων αυτής της κατηγορίας, εκτός από την εξαντλητική αναζήτηση, που δεν μπορεί να εφαρμοστεί παρά μόνο στα προβλήματα προσομοιώσεις, λόγω των εξαιρετικά μεγάλων διαστημάτων αναζήτησης των περιπτώσεων των πραγματικών προβλημάτων.

Ο αυτοματοποιημένος προγραμματισμός, από την άλλη μεριά, είναι ένα ζήτημα μεγάλου ενδιαφέροντος δεδομένου ότι μπορεί να εξοικονομήσει πολλή ανθρώπινη εργασία, σε οργανισμούς και επιχειρήσεις, και να παρέχει βέλτιστες λύσεις στην ικανοποίηση περιορισμού εντός λεπτών, τα οποία μπορούν να ωθήσουν την παραγωγικότητα, την ποιότητα της εκπαίδευσης, την ποιότητα της εξυπηρέτησης και τελικά την ποιότητα ζωής.

Εντούτοις, τα μεγάλης κλίμακας χρονοδιαγράμματα, όπως τα πανεπιστημιακά χρονοδιαγράμματα, μπορεί να χρειαστούν πολύ μεγάλη προσπάθεια και πάρα πολλές ώρες εργασίας, από ένα καταρτισμένο πρόσωπο ή μια ομάδα, προκειμένου να παραχθούν υψηλής ποιότητας χρονοδιαγράμματα που θα παρέχουν τη βέλτιστη ικανοποίηση των περιορισμών και τη βελτιστοποίηση των στόχων του χρονοδιαγράμματος συγχρόνως.

2.2 Μέθοδοι που εφαρμόζονται στο χρονοπρογραμματισμό

Ένας μεγάλος αριθμός διαφορετικών μεθόδων έχει προταθεί ήδη στη λογοτεχνία για την επίλυση των προβλημάτων προγραμματισμού. Αυτές οι μέθοδοι προέρχονται από διάφορους επιστημονικούς κλάδους όπως την Έρευνα Διαδικασιών (Operations Research), την Τεχνητή Νοημοσύνη (Artificial Intelligence), και την Υπολογιστική Νοημοσύνη (Computational Intelligence) και μπορούν να διαιρεθούν σε τέσσερις κατηγορίες:

1) Διαδοχικές μέθοδοι που χειρίζονται τα προβλήματα χρονοπρογραμματισμού ως προβλήματα γραφημάτων. Γενικά, διατάσουν τα γεγονότα χρησιμοποιώντας ευρετικές μεθόδους

(heuristics) και ορίζουν έπειτα τα γεγονότα διαδοχικά στους έγκυρους υποδοχείς χρόνου κατά τέτοιο τρόπο ώστε να μην παραβιάζεται κανένας περιορισμός για κάθε υποδοχέα.

2) Μέθοδοι συστάδων, στις οποίες το πρόβλημα διαιρείται σε ένα πλήθος από σύνολα γεγονότων. Κάθε σύνολο ορίζεται έτσι ώστε να ικανοποιεί όλους τους ισχυρούς περιορισμούς και στη συνέχεια, τα σύνολα ορίζονται σε υποδοχείς πραγματικού χρόνου για να ικανοποιήσουν και τους χαλαρούς περιορισμούς.

3) Μέθοδοι που βασίζονται σε περιορισμούς, σύμφωνα με τους οποίους ένα πρόβλημα χρονοπρογραμματισμού παίρνει τη μορφή ενός συνόλου μεταβλητών (γεγονότα) στο οποίο οι τιμές (π.χ. καθηγητές, αίθουσες) πρέπει να οριστούν προκειμένου να ικανοποιήσουν ένα πλήθος από περιορισμούς.

4) Meta-heuristic μέθοδοι, όπως οι Γενετικοί Αλγόριθμοι, Simulated Annealing, Tabu Search, και άλλες ευρετικές (heuristic) προσεγγίσεις, που εμπνέονται συνήθως από τη φύση, και εφαρμόζουν διαδικασίες που μιμούνται τη φύση στις λύσεις ή τους πληθυσμούς των λύσεων, προκειμένου αυτοί να εξελιχθούν προς τη βελτιστοποίηση.

2.3 Η προσέγγισή μας

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.

Σε αυτή την εργασία επικεντρωνόμαστε στην επίλυση πανεπιστημιακών προβλημάτων χρονοπρογραμματισμού με τη χρήση των Αλγορίθμων Τοπικής Αναζήτησης, δηλαδή με το χρονοπρογραμματισμό διεξαγωγής διαλέξεων.

Κεφάλαιο 3: Προβλήματα Ικανοποίησης Περιορισμών (Constraint Satisfaction Problems, CSP)

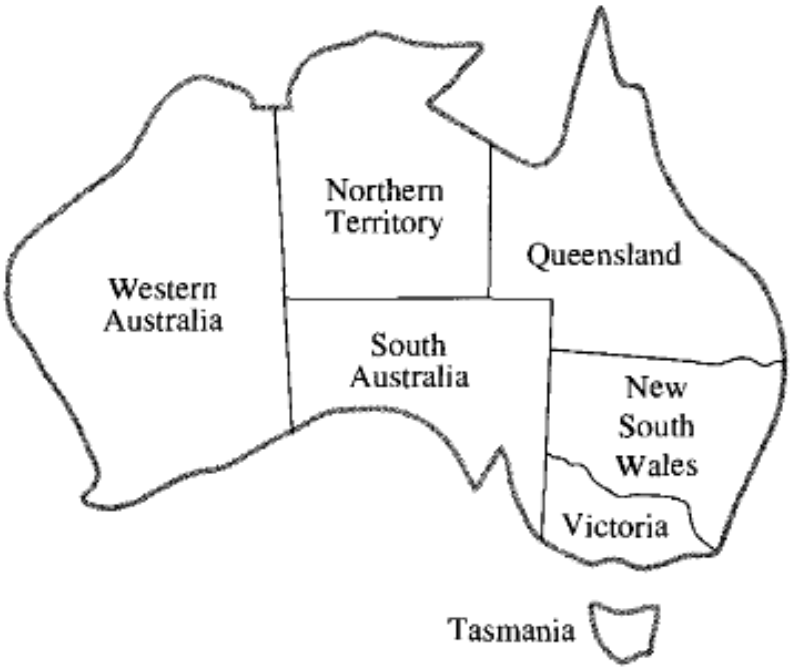
Η παρουσίαση και τα παραδείγματα αυτού του κεφαλαίου βασίζονται στο κεφάλαιο 5 του βιβλίου Τεχνητή Νοημοσύνη Μία Σύγχρονη Προσέγγιση (Artificial Intelligence A Modern Approach) των Stuart Russell και Peter Norvig.

3.1 Ορισμός

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". Ένα πρόβλημα ικανοποίηση περιορισμών (constraint satisfaction problem,CSP) ορίζεται από ένα σύνολο μεταβλητών, X_1, X_2, …, X_n, και από ένα σύνολο περιορισμών, C_1, C_2, …, C_m. Κάθε μεταβλητή X_i έχει ένα κενό πεδίο (domain)Di των δυνατών τιμών της. Κάθε περιορισμός C_i αναφέρεται σε κάποιο υποσύνολο των μεταβλητών και καθορίζει τους επιτρεπτούς συνδυασμούς τιμών γι' αυτό το υποσύνολο. Μια κατάσταση του προβλήματος ορίζεται με ανάθεση τιμών σε μερικές ή όλες τις μεταβλητές, $\{X_i = v_i, X_j = v_j, \ldots\}$. Μια ανάθεση τιμής που δεν παραβιάζει κανέναν περιορισμό ονομάζεται συνεπής(consistent) ή νόμιμη ανάθεση. Πλήρης ανάθεση είναι μια ανάθεση που περιλαμβάνει όλες τις μεταβλητές, και λύση ενός προβλήματος ικανοποίησης περιορισμών είναι μια πλήρης ανάθεση τιμών που ικανοποιεί όλους τους περιορισμούς. Μερικά προβλήματα ικανοποίησης περιορισμών απαιτούν επίσης μια λύση που μεγιστοποιεί μία αντικειμενική συνάρτηση(objective function).

3.2 Παραδείγματα Προβλημάτων Ικανοποίησης Περιορισμών

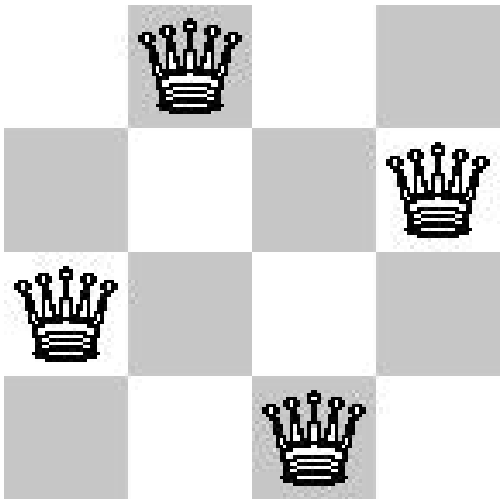
Το πρόβλημα του χρωματισμού χάρτη (map coloring problem) αποτελεί ένα από τα αρχέτυπα CSP προβλήματα. Το πρόβλημα αποτελείται από τον χρωματισμό διαφορετικών περιοχών ενός προκαθορισμένου χάρτη, με περιορισμένο αριθμό χρωμάτων, και υπόκειται στην περιοριστική συνθήκη ότι δύο παρακείμενες περιοχές δεν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



Σχήμα 3.1 : Το πρόβλημα χρωματισμού χάρτη

Ο παρακάτω περιορισμός εγγυάται ότι κάθε ζεύγος παρακείμενων περιοχών δεν μπορεί να έχει το ίδιο χρώμα: WA ≠ NT ∧ WA ≠ SA ∧ NT ≠ SA ∧ NT ≠ Q ∧ SA ≠ Q ∧ SA ≠ NSW ∧ SA ≠ V ∧ Q ≠ NSW ∧ NSW ≠ V

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 4ων βασιλισσών όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 3.2 : Το πρόβλημα των 4ων βασιλισσών

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

$R1 \neq R2 \wedge R1 \neq R3 \wedge R1 \neq R4 \wedge R2 \neq R3 \wedge R2 \neq R4 \wedge R3 \neq R4$,

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

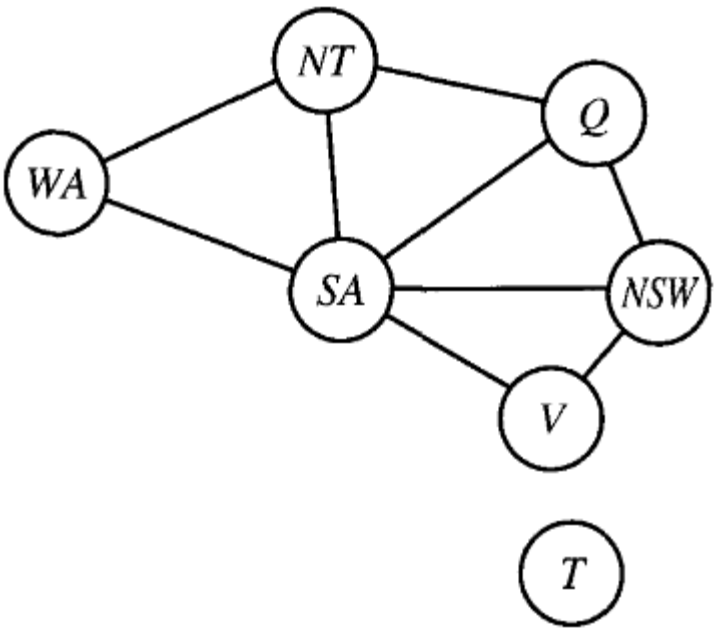
$C1 \neq C2 \wedge C1 \neq C3 \wedge C1 \neq C4 \wedge C2 \neq C3 \wedge C2 \neq C4 \wedge C3 \neq C4$, εξασφαλίζει ότι δύο βασίλισσες δεν μπορούν να βρίσκονται στην ίδια στήλη, και τέλος οι περιορισμοί:

$C1 - R1 \neq C2 - R2 \wedge C1 - R1 \neq C3 - R3 \wedge C1 - R1 \neq C4 - R4 \wedge C2 - R2 \neq C3 - R3 \wedge C2 - R2 \neq C4 - R4 \wedge C3 - R3 \neq C4 - R4$

και

$C1 + R1 \neq C2 + R2 \wedge C1 + R1 \neq C3 + R3 \wedge C1 + R1 \neq C4 + R4 \wedge C2 + R2 \neq C3 + R3 \wedge C2 + R2 \neq C4 + R4 \wedge C3 + R3 \neq C4 + R4$, εξασφαλίζουν ότι δύο βασίλισσες δεν μπορούν να βρίσκονται στην ίδια διαγώνιο.

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.



Σχήμα 3.3 : Αναπαράσταση περιορισμών με γράφο

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.

3.3 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.

3.4 Περιορισμοί

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

3.4.1 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.

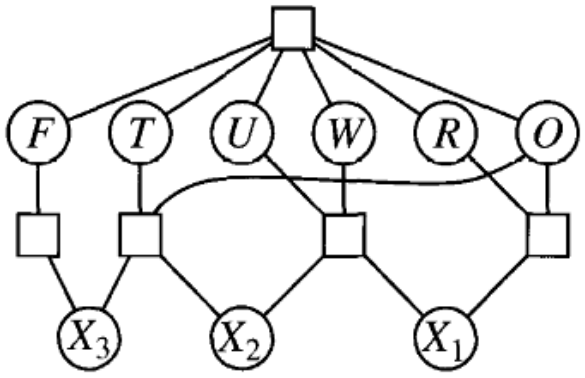
3.4.2 Δυαδικός Περιορισμός (Binary Constraint)

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 είδαμε προηγουμένως.

3.4.3 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.

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



Σχήμα 3.4 : 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:

$$O + O = R + 10 \cdot X1$$
$$X1 + W + W = U + 10 \cdot X2$$
$$X2 + T + T = O + 10 \cdot X3$$

X3 = F

όπου οι μεταβλητές X1, X2 και X3 είναι βοηθητικές μεταβλητές και αναπαριστούν το κρατούμενο 0 ή 1. Οι Υψηλού βαθμού περιορισμοί μπορούν να απεικονιστούν από ένα υπέρ-γράφο περιορισμών (constraint hypergraph) όπως αυτός του σχήματος, στον οποίο κάθε περιορισμός απεικονίζεται με ένα τετράγωνο και οι ακμές οι οποίες συνδέουν τον περιορισμό με τις συσχετιζόμενες μεταβλητές.

3.4.4 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 ή και τοπικής αναζήτησης. Οι Εύκαμπτοι περιορισμοί μπορούν να εκφραστούν χρησιμοποιώντας κάποια βάρη σε σχέση με τους αντίστοιχους περιορισμούς. Κάθε φορά που κάποιος περιορισμός παραβιάζεται, η αντίστοιχη τιμή βάρους προστίθεται στη συνάρτηση κόστους, βάση τις οποίας επιλέγεται ή απορρίπτεται αντίστοιχα μία λύση. Για παράδειγμα ας θεωρήσουμε το παρόν 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.

3.5 Τοπική Αναζήτηση στα Προβλήματα Ικανοποίησης Περιορισμών

Οι αλγόριθμοι τοπική αναζήτησης αποδεικνύονται πολύ αποτελεσματικοί για την επίλυση πολλών προβλημάτων ικανοποίησης περιορισμών. Χρησιμοποιούν μία διατύπωση με πλήρεις καταστάσεις. Η αρχική κατάσταση αναθέτει τιμή σε κάθε μεταβλητή και η συνάρτηση διαδοχών συνήθως αναλαμβάνει να αλλάζει την τιμή μίας-μίας μεταβλητής. Στο πρόβλημα των βασιλισσών, εικόνα **Νούμερο,** η αρχική κατάσταση θα μπορούσε να είναι μια τυχαία διάταξη των 4 βασιλισσών σε 4 στήλες και η συνάρτηση διαδοχών παίρνει μία βασίλισσα και εξετάζει κινήσεις της σε άλλες θέσεις της στήλης.

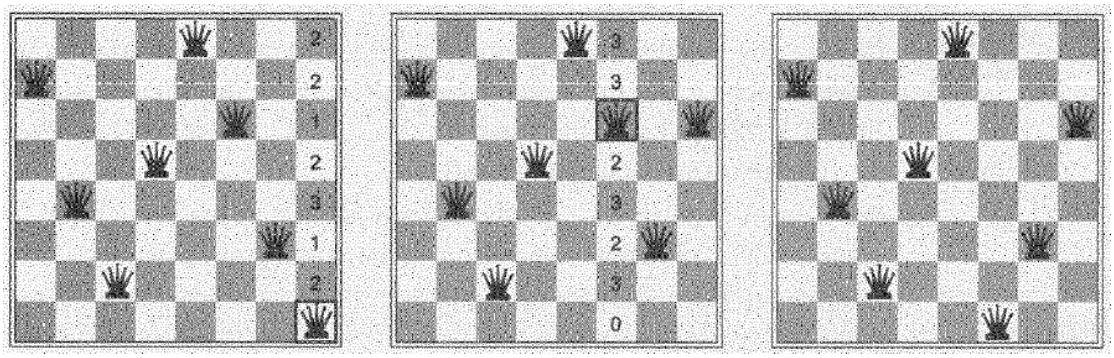
Για την επιλογή νέας τιμής για μια μεταβλητή, ο πιο συνήθης ευρετικός μηχανισμός είναι να επιλέγεται η τιμή που προκύπτει με τον ελάχιστο αριθμό συγκρούσεων με άλλες μεταβλητές, δηλαδή τον ευρετικό μηχανισμό των **ελαχίστων συγκρούσεων**(**min-conflicts**). Ο ψευδοκώδικας του αλγορίθμου παρουσιάζεται στην συνέχεια.

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for csp
  for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen, conflicted variable from VARIABLES[csp]
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
  return failure
```

Ο αλγόριθμος των ελαχίστων συγκρούσεων για την επίλυση προβλημάτων ικανοποίησης περιορισμών με τοπική αναζήτηση. Η αρχική κατάσταση μπορεί να επιλέγεται τυχαία ή με μία διαδικασία άπληστης ανάθεσης τιμών που επιλέγει μια τιμή ελαχίστων συγκρούσεων για την κάθε μεταβλητή με τη σειρά. Η συνάρτηση CONFLICTS καταμετρά τον αριθμό των περιορισμών που παραβιάζονται από μια συγκεκριμένη τιμή, με διαδεδομένη την υπόλοιπη τρέχουσα ανάθεση τιμών.

Ακολουθεί ένα παράδειγμα για την λύση ενός προβλήματος 8 βασιλισσών σε δύο κινήσεις χρησιμοποιώντας την μέθοδο των ελαχίστων συγκρούσεων.



Σχήμα 3.5 : Λύση δύο βημάτων για ένα πρόβλημα 8 βασιλισσών που χρησιμοποιεί τη μέθοδο των ελαχίστων συγκρούσεων.

Σε κάθε στάδιο επιλέγεται μία βασίλισσα για νέα ανάθεση τιμής πάνω στη στήλη της. Ο αριθμός των συγκρούσεων (στη συγκεκριμένη περίπτωση ο αριθμός των βασιλισσών που απειλούνται) εμφανίζεται ε κάθε τετράγωνο. Ο αλγόριθμος μετακινεί τη βασίλισσα στο τετράγωνο των ελαχίστων συγκρούσεων, αίροντας τις ισοδυναμίες τυχαία.

Η μέθοδος των ελαχίστων συγκρούσεων είναι εκπληκτικά αποτελεσματική για πολλά προβλήματα ικανοποίησης περιορισμών, ιδιαίτερα όταν του δοθεί μια λογική αρχική κατάσταση. Μία από τις εντυπωσιακές ιδιότητες του αλγορίθμου είναι πως στο πρόβλημα των n-βασιλισσών ο χρόνος εκτέλεσης της μεθόδου των ελαχίστων συγκρούσεων είναι σχεδόν ανεξάρτητος από το μέγεθος του προβλήματος.

Άλλο ένα πλεονέκτημα της τοπικής αναζήτησης είναι ότι μπορεί να χρησιμοποιείται σε online περιβάλλοντα όπου το πρόβλημα αλλάζει. Αυτό είναι ιδιαίτερα σημαντικό στα προβλήματα χρονοπρογραμματισμού. Για παράδειγμα ένα εβδομαδιαίος χρονοπρογραμματισμός αεροπορικών πτήσεων μπορεί να περιλαμβάνει χιλιάδες πτήσεις και δεκάδες χιλιάδες τοποθετήσεις προσωπικού, αλλά η κακοκαιρία σε ένα αεροδρόμιο μπορεί να κάνει τον χρονοπρογραμματισμό ανέφικτο, για αυτό θα θέλαμε να το χρονοπρογραμματισμό με έναν ελάχιστο αριθμό αλλαγών. Αυτό μπορεί να γίνει εύκολα με έναν αλγόριθμο τοπικής αναζήτησης που ξεκινά από τον τρέχοντα χρονοπρογραμματισμό.

Κεφάλαιο 4 : Βάση Δεδομένων - Περιορισμοί

4.1 Εισαγωγή

Ένα πρόβλημα κατασκευής ωρολογίου προγράμματος χαρακτηρίζεται από μεγάλη πολυπλοκότητα και επηρεάζεται από ένα πλήθος παραμέτρων. Εάν λάβουμε υπόψη μας το πλήθος των αιθουσών, το μέσο όρο των ωρών διδασκαλίας ανά ημέρα για τις πέντε ημέρες τις εβδομάδας και το πλήθος το μαθημάτων επί τον μέσο όρο των διδακτικών ωρών που απαιτούν, τότε τα πιθανά ωρολόγια προγράμματα είναι υπερβολικά πολλά. Χρησιμοποιώντας ένα σύνολο περιορισμών μειώνει κατά πολύ το μέγεθος του προβλήματος, αλλά αυξάνει την πολυπλοκότητά του και τον κίνδυνο να μη μπορεί να βρεθεί λύση. Σε κάθε περίπτωση, το πρόγραμμα πρέπει να είναι ευέλικτο, παραμετρικό και εύκολα τροποποιήσιμο, ώστε να μπορεί να αντιμετωπίζει τις όποιες απαιτήσεις.

4.2 Δεδομένα

Ημέρες: Οι ημέρες θα είναι πέντε, από Δευτέρα μέχρι Παρασκευή, με το Σαββατοκύριακο ελεύθερο. Το πλήθος των ημερών δεν θα είναι μεταβλητό, καθώς δεν πρόκειται να αλλάξει στο μέλλον (όπως για παράδειγμα στο πρόγραμμα ενός φροντιστηρίου).

Διδακτικές περίοδοι: Οι διδακτικές περίοδοι αφορούν τις ώρες στις οποίες θα γίνονται τα μαθήματα. Μία από τις αρχικές υποχωρήσεις που έγιναν στο πρόγραμμα είναι πως ένα μάθημα πρέπει να είναι δίωρο ή τρίωρο, άρα θεωρώντας πως το Πανεπιστήμιο έχει ώρες λειτουργίας από τις 9:00 έως τις 20:00 τότε θα έχουμε δύο σύνολα από πιθανές επιλογές για τα μαθήματα τα οποία είναι:

- Τα δίωρα = {9:00-11:00, 10:00-12:00,…,18:00-20:00}.
- Τα τρίωρα = {9:00-12:00,10:00-13:00,…,17:00-20:00}.

Μαθήματα: Το κυριότερο στοιχείο του προγράμματος είναι το "Μάθημα" το οποίο πρέπει να τοποθετήσουμε κατάλληλα μέσα στο ωρολόγιο πρόγραμμα. Τα μαθήματα θα περιέχουν ένα σύνολο από χαρακτηριστικά τα οποία θα είναι ο καθηγητής που θα κάνει το μάθημα, το εξάμηνο το οποίο αφορά το μάθημα, εάν είναι θεωρητικό ή εργαστηριακό και τα δίωρα και τρίωρα στα οποία χωρίζεται το μάθημα.

Καθηγητές: Οι καθηγητές που θα διδάσκουν τα μαθήματα Κάθε ένας τους έχει χαρακτηριστικά όπως το όνομά τους και τα μαθήματα για τα οποία είναι υπεύθυνοι.

Εξάμηνα: Τα εξάμηνο στα οποία χωρίζονται οι φοιτητές. Τα χαρακτηριστικά που περιέχει αφορούν τις κατευθύνσεις που μπορούν να επιλέξουν οι φοιτητές (Α', Β', ή Γ') από το τέταρτο έτος (έβδομο εξάμηνο) και μετά. Στο πρόγραμμα επιλέξαμε να μην γίνεται έλεγχος του πλήθους των ατόμων που ανήκουν σε κάθε εξάμηνο.

Αίθουσες: Οι αίθουσες στις οποίες θα γίνονται τα μαθήματα. Χωρίζονται σε αίθουσες θεωρίας και εργαστηρίου. Το πλήθος των αιθουσών θα είναι μεταβλητό. Επίσης πρόγραμμα επιλέξαμε το μέγεθος των αιθουσών να μην είναι γνωστό.

4.3 Περιορισμοί.

Όπως είπαμε νωρίτερα το πρόβλημα αυτό ανήκει στην κατηγορία των Προβλημάτων Ικανοποίησης Περιορισμών. Δηλαδή περιγράφεται από ένα σύνολο περιορισμών οι οποίοι πρέπει να ικανοποιηθούν ώστε να λυθεί το πρόβλημα. Αυτοί είναι:

- Ένας καθηγητής να μην διδάσκει δύο μαθήματα στην ίδια χρονική περίοδο.
- Να μη διδάσκονται δύο διαφορετικά μαθήματα του ίδιου εξαμήνου στην ίδια χρονική περίοδο..
- Να μη γίνονται δύο διαφορετικά μαθήματα στην ίδια αίθουσα στην ίδια χρονική περίοδο.
- Αν το ίδιο μάθημα έχει περισσότερα από ένα δίωρα ή τρίωρα τότε αυτά να διδάσκονται σε διαφορετικές μέρες. Δηλαδή εάν ένα μάθημα έχει ένα δίωρο και ένα τρίωρο τότε αυτά να μην διδάσκονται την ίδια μέρα.
- Οι καθηγητές να μην ξεπερνούν ένα όριο μαθημάτων ανά μέρα ώστε να μην έχουν επιβαρυμένο πρόγραμμα. Για παράδειγμα εάν αυτό το όριο είναι δύο, τότε να μην έχει περισσότερες από δύο δίωρες οι τρίωρες παραδώσεις μαθημάτων.
- Τα εξάμηνα να μην έχουν περισσότερα από ένα όριο μαθημάτων ανά μέρα ώστε να μην έχουν επιβαρυμένο πρόγραμμα. Για παράδειγμα εάν αυτό το όριο είναι τέσσερα, τότε να μην πρέπει να παρακολουθήσουν περισσότερα από τέσσερα δίωρα οι τρίωρα μαθήματα.

Για την επίλυση του προβλήματος και την ικανοποίηση όλων των περιορισμών θα χρησιμοποιήσουμε τον αλγόριθμο ελαχίστων συγκρούσεων (min-conflicts). Προφανώς οι υποψήφιες λύσεις θα πρέπει να ικανοποιούν όλους τους προαναφερθέντες περιορισμούς, αλλά από όλες αυτές θα δεχτούμε την καλύτερη. Το κριτήριο για την καλύτερη λύση θα είναι να έχουν μοιραστεί τα μαθήματα όσων το δυνατών καλύτερα μέσα στο ωρολόγιο πρόγραμμα.

4.4 Μοντελοποίηση του Προβλήματος

4.4.1 Δεδομένα στο Πρόγραμμα

Το κυριότερο στοιχείο του προγράμματος είναι το "Μάθημα" το οποίο πρέπει να τοποθετήσουμε κατάλληλα μέσα στο ωρολόγιο πρόγραμμα. Μέσα στο πρόγραμμα το μάθημα αντιστοιχεί στην κλάση **Subject** (Subject στα αγγλικά σημαίνει Μάθημα). Για την ακρίβεια ένα στιγμιότυπο της κλάσης Subject δεν αντιπροσωπεύει ένα μάθημα, αλλά τις διαλέξεις στις οποίες είναι χωρισμένο το μάθημα. Για παράδειγμα αν ένα μάθημα χρειάζεται να έχει τέσσερις ώρες διδασκαλίας μέσα στην εβδομάδα τότε θα πρέπει να χωριστεί σε δύο δίωρα και να τοποθετηθεί μέσα στο Ω.Π(ωρολόγιο πρόγραμμα), οπότε θα έχουμε δύο στιγμιότυπα της κλάσης Subject που θα αντιστοιχούν στις δύο δίωρες διαλέξεις του μαθήματος.

Η κλάση **Subject** έχει κάποιες ιδιότητες οι οποίες αφορούν τα απαραίτητα χαρακτηριστικά του κάθε μαθήματος, και είναι:

Sub_id: Είναι τύπου integer και αφορά το μάθημα στο οποίο ανήκει αυτή η διδασκαλία. Για παράδειγμα εάν ένα μάθημα απαιτεί πέντε ώρες θεωρίας τότε αυτές θα χωριστούν σε ένα δίωρο και ένα τρίωρο, δηλαδή δύο στιγμιότυπα της κλάσης Subject στο πρόγραμμα. Αυτή η ιδιότητα θα μας βοηθήσει στο να μην γίνονται οι διδασκαλίες του ίδιου μαθήματος στην ίδια ημέρα.

Prof: Είναι τύπου integer και αφορά τον καθηγητή που παραδίδει το μάθημα (από το αγγλικό Professor που σημαίνει καθηγητής). Ο λόγος για τον οποίο δεν είναι τύπου char ή string είναι για μεγαλύτερη ταχύτητα στην εκτέλεση του προγράμματος, ώστε να αποφευχθεί η επαναλαμβανόμενη σύγκριση μεταξύ λέξεων. Δηλαδή δίνεται σε κάθε καθηγητή ένας αριθμός ώστε να γίνονται οι συγκρίσεις γρηγορότερα, ενώ η αντιστοιχία των ονομάτων των καθηγητών με τους αριθμούς τους διατηρείται σε ένα αρχείο ώστε στην τελική εμφάνιση του προγράμματος να εμφανίζονται με τα ονόματά τους.

Sem: Είναι τύπου integer και αφορά το εξάμηνο στο οποίο διδάσκεται το μάθημα (από το αγγλικό Semester που σημαίνει εξάμηνο). Αυτή η ιδιότητα είναι απαραίτητη για τους περιορισμούς που αφορούν τα εξάμηνα.

Kind: Είναι τύπου integer και αφορά το είδος του μαθήματος, δηλαδή εάν είναι θεωρητικό μάθημα ή εργαστηριακό (kind στα αγγλικά σημαίνει είδος). Θα παίρνει δύο τιμές που θα είναι 0 εάν είναι θεωρητικό μάθημα και 1 εάν είναι εργαστηριακό. Με αυτόν τον τρόπο μπορούμε να ξεχωρίσουμε εάν το μάθημα πρέπει να γίνει σε αίθουσα εργαστηρίου ή θεωρίας.

Hours: Είναι τύπου integer και αφορά την διάρκεια της διδασκαλίας, δηλαδή εάν το μάθημα θα διαρκεί δύο ή τρείς ώρες (hours στα αγγλικά σημαίνει ώρες). Θα παίρνει δύο τιμές που θα είναι 0 εάν είναι δίωρο μάθημα και 1 εάν είναι τρίωρο.

Η κλάση Subject περιέχει μόνο αυτές τις ιδιότητες γιατί αυτά είναι τα χαρακτηριστικά του μαθήματος τα οποία θα είναι σταθερά κατά την εκτέλεση του προγράμματος. Τα χαρακτηριστικά του μαθήματος τα οποία μεταβάλλονται κατά την εκτέλεση του προγράμματος τοποθετήθηκαν σε άλλη κλάση.

Η κλάση που περιέχει τα μη σταθερά χαρακτηριστικά του μαθήματος είναι ή **Info** (σύντομο για Information το οποίο σημαίνει πληροφορίες). Αντίστοιχα με την Subject και σε αυτήν την κλάση πρέπει να έχουμε τόσα στιγμιότυπα όσες είναι και οι περίοδοι διδασκαλίας του κάθε μαθήματος, δηλαδή εάν ένα μάθημα έχει χωρίζεται σε δύο δίωρα τότε θα έχουμε δύο στιγμιότυπα της κλάσης για το ίδιο μάθημα.

Η κλάση **Info** έχει κάποιες ιδιότητες οι οποίες, όπως είπαμε, αφορούν τα μη σταθερά χαρακτηριστικά του κάθε μαθήματος, και είναι:

Day: Είναι τύπου integer και αφορά την ημέρα στην οποία γίνεται το μάθημα (day στα αγγλικά σημαίνει ημέρα). Κατά την εκτέλεση του προγράμματος μπορεί να πάρει τιμές από το 1 έως το 5, που αντιστοιχούν στις πέντε μέρες τις εβδομάδας Δευτέρα έως Παρασκευή.

Hour: Είναι τύπου integer και αφορά την ώρα στην οποία γίνεται το κάθε μάθημα (hour στα αγγλικά σημαίνει ώρα). Κάθε πιθανό δίωρο και τρίωρο μέσα στην μέρα θα αντιστοιχεί σε έναν αριθμό. Το πλήθος των τιμών που μπορεί να πάρει αυτή η μεταβλητή κατά την εκτέλεση του προγράμματος μπορεί αν διαφέρουν ανάλογα με τις ώρες λειτουργίας του κάθε Πανεπιστημίου. Για παράδειγμα, εάν οι ώρες λειτουργίας είναι από 09:00 έως 20:00, τότε όπως είδαμε πριν τα σύνολα των δίωρων και των τρίωρων θα είναι:

- Τα δίωρα = {9:00-11:00, 10:00-12:00,…,18:00-20:00}.
- Τα τρίωρα = {9:00-12:00,10:00-13:00,…,17:00-20:00}.

Οπότε οι αντιστοίχηση σε αριθμούς ώστε να μπορεί να γίνεται εύκολα έλεγχος μέσα στο πρόγραμμα περιγράφεται στον παρακάτω πίνακα:

Δίωρα	Αριθμοί στο πρόγραμμα	Τρίωρα	Αριθμοί στο πρόγραμμα
09:00-11:00	1	09:00-12:00	2
10:00-12:00	3	10:00-13:00	4
11:00-13:00	5	11:00-14:00	6
12:00-14:00	7	12:00-15:00	8
13:00-15:00	9	13:00-16:00	10
14:00-16:00	11	14:00-17:00	12
15:00-17:00	13	15:00-18:00	14
16:00-18:00	15	16:00-19:00	16
17:00-19:00	17	17:00-20:00	18
18:00-20:00	19		

Πίνακας 4.1 : Αντιστοίχηση των αριθμών των χρονικών περιόδων με τους αριθμούς του προγράμματος.

Στον πίνακα βλέπουμε πως υπάρχουν συνολικά 19 δίωρα και τρίωρα, και πως τα δίωρα είναι μονοί αριθμοί ενώ τα τρίωρα είναι ζυγοί. Αυτό θα δούμε πως είναι πολύ χρήσιμο στην ανάθεση των δίωρων και των τρίωρων στα μαθήματα, αλλά και στον έλεγχο ικανοποίησης των περιορισμών.

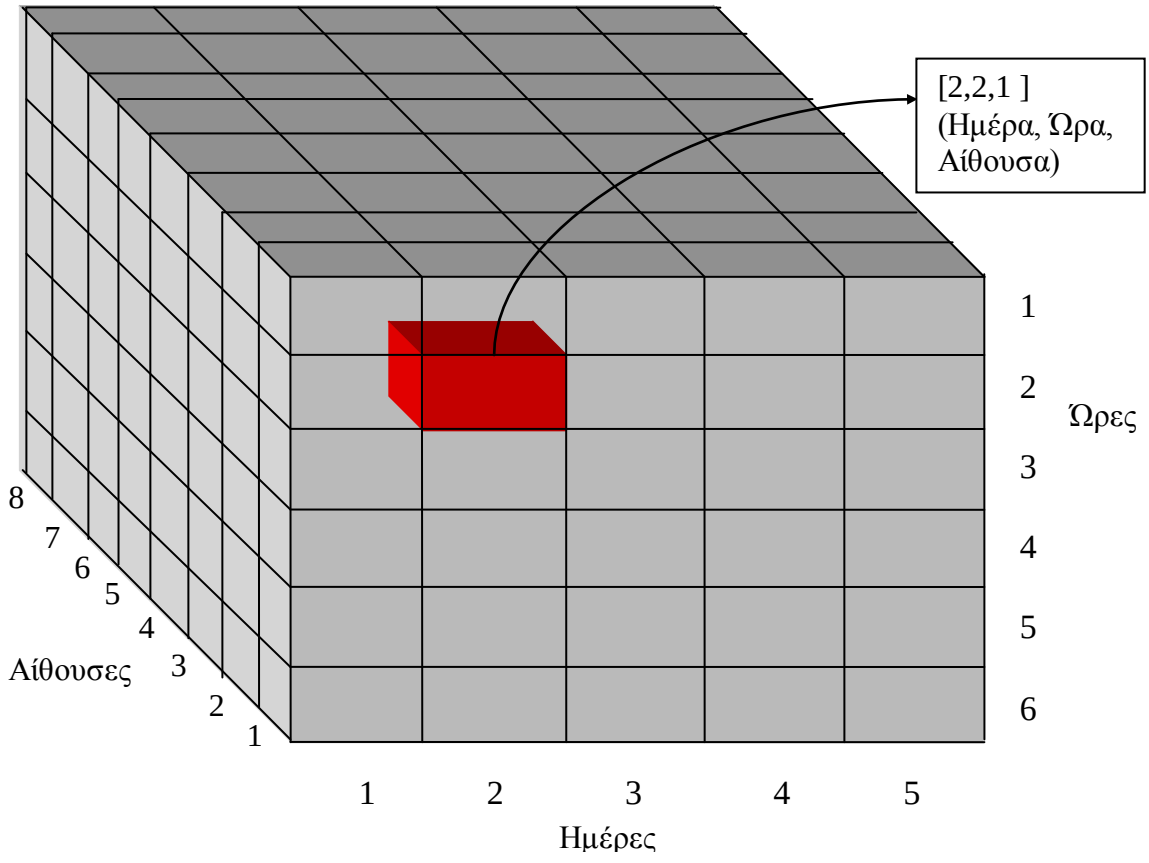
Room: Είναι τύπου integer και αφορά την αίθουσα στην οποία θα γίνει το μάθημα (room στα αγγλικά σημαίνει αίθουσα). Σε αυτή την μεταβλητή δίνεται ένας αριθμός ανάλογα με το αν το μάθημα είναι θεωρητικό ή εργαστηριακό. Για παράδειγμα εάν στο κτήριο του Πανεπιστημίου υπάρχουν έξι αίθουσες για θεωρία και τέσσερα εργαστήρια τότε η μεταβλητή room θα μπορεί να πάρει τιμές από ένα έως δέκα, με τις πρώτες έξι να αντιστοιχούν σε αίθουσες θεωρίας και τις τέσσερις τελευταίες σε εργαστήρια.

Ο λόγος που τα χαρακτηριστικά του μαθήματος χωρίστηκαν σε δύο κλάσεις, εκτός από το ότι αυτά της Subject είναι σταθερά ενώ αυτά της Info δεν είναι, γίνεται πιο κατανοητό εάν δούμε την βάση δεδομένων πάνω στην οποία βασίστηκαν οι κλάσεις Subject και Info.

4.4.2 Ωρολόγιο Πρόγραμμα

Εάν υποθέσουμε πως έχουμε έναν τρισδιάστατο πίνακα στον οποίο οι γραμμές αντιστοιχούν στις μέρες της εβδομάδας, οι στήλες είναι τα δίωρα και τα τρίωρα στα οποία χωρίζονται οι ώρες λειτουργίας του Πανεπιστημίου και η τρίτη διάσταση είναι οι συνολικές αίθουσες του Πανεπιστημίου. Δηλαδή κάθε ιδιότητα της κλάσης Info να αντιστοιχεί σε μία διάσταση.

Για να γίνει πιο κατανοητό μπορούμε να δούμε μία γραφική αναπαράσταση αυτού του πίνακα με διαστάσεις αυτές που πήραμε σαν παραδείγματα πριν. Δηλαδή πέντε στήλες για τις μέρες, 19 γραμμές για τα δίωρα και τρίωρα στα οποία χωρίζονται οι ώρες λειτουργίας του Πανεπιστημίου εάν είναι από 09:00 έως 20:00, και άλλες 10 θέσεις κατά βάθος (τρίτη διάσταση) εάν έχω έξι αίθουσες θεωρίας και τέσσερα εργαστήρια.



Σχήμα 4.1 : Γραφική απεικόνιση του τρισδιάστατου πίνακα του προγράμματος.

Κάθε θέση του πίνακα αντιστοιχεί σε μία πιθανή θέση για ένα μάθημα, δηλαδή εάν ένα μάθημα βρίσκεται στην θέση [2,13,6] τότε αυτό σημαίνει πως θα γίνεται την ημέρα Τρίτη, στο δίωρο 15:00-17:00 και στην έκτη αίθουσα θεωρία. Κατά την εκτέλεση του προγράμματος τα μαθήματα των οποίων η θέση δεν ικανοποιεί τους περιορισμούς τότε αλλάζει θέση σε μία από τις άλλες μέσα στον πίνακα. Έτσι με την χρήση αυτού του πίνακα γίνεται πιο κατανοητή η καταχώρηση τιμών στις ιδιότητες της κλάσης Info.

Η κλάση που αντιστοιχεί στο Ωρολόγιο Πρόγραμμα είναι η **Schedule** (schedule στα αγγλικά σημαίνει πρόγραμμα). Στόχος αυτής της κλάσης είναι να χρησιμοποιήσει τον τρισδιάστατο πίνακα. Οι σημαντικότερες ιδιότητες της κλάσης αυτής είναι:

Matrix: Είναι τύπου της κλάσης Info και δείκτης (pointer). Το matrix στα αγγλικά σημαίνει πίνακας και αναφέρεται στον τρισδιάστατο πίνακα που περιγράψαμε προηγουμένως. Είναι δείκτης για να γίνει αργότερα στο πρόγραμμα ένας δυναμικός πίνακας μεγέθους ίσου με το πλήθος των μαθημάτων. Με αυτό τον τρόπο δίνουμε σε όλα τα μαθήματα τις τρείς μεταβλητές, που αναφέρονται στην ώρα, στην ημέρα και την αίθουσα, και αντιστοιχούν σε μια θέση στον τρισδιάστατο πίνακα.

List: Είναι τύπου της κλάσης Subject και δείκτης (στα αγγλικά list σημαίνει λίστα). Όπως και η προηγούμενη ιδιότητα θα γίνει και αυτή ένας δυναμικός πίνακας με μέγεθος το πλήθος των μαθημάτων. Με αυτόν τον τρόπο αντιστοιχίζουμε σε κάθε μάθημα τις ιδιότητες που είναι σταθερές, δηλαδή τον καθηγητή που παραδίδει το μάθημα, το εξάμηνο που το παρακολουθεί, το αν είναι δίωρο ή τρίωρο και εάν είναι θεωρητικό ή εργαστηριακό.

Το σημαντικό για αυτές τις δύο ιδιότητες είναι πως, όταν θα γίνουν πίνακες, θα υπάρχει αντιστοιχία μεταξύ των θέσεων τους. Δηλαδή η πρώτη θέση του κάθε πίνακα θα περιέχει πληροφορίες για το ίδιο μάθημα. Με αυτό τον τρόπο σε κάθε κουτάκι του τρισδιάστατου πίνακα του οποίου οι θέσεις περιγράφονται από τις ιδιότητες τις κλάσης Info αντιστοιχίζονται οι ιδιότητες της κλάσης Subject.

4.4.3 Περιορισμοί στο Πρόγραμμα

Όπως έχουμε πει για να βρούμε το κόστος του ωρολογίου προγράμματος πρέπει να ελέγξουμε πόσοι περιορισμοί παραβιάζονται. Για αυτό λοιπόν μέσα στο πρόγραμμα θα υλοποιήσουμε μία συνάρτηση κόστους οι οποία θα μα επιστρέφει το πλήθος των περιορισμών που παραβιάζονται. Και για να γίνει αυτό πρέπει να μοντελοποιήσουμε τους περιορισμούς. Οπότε παίρνουμε στη σειρά τους περιορισμούς όπως τους αναλύσαμε στην παράγραφο **ΑΡΙΘΜΟΣ ΠΑΡΑΓΑΦΟΥ**.

Κοινός καθηγητής

Η σειρά των διαδικασιών για να ελέγξουμε εάν ένας καθηγητής κάνει ένα μάθημα στην ίδια χρονική περίοδο είναι:

1) Αρχικά ελέγχω ένα δύο μαθήματα γίνονται την ίδια ημέρα, όπου στο πρόγραμμα θα είναι:

```
Matrix[i].get_day() == Matrix[j].get_day()
```

2) Στη συνέχεια ελέγχουμε εάν ένα μάθημα γίνεται σε κάποια χρονική περίοδο που συμπίπτει στην χρονική περίοδο που έχει τοποθετηθεί κάποιο άλλο. Για να το καταφέρουμε αυτό πρέπει να θυμηθούμε πως μοντελοποιήθηκαν οι χρονικές περίοδοι (δίωρα, τρίωρα) στην προηγούμενη παράγραφο (4.4.1). Ας θεωρήσουμε ότι ένα μάθημα είναι δίωρο και έχει τοποθετηθεί στη χρονική περίοδο 13:00-15:00, η οποία στο πρόγραμμα αντιστοιχεί στον αριθμό 9, τότε οι χρονικές περίοδοι με τις οποίες συμπίπτει (μαζί με τους αριθμούς από τους οποίους αντιπροσωπεύονται στο πρόγραμμα), εκτός από την ίδια, είναι οι 11:00-13:00 (7), 14:00-16:00 (11), 11:00-14:00 (6), 12:00-15:00 (8), 13:00-16:00 (10) και 14:00-17:00 (12). Αυτό που παρατηρούμε είναι πως η χρονική περίοδος με αριθμό 9 συμπίπτει με τις χρονικές περιόδους από 6 έως 12, δηλαδή από -3 έως +3 το οποίο συμβαίνει για κάθε δίωρο. Αντίστοιχα στην περίπτωση που το μάθημα είναι τρίωρο, δηλαδή βρίσκεται σε κάποια χρονική

περίοδο με ζυγό αριθμό, ελέγχουμε εάν τα υπόλοιπα μαθήματα έχουν τοποθετηθεί στο εύρος των χρονικών περιόδων από -4 έως +4. Για παράδειγμα εάν ένα μάθημα έχει τοποθετηθεί στην χρονική περίοδο 12:00-15:00 (8) θα πρέπει να ελέγξω τα μαθήματα που έχουν τοποθετηθεί στις χρονικές περιόδους από 10:00-13:00 (4) έως 14:00-17:00 (12). Έτσι βλέπω πως θα υπάρχουν δύο περιπτώσεις:

Ελέγχω εάν το μάθημα είναι δίωρο κοιτώντας εάν ο αριθμός της χρονικής του περιόδου είναι μονός (από το υπόλοιπο της διαίρεσης με το δύο), και στην συνέχεια ελέγχω εάν το άλλο μάθημα βρίσκεται στο εύρος -3 έως +3, το οποίο στο πρόγραμμα γίνεται ως εξής:

```
     Matrix[i].get_hour()%2==1                  &&
Matrix[j].get_hour()>=Matrix[i].get_hour()-3    &&
Matrix[j].get_hour()<=Matrix[i].get_hour()+3
```

Ελέγχω εάν το μάθημα είναι τρίωρο κοιτώντας εάν ο αριθμός της χρονικής του περιόδου είναι ζυγός (από το υπόλοιπο της διαίρεσης με το δύο), και στην συνέχεια ελέγχω εάν το άλλο μάθημα βρίσκεται στο εύρος -4 έως +4, το οποίο στο πρόγραμμα γίνεται ως εξής:

```
     Matrix[i].get_hour()%2==0                  &&
Matrix[j].get_hour()>=Matrix[i].get_hour()-4    &&
Matrix[j].get_hour()<=Matrix[i].get_hour()+4
```

3) Τέλος, εφόσον ξέρω πως δύο μαθήματα γίνονται την ίδια μέρα και σε χρονικές περιόδους που συμπίπτουν ελέγχω εάν για αυτά τα μαθήματα είναι υπεύθυνος ο ίδιος καθηγητής, το οποίο στο πρόγραμμα είναι:

```
List[i].get_prof()==List[j].get_prof()
```

Κοινό Εξάμηνο

Για να ελέγξω εάν ένα εξάμηνο κάνει δύο μαθήματα την ίδια ημέρα και σε χρονικές περιόδους που συμπίπτουν πρέπει να έχω κάνει τα δύο πρώτα βήματα που αναφέρθηκαν προηγουμένως και στην συνέχεια να ελέγξω εάν τα δύο μαθήματα που έχω βρει ανήκουν στο ίδιο εξάμηνο, το οποίο στο πρόγραμμα είναι:

```
List[i].get_sem()==List[j].get_sem()
```

Κοινή Αίθουσα

Αντίστοιχα με τους δύο προηγούμενους περιορισμούς για να ελέγξω εάν δύο μαθήματα γίνονται στην ίδια αίθουσα την ίδια ημέρα και σε χρονικές περιόδους που συμπίπτουν πρέπει να έχω κάνει τα δύο πρώτα βήματα, και στην συνέχεα να κάνω τον εξής έλεγχο στο πρόγραμμα:

```
Matrix[i].get_room()==Matrix[j].get_room()
```

Οι διδασκαλίες του μαθήματος να γίνονται σε διαφορετικές μέρες.

Το απαραίτητο σε αυτόν τον περιορισμό είναι να μοιράζονται οι διδασκαλίες των μαθημάτων. Η πιο συνηθισμένη περίπτωση είναι για ένα μάθημα να έχουμε δύο διδασκαλίες οι οποίες θέλουμε να μην γίνονται την ίδια ημέρα, αλλά υπάρχουν και άλλες περιπτώσεις μαθημάτων όπου έχουν περισσότερες από δύο διδασκαλίες.

Ας δούμε το πιο ακραίο παράδειγμα στην τμήμα μας όπου είναι αυτό του μαθήματος της Ηλεκτρονικής ΙΙ το οποίο έχει πέντε διδασκαλίες, μία δίωρη και μία τρίωρη διδασκαλία θεωρίας και τρείς δίωρες διδασκαλίες εργαστηρίου. Σε αυτήν τη περίπτωση δεν θέλουμε να μοιράσουμε τις 5 αυτές διδασκαλίες στις πέντε ημέρες τις εβδομάδας αλλά ούτε και να τις έχουμε και τις πέντε την ίδια ημέρα. Οπότε πρέπει να έχουμε έναν σύνθετο τρόπο καταμερισμού των διδασκαλιών.

Αυτό που επιλέχθηκε είναι σε περίπτωση που ένα μάθημα έχει ζυγό αριθμό διδασκαλιών τότε να γίνονται το πολύ οι μισές σε μία ημέρα, για παράδειγμα εάν ένα μάθημα χωρίζεται τέσσερις διδασκαλίες να μπορούν να γίνουν το πολύ δύο από αυτές την ίδια ημέρα. Στην περίπτωση που το πλήθος των διδασκαλιών στο οποίο χωρίζεται ένα μάθημα είναι μονό, τότε να μπορούν να γίνουν οι μισές στρογγυλοποιώντας προς τον μεγαλύτερο ακέραιο, δηλαδή όταν ένα μάθημα χωρίζεται τρεις διδασκαλίες να μπορούν να γίνουν το πολύ δύο σε μια μέρα.

Για να γίνει αυτός ο έλεγχος στο πρόγραμμα πρέπει να ξέρουμε το πλήθος των διδασκαλιών ανά μάθημα, το οποίο γίνεται έχοντας τον δυναμικό πίνακα `lec_per_sub` (lecture per sub ,δηλ. διδασκαλία ανα μάθημα), και χωρίζοντας τις δύο περιπτώσεις:

1) Το μάθημα να έχει ζυγό αριθμό διδασκαλιών, άρα να μην ξεπερνάει τις μισές ανά ημέρα, και στο πρόγραμμα είναι:

```
lec_perday_sub>lec_per_sub[List[i].get_sub_id()]/2
```

2) Το μάθημα να έχει μονό αριθμό διδασκαλιών, άρα να μην ξεπερνάει τις μισές στρογγυλοποιώντας προς τον μεγαλύτερο ακέραιο ανά ημέρα, και στο πρόγραμμα είναι:

```
lec_perday_sub>(lec_per_sub[List[i].get_sub_i
d()]/2)+1
```

Οι καθηγητές να μην ξεπερνούν ένα όριο μαθημάτων ανά ημέρα

Για αυτόν τον περιορισμό θα έχουμε ορίσει στη αρχή του προγράμματος το όριο των μαθημάτων που επιτρέπεται να κάνει ένα καθηγητής σε μία ημέρα και είναι η μεταβλητή `prof_lim` (professor limit, δήλ. το όριο του καθηγητή) και θα ελέγχουμε εάν ο καθηγητής του μαθήματος το οποίο μας ενδιαφέρει κάνει περισσότερα από το όριο μαθήματα στην ίδια μέρα, και το πλήθος θα το κρατάμε στην μεταβλητή `lectures_perday_prof` (δήλ. μαθήματα που κάνει ένας καθηγητής ανά ημέρα). Στο πρόγραμμα θα είναι:

```
lectures_perday_prof>prof_lim
```

Τα εξάμηνα να μην ξεπερνούν ένα όριο μαθημάτων ανά ημέρα

Για αυτόν τον περιορισμό θα έχουμε ορίσει στη αρχή του προγράμματος το όριο των μαθημάτων που επιτρέπεται να κάνει ένα εξάμηνο σε μία ημέρα και είναι η μεταβλητή `sem_lim` (semester limit, δήλ. το όριο του εξαμήνου) και θα ελέγχουμε εάν το εξάμηνο στο οποίο ανήκει το μάθημα το οποίο μας ενδιαφέρει κάνει περισσότερα από το όριο μαθήματα στην ίδια μέρα, και το πλήθος θα το κρατάμε στην μεταβλητή `lectures_perday_sem` (δήλ. μαθήματα που κάνει ένα εξάμηνο ανά ημέρα). Στο πρόγραμμα θα είναι:

```
lectures_perday_sem>sem_lim
```

4.4.4 Αρχικοποίηση του Πίνακα

Για να έχουμε μία πιο ευρεία εικόνα της αποδοτικότητας των αλγορίθμων που θα εφαρμοστούν πάνω στο ωρολόγιο πρόγραμμα η αρχικοποίηση θα γίνεται με δύο διαφορετικούς τρόπους. Μία εντελώς τυχαία ταξινόμηση των διδασκαλιών στο πρόγραμμα και μία άπληστη μέθοδο η οποία θα τοποθετεί τα μαθήματα σε θέσεις όπου το κόστος θα είναι ελάχιστο.

1) Η τυχαία ταξινόμηση επιτυγχάνεται πολύ απλά χρησιμοποιώντας τη συνάρτηση rand (random, δηλ. τυχαία) της c++. Με αυτή την μέθοδο υπάρχει ένα εύρος πιθανών αποτελεσμάτων από το να έχουμε μία τέλεια ταξινόμηση, δηλαδή μία αποδεκτή λύση του προβλήματος, μέχρι το να ταξινομηθούν όλες οι διδασκαλίες στην ίδια χρονική περίοδο και την ίδια αίθουσα.

2) Η άπληστη μέθοδος είναι λίγο πιο περίπλοκη και χρονοβόρα, επειδή καθώς θα ταξινομούνται οι διαλέξεις θα γίνεται έλεγχος τον περιορισμών που παραβιάζονται για κάθε πιθανή θέση μέσα στο ωρολόγιο πρόβλημα. Με αυτό τον τρόπο θα έχουμε μια αρχική κατάσταση η οποία θα έχει ένα μικρότερο κόστος από αυτό που πιθανότατα θα μας έδινε η τυχαία αρχικοποίηση, οπότε πιθανότατα οι αλγόριθμοι θα χρειάζονται ένα μικρότερο αριθμό κινήσεων για να βρουν λύση. Το μειονέκτημα αυτής της μεθόδου είναι πως πολύ συχνά μπορεί να προκαλέσει στους αλγορίθμους να κολλήσουν σε τοπικό βέλτιστο.

Κεφάλαιο 5 : Ανάλυση Αλγορίθμων

5.1 Γενικά

Στο κεφάλαιο αυτό αναλύονται οι αλγόριθμοι τοπικής αναζήτησης που αναπτύχθηκαν για το πρόβλημα κατασκευής του προγράμματος μαθημάτων ενός ακαδημαϊκού εξαμήνου .Για κάθε αλγόριθμο που αναλύεται θα δίνεται λεπτομερής περιγραφή του, ψευδοκώδικας καθώς και ο κώδικας σε C++ όπως υλοποιήθηκε στο πρόγραμμα, και ο τρόπος διαχείρισης των δομών δεδομένων που αναπτύχθηκαν στο προηγούμενο κεφάλαιο .Ένα κοινό χαρακτηριστικό των αλγοριθμικών μεθόδων που αναλύθηκαν είναι ότι ανήκουν στην κατηγορία των ευρετικών μηχανισμών, εκτός του ότι είναι αλγόριθμοι τοπικής αναζήτησης φυσικά .Οι ευρετικοί μηχανισμοί ξεχωρίζουν από τις υπόλοιπες γιατί χρησιμοποιούν κάποια πληροφορία για να φτάσουν προς την λύση .Να σημειωθεί ότι η πληροφορία που χρησιμοποιούν δείχνει μία εκτίμηση για τον δρόμο προς την λύση ,και όχι την λύση.

Οι αλγόριθμοι τοπικής αναζήτησης που επιλέχθηκαν, χρησιμοποιήθηκαν και αξιολογήθηκαν είναι τρεις : ο αλγόριθμος ελαχίστων συγκρούσεων (Min-conflicts), ο αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις (Min-conflicts with random restarts), και ο αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις και επιλογή της μεταβλητής με τις περισσότερες συγκρούσεις (Min-conflicts with random restarts and selecting the most Conflicting variable) .

5.2 Αλγόριθμος ελαχίστων συγκρούσεων (Min-conflicts)

5.2.1 Εισαγωγή

Αν και η Τεχνητή Νοημοσύνη γνώριζε και αντιμετώπιζε Προβλήματα Ικανοποίησης Περιορισμών για πολλά χρόνια, δεν ήταν μέχρι τις αρχές της δεκαετίας του 1990 όταν η διαδικασία για την επίλυση των μεγάλων CSP κωδικοποιήθηκε σε αλγοριθμική μορφή. Αρχικά, ο Mark Johnston του Επιστημονικού Ινστιτούτου Διαστημικού Τηλεσκοπίου έψαξε για μια μέθοδο για να προγραμματίσει τις αστρονομικές παρατηρήσεις με το διαστημικό τηλεσκόπιο Hubble. Στο πλαίσιο των πειραμάτων αυτών, δημιούργησε ένα νευρωνικό δίκτυο ικανό να λύσει ένα πρόβλημα N-βασιλισσών (για 1024 βασίλισσες). Ο Steven Minton και ο Andy Philips ανέλυσαν τον αλγόριθμο νευρωνικού δικτύου και τον διαχώρισε σε δύο φάσεις: (1) μια αρχική ανάθεση χρησιμοποιώντας ένα άπληστο αλγόριθμο και (2) τη φάση ελαχιστοποίησης των συγκρούσεων (που αργότερα ονομάστηκε «min-conflicts»).

Ο αλγόριθμος ελαχίστων συγκρούσεων έχει χρησιμοποιηθεί ευρέως σε προβλήματα περιορισμών ,και ειδικότερα σε προβλήματα timetabling . Η επιτυχία του οφείλεται στην μεγάλη ταχύτητα για επίλυση και στην απλή φιλοσοφία του. Με τον αλγόριθμο αυτό επιλύεται εύκολα το πρόβλημα κατασκευής του εβδομαδιαίου προγράμματος μαθημάτων ενός ακαδημαϊκού εξαμήνου .

5.2.2 Ανάλυση αλγορίθμου ελαχίστων συγκρούσεων

Ο αλγόριθμος ελαχίστων συγκρούσεων ή πιο απλά min-conflicts , δέχεται ως είσοδο ένα πρόβλημα ικανοποίησης περιορισμών και επιστρέφει μία λύση ή

αποτυχία .Οι είσοδοί του είναι το πρόβλημα περιορισμών(CSP) και ένας μέγιστος αριθμός βημάτων (max_βήματα) που επιτρέπεται να τρέξει ο αλγόριθμος πριν παραιτηθεί από την προσπάθεια για επίλυση .Τα βήματα του αλγορίθμου έχουν ως εξής :

1. Γίνεται μία πλήρης αρχική ανάθεση τυχαίων τιμών στις μεταβλητές του προβλήματος.
2. Ο αλγόριθμος μπαίνει σε βρόγχο ,ο οποίος έχει όριο βημάτων τον μέγιστο αριθμό βημάτων που έχει ως είσοδο ο αλγόριθμος.
3. Αν η ανάθεση τιμών που έχει γίνει είναι μία λύση που ικανοποιεί τους περιορισμούς τότε επιστρέφεται αυτή η πλήρης ανάθεση και σταματάει η λειτουργία του αλγορίθμου.
4. Αν δεν ισχύει το προηγούμενο, τότε επιλέγεται μία τυχαία μεταβλητή που έχει συγκρούσεις από το σύνολο μεταβλητών.
5. Επιλέγεται μία τιμή που ελαχιστοποιεί τις συγκρούσεις για την μεταβλητή που επιλέχθηκε πριν.
6. Η μεταβλητή που επισκευάστηκε εισάγεται στο σύνολο των μεταβλητών ,και έτσι υπάρχει μία άλλη πλήρης ανάθεση τιμών.
7. Αν δεν έχει βρεθεί λύση και εξαντληθεί ο αριθμός των βημάτων του αλγορίθμου ,τότε ο αλγόριθμος επιστρέφει αποτυχία .

Ακολουθεί ο ψευδοκώδικας που περιγράφεται πλήρως τη λειτουργία του min-conflicts :

Function MIN-CONFLICTS(csp,max_βήματα) επιστρέφει μία λύση ή αποτυχία.

inputs : csp, ένα πρόβλημα ικανοποίησης περιορισμών,
max_βήματα, αριθμός βημάτων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια

τρέχουσα ← μία πλήρης ανάθεση τιμών για το csp
for i=1 to max_ βήματα **do**
if τρέχουσα είναι λύση για το csp **then return** τρέχουσα

μεταβλητή ← τυχαία επιλεγμένη μεταβλητή με διένεξη από το VARIABLES[CSP]

τιμή ← μία τιμή u της *μεταβλητής* που ελαχιστοποιεί την CONFLICTS(μεταβλητή,u,τρέχουσα,CSP)

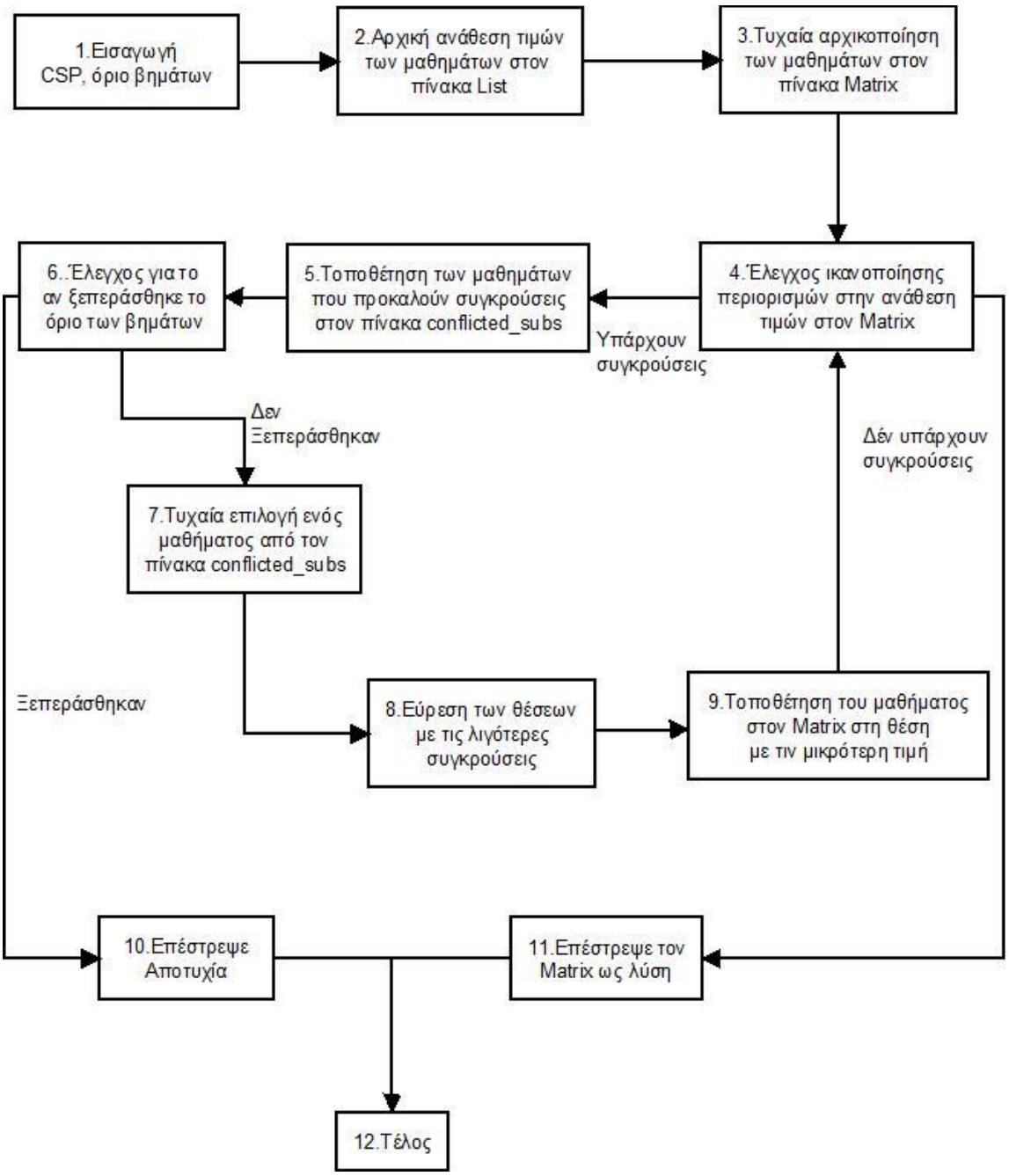
τίθεται μεταβλητή = τιμή στη τρέχουσα
return αποτυχία

Το σύνολο VARIABLES[CSP] είναι το σύνολο των μεταβλητών του προβλήματος για επίλυση .Η συνάρτηση CONFLICTS(μεταβλητή,u,τρέχουσα,CSP) μετρά τον αριθμό των περιορισμών που παραβιάζονται από μία συγκεκριμένη τιμή, αφού συγκριθεί με τις υπόλοιπες μεταβλητές μίας πλήρους ανάθεσης .

5.2.3 Διαχείριση δομών δεδομένων από min-conflicts

Η αλληλεπίδραση του αλγορίθμου με τις δομές δεδομένων παρουσιάζει μεγάλο ενδιαφέρον γιατί δείχνει ,όχι μόνο πως διαχειρίζονται οι δομές δεδομένων, αλλά και την κίνηση του αλγόριθμου προς την λύση σε σχέση με τις δομές.

Διαγραμματικά ο αλγόριθμος φαίνεται στο σχήμα που ακολουθεί :



Σχήμα 5.1 : Διάγραμμα διαχείρισης των δομών δεδομένων από min-conflicts.

Το διάγραμμα του αλγορίθμου min-conflicts αναλύεται ,όσον αφορά τις καταστάσεις στα παρακάτω βήματα:

1. Ορίζεται το αρχικό πρόβλημα και δίνεται ο μέγιστος αριθμός βημάτων. Επίσης, δίνεται ο αριθμός των διαθέσιμων δωματίων. Η είσοδος που παίρνει το πρόγραμμα έχει τη μορφή του εξής πίνακα:

Αρ. Μαθήματος (sub_id)	Καθηγητής (prof)	Εξάμηνο (sem)	Είδος Μαθήματος (kind)	Χρόνος (hours)
0	1	1	0	1
1	2	1	0	0
1	2	1	1	0
2	3	1	0	1
2	3	1	1	0
3	4	1	0	0
3	4	1	0	1
3	4	1	1	0
4	5	1	0	1
4	5	1	1	1
5	1	1	0	0
5	1	1	1	1
6	6	1	0	1
6	6	1	0	0
7	2	3	0	1
7	2	3	1	0
7	2	3	1	0
8	7	3	0	0

Πίνακας 5.1 : Μορφή της εισόδου που παίρνει το πρόγραμμα.

2. Ο πίνακας αυτός στην υλοποίηση παίρνεται σαν είσοδος από ένα αρχείο txt. Ύστερα, τα στοιχεία από το αρχείο μπαίνουν σε κάθε γραμμή του πίνακα List, στα αντίστοιχα πεδία της κάθε διδασκαλίας. Δηλαδή ,για τον προηγούμενο πίνακα η $1^{η}$ διδασκαλία θα έχει το νούμερο 1, θα ανήκει στο μάθημα1, θα το διδάσκει ο καθηγητής με νούμερο 1, θα είναι μάθημα του πρώτου εξαμήνου σπουδών και θα είναι δίωρο θεωρίας. Παρόμοια αρχικοποιούνται και τα υπόλοιπα μαθήματα.

3. Στη συνέχεια αρχικοποιείται ο πίνακας Matrix σύμφωνα με τον προηγούμενο πίνακα και παίρνουν τιμές η ημέρα ,η ώρα και το δωμάτιο .Οι αρχικές αυτές τιμές δίνονται με ένα γεννήτορα τυχαίων αριθμών (random number generator), όπως αναφέραμε, ανάλογα με το πεδίο τιμών για κάθε πεδίο του ή με μία άπληστη αρχικοποίηση.

4. Στο βήμα αυτό εισάγεται το πρόβλημα στον βρόγχο και ελέγχονται όλοι οι περιορισμοί (βλέπε κεφ.4) στον πίνακα Matrix. Επίσης ,αυξάνονται τα βήματα του

αλγορίθμου κατά ένα. Αν δεν υπάρχουν συγκρούσεις τότε ο αλγόριθμος πηγαίνει στο [11] .Αν υπάρχουν ,τότε πηγαίνει στο [5] .

5. Βρίσκει τα μαθήματα που προκαλούν συγκρούσεις και τα τοποθετεί στον πίνακα conflicted_subs, έτσι στη συνέχει μπορεί ο αλγόριθμος να επιλέξει κάποιο για να αλλάξει.

6. Γίνεται έλεγχος για το αν έχουν τελειώσει τα βήματα που μπορεί να τρέξει ακόμα ο αλγόριθμος. Αν έχουν τελειώσει ,τότε ο αλγόριθμος δεν έχει καταφέρει να λύσει το πρόβλημα και μεταβαίνει στην κατάσταση [10]. Αν τα βήματα δεν έχουν τελειώσει ,τότε ο αλγόριθμος πηγαίνει στο [7].

7. Επιλέγεται τυχαία ένα μάθημα (διδασκαλία στην ουσία) από αυτά που προκαλούν συγκρούσεις. Η επιλογή αυτή γίνεται τυχαία πάλι με ένα random number generator. Αυτό γίνεται στη συνάρτηση find_collisions.

8. Αφού επιλεχθεί ένα μάθημα το οποίο προκαλεί συγκρούσεις, τότε γίνεται ένας έλεγχος για το ποια θέση στο ωρολόγιο πρόγραμμα ελαχιστοποιεί τις συγκρούσεις για αυτό το μάθημα. Εάν περισσότερες από μια θέσεις ελαχιστοποιούν τις συγκρούσεις τότε επιλέγεται τυχαία μια από αυτές. Αυτό γίνεται στη συνάρτηση change_conflicted.

9. Όταν επιλεχθεί η νέα θέση για το μάθημα τότε ανανεώνεται ο πίνακας Matrix και έχουμε το νέο ωρολόγιο πρόγραμμα ώστε να δούμε εάν έχει λυθεί το πρόβλημα, για αυτό επιστρέφει στο [4].

10. Στην περίπτωση όπου το πρόβλημα δεν έχει λύση ή εξαντλήθηκε ο μέγιστος αριθμός βημάτων (από το [6]) τότε ο αλγόριθμος απέτυχε.

11. Στην περίπτωση όπου όλοι οι περιορισμοί ικανοποιούνται τότε ο αλγόριθμος έχει βρει λύση και αυτή είναι η τρέχουσα κατάσταση του πίνακα Matrix.

5.2.4 Υλοποίηση του αλγόριθμου min-conflicts στο πρόγραμμα

Στη συνέχεια θα δούμε μία παρουσίαση και επεξήγηση του πως υλοποιήθηκε ο αλγόριθμος αυτός μέσα στο πρόγραμμα. Ακολουθεί το κομμάτι του κώδικα που αντιστοιχεί στον αλγόριθμο min-conflicts.

```
create_List();
create_Matrix();
do{
      count = check_for_collisions();
      if(count!=0){
            find_collisions();
            change_conflicted();
      }
      num_of_tries++;
}while(count!=0 && num_of_tries<lim_of_tries);
```

Στην αρχή έχουμε την συνάρτηση create_List η οποία παίρνει τις πληροφορίες για τα μαθήματα και τις αποθηκεύει μέσα στο πρόγραμμα στον δυναμικό πίνακα List. Η επόμενη συνάρτηση είναι η create_Matrix η οποία δημιουργεί μία τυχαία αρχικοποίηση του ωρολογίου προγράμματος, ταξινομώντας το κάθε μάθημα σε μία τυχαία μέρα, ώρα και αίθουσα.

Στη συνέχεια έχουμε τον βρόγχο, με πλήθος επαναλήψεων ίσο με το όριο των βημάτων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια να λυθεί το πρόβλημα. Από την συνάρτηση check_for_collisions μας επιστρέφεται το πλήθος των συγκρούσεων μέσα στο πρόγραμμα. Εάν το πλήθος αυτό είναι μηδέν, τότε το πρόβλημα είναι λυμένο, αλλά εάν δεν είναι μηδέν τότε καλούνται δύο άλλες συναρτήσεις, η find_collisions η οποία βρίσκει ποια μαθήματα προκαλούν συγκρούσεις και η change_conflicted η οποία επιλέγει τυχαία ένα από αυτά τα μαθήματα και το τοποθετεί στην καλύτερη πιθανή θέση στο πρόγραμμα ώστε να ελαχιστοποιηθούν η συγκρούσεις που προκαλεί.

Τέλος έχουμε τον έλεγχο για το αν έχει βρεθεί λύση ή αν έχουν τελειώσει τα επιτρεπτά βήματα.

Το μειονέκτημα του min-conflicts είναι ότι μπορεί να οδηγηθεί σε αδιέξοδο πολύ εύκολα. Το προηγούμενο μπορεί να συμβεί γιατί ο δρόμος προς την λύση ξεκινά από ένα σημείο, την πλήρη αρχική ανάθεση που κάνει ο αλγόριθμος. Υπάρχει περίπτωση από την αρχική ανάθεση για να φτάσει κανείς στην λύση να χρειάζονται πολλά βήματα απ' ότι από μία άλλη αρχική ανάθεση .Αυτό στο min-conflicts δεν μπορεί να γίνει γιατί δεν γίνονται επανεκκινήσεις, δηλαδή πρακτικά δεν γίνονται παραπάνω από μία αρχικές πλήρεις αναθέσεις. Το κενό αυτό έρχεται να αναπληρώσει ο επόμενος αλγόριθμος.

5.3. Αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις (Min-conflicts with random restarts)

5.3.1 Εισαγωγή

Ένας παρεμφερής αλγόριθμος που βασίζεται στη min-conflicts είναι ο αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις (Min-Coflicts with random restarts) .Για ευκολία θα χρησιμοποιείται το όνομα Min-conflicts-WR για τον αλγόριθμο αυτό .Ο Random-Restarts έχει μία βασική διαφορά με την min-conflicts : μπορεί να κάνει επανεκκίνηση στο πρόβλημα που εξετάζει . Όλες οι διαφορές με την min-conflicts αναλύονται καλύτερα παρακάτω .

5.3.2 Ανάλυση αλγορίθμου Min-conflicts-WR

Ο αλγόριθμος Min-conflicts-WR, δέχεται ως είσοδο ένα πρόβλημα ικανοποίησης περιορισμών και επιστρέφει μία λύση ή αποτυχία .Οι είσοδοί του είναι το πρόβλημα περιορισμών(CSP), ένας μέγιστος αριθμός βημάτων (max_ βήματα) για επισκευή ,και ένας μέγιστος αριθμός επανεκκινήσεων που επιτρέπεται να τρέξει ο αλγόριθμος πριν παραιτηθεί από την προσπάθεια για επίλυση .Τα βήματα του αλγορίθμου έχουν ως εξής :

1. **Αφού δοθούν οι είσοδοι για το πρόβλημα ο αλγόριθμος μπαίνει σε βρόγχο, ο οποίος έχει όριο βημάτων τον μέγιστο αριθμό επανεκκινήσεων που έχει ως είσοδο ο αλγόριθμος.**
2. Γίνεται μία πλήρης αρχική ανάθεση τυχαίων τιμών στις μεταβλητές του προβλήματος.
3. Ο αλγόριθμος μπαίνει σε βρόγχο, ο οποίος έχει όριο βημάτων τον μέγιστο αριθμό βημάτων που έχει ως είσοδο ο αλγόριθμος.
4. Αν η ανάθεση τιμών που έχει γίνει είναι μία λύση που ικανοποιεί τους περιορισμούς τότε επιστρέφεται αυτή η πλήρης ανάθεση και σταματάει η λειτουργία του αλγορίθμου.
5. Αν δεν ισχύει το προηγούμενο, τότε επιλέγεται μία τυχαία μεταβλητή που έχει συγκρούσεις από το σύνολο μεταβλητών.
6. Επιλέγεται μία τιμή που ελαχιστοποιεί τις συγκρούσεις για την μεταβλητή που επιλέχθηκε πριν.
7. Η μεταβλητή που επισκευάστηκε εισάγεται στο σύνολο των μεταβλητών ,και έτσι υπάρχει μία άλλη πλήρης ανάθεση τιμών.
8. **Αν δεν έχει βρεθεί λύση και εξαντληθεί ο αριθμός των βημάτων του αλγορίθμου, τότε ο αλγόριθμος επιστρέφει στο 2.**
9. **Αν δεν έχει βρεθεί λύση και εξαντληθεί ο αριθμός των επανεκκινήσεων του αλγορίθμου, τότε ο αλγόριθμος επιστρέφει αποτυχία.**

Με τον ψευδοκώδικα που ακολουθεί περιγράφεται πλήρως η λειτουργία της Min-conflicts-WR:

Function MIN-CONFLICTS-WR(csp,max_βήματα) επιστρέφει μία λύση ή αποτυχία.

inputs : csp, ένα πρόβλημα ικανοποίησης περιορισμών,
max_βήματα, αριθμός βημάτων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια
max_επανεκκινήσεις, αριθμός επανεκκινήσεων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια

for j=1 to max_επανεκκινήσεις **do**
τρέχουσα ← μία πλήρης ανάθεση τιμών για το csp
for i=1 to max_ βήματα **do**
if τρέχουσα είναι λύση για το csp **then return** τρέχουσα

μεταβλητή ← τυχαία επιλεγμένη μεταβλητή με διένεξη από το VARIABLES[CSP]

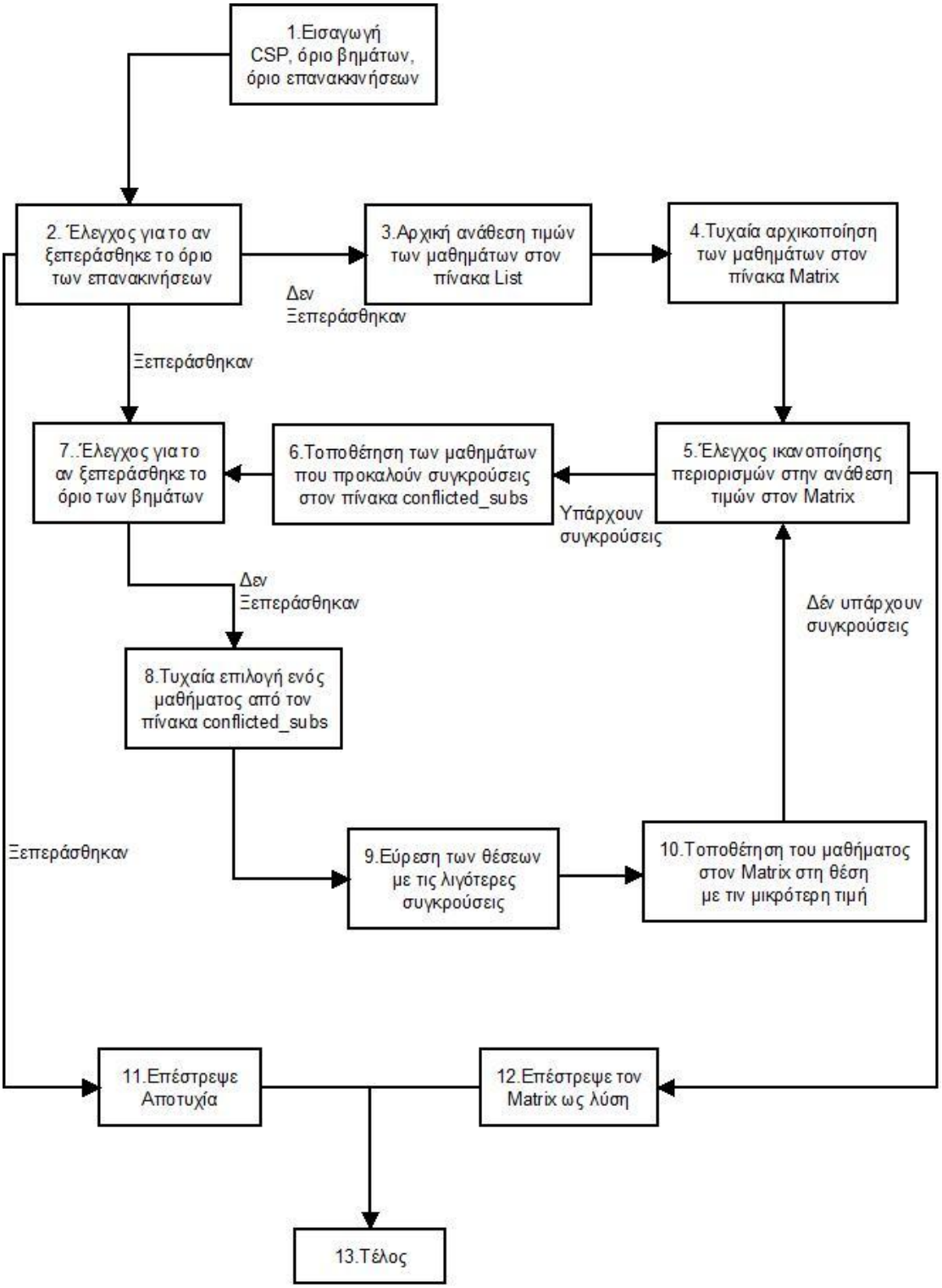
τιμή ← μία τιμή u της *μεταβλητής* που ελαχιστοποιεί την CONFLICTS(μεταβλητή,u,τρέχουσα,CSP)

τίθεται μεταβλητή = τιμή στη τρέχουσα
return αποτυχία

5.3.3 Διαχείριση δομών δεδομένων από Min-conflicts-WR

Η διαφορά του Min-conflicts-WR με το min-conflicts δεν είναι μόνο ο σχεδιασμός του αλγορίθμου .Στο πρόβλημα της κατασκευής του προγράμματος μαθημάτων γίνεται και διαφορετική διαχείριση των δομών δεδομένων .

Διαγραμματικά ο αλγόριθμος φαίνεται στο σχήμα που ακολουθεί :



Σχήμα 5.2 : Διάγραμμα διαχείρισης των δομών δεδομένων από min-conflicts-wr.

Το διάγραμμα του αλγορίθμου Min-conflicts-WR διαφέρει όσον αφορά τις καταστάσεις στα παρακάτω βήματα από την min-conflicts:

- Μετά από την εισαγωγή στο πρώτο βρόγχο ,που ελέγχει τον αριθμό των επανεκκινήσεων ,γίνεται η ανάθεση τιμών (επανεκκίνηση) και όχι στην αρχή του αλγορίθμου.
- Η πρώτη κατάσταση μετά την είσοδο των δεδομένων είναι η εισαγωγή στον βρόγχο που είναι υπεύθυνος για τις επανεκκινήσεις .Μετά από αυτή την κατάσταση υπάρχουν δύο περιπτώσεις .Αν έχουν τελειώσει τα βήματα επανεκκίνησης ο αλγόριθμος επιστρέφει αποτυχία .Αν δεν έχουν τελειώσει τα βήματα επανεκκίνησης ο αλγόριθμος προχωρά σε πλήρη ανάθεση τιμών.

Ο αλγόριθμος Min-conflicts-WR είναι στην ουσία ένας αλγόριθμος min-conflicts που κάνει επανεκκινήσεις στο πρόβλημα. Με αυτό τον τρόπο, γίνεται πιο δύσκολο να 'κολλήσει' ο αλγόριθμος σε τοπικά μέγιστα, γιατί όταν κολλήσει στο πρώτο τοπικό μέγιστο θα γίνει επανεκκίνηση. Βέβαια, ακόμα υπάρχει η περίπτωση του τοπικού μέγιστου στην περίπτωση που τελειώσουν τα βήματα επανεκκίνησης. Το μειονέκτημα και στις δύο προηγούμενες αλγοριθμικές τεχνικές είναι ότι η μεταβλητή, το μάθημα δηλαδή, επιλέγεται τυχαία από το σύνολο των μεταβλητών που έχουν conflicts. Την τυχαιότητα της επιλογής μεταβλητής για επισκευή έρχεται να εξαλείψει ο επόμενος αλγόριθμος.

5.3.4.Υλοποίηση του αλγόριθμου min-conflict-wr στο πρόγραμμα

Εδώ θα δούμε μία παρουσίαση και επεξήγηση του πως υλοποιήθηκε ο αλγόριθμος αυτός μέσα στο πρόγραμμα. Ακολουθεί το κομμάτι του κώδικα που αντιστοιχεί στον αλγόριθμο min-conflicts-wr.

```
do{
      create_List();
      create_Matrix();
      num_of_tries=0;
      do{
            count = check_for_collisions();
            if(count!=0){
                  find_collisions();
                  change_conflicted();
            }
            num_of_tries++;
      }while(count!=0 && num_of_tries<lim_of_tries);
      num_of_starts++;
}while(count!=0 && num_of_starts<lim_of_starts);
```

Στην αρχή έχουμε ένα βρόγχο, με πλήθος επαναλήψεων το όριο των επανακκινήσεων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια να λυθεί το πρόβλημα.

Παρόμοια με τον αλγόριθμο min-conflicts έχουμε τις συναρτήσεις create_List και create_Matrix η οποίες αρχικοποιούν τους πίνακες List και Matrix αντίστοιχα. Στη συνέχεια έχουμε τον βρόγχο, με πλήθος επαναλήψεων ίσο με το όριο των βημάτων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια να λυθεί το πρόβλημα και τις συναρτήσεις που υλοποιούν τα βήματα του αλγορίθμου, την τυχαία επιλογή

μιας μεταβλητής που έχει conflicts και την τοποθέτηση της στην καλύτερη πιθανή θέση στο πρόγραμμα.

Στο τέλος, πρώτα έχουμε τον έλεγχο για το αν έχει βρεθεί λύση ή αν έχουν τελειώσει τα επιτρεπτά βήματα, και στη συνέχεια τον έλεγχο για το αν έχει βρεθεί λύση ή αν έχουν τελειώσει οι επιτρεπτές επανεκκινήσεις.

Ο αλγόριθμος Min-conflicts-WR είναι στην ουσία ένας αλγόριθμος min-conflicts που κάνει επανεκκινήσεις στο πρόβλημα. Με αυτό τον τρόπο, γίνεται πιο δύσκολο να 'κολλήσει' ο αλγόριθμος σε τοπικά μέγιστα, γιατί όταν κολλήσει στο πρώτο τοπικό μέγιστο θα γίνει επανεκκίνηση. Βέβαια, ακόμα υπάρχει η περίπτωση του τοπικού μέγιστου στην περίπτωση που τελειώσουν τα βήματα επανεκκίνησης. Το μειονέκτημα και στις δύο προηγούμενες αλγοριθμικές τεχνικές είναι ότι η μεταβλητή, το μάθημα δηλαδή, επιλέγεται τυχαία από το σύνολο των μεταβλητών που έχουν conflicts. Την τυχαιότητα της επιλογής μεταβλητής για επισκευή έρχεται να εξαλείψει ο επόμενος αλγόριθμος.

5.4. Αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις και επιλογή της μεταβλητής με τις περισσότερες συγκρούσεις (Min-conflicts with random restarts and selecting the most Conflicting variable)

5.4.1.Εισαγωγή

Ένας άλλος παρεμφερής αλγόριθμος που βασίζεται στη min-conflicts είναι ο αλγόριθμος ελαχίστων συγκρούσεων με επανεκκινήσεις και επιλογή της μεταβλητής με τις περισσότερες συγκρούσεις (Min-Coflicts with random restarts and selecting the most conflicting variable) .Για ευκολία θα χρησιμοποιείται το όνομα Most-Conflicts-WR τον αλγόριθμο αυτό. Οι διαφορές με τους δύο προηγούμενους αλγορίθμους αναλύονται παρακάτω.

5.4.2.Ανάλυση αλγορίθμου Most-Conflicted-Variable

Ο αλγόριθμος Most-Conflicts-WR, δέχεται ως είσοδο ένα πρόβλημα ικανοποίησης περιορισμών και επιστρέφει μία λύση ή αποτυχία .Οι είσοδοί του είναι το πρόβλημα περιορισμών(CSP), ένας μέγιστος αριθμός βημάτων (max_βήματα) για επισκευή, και ένας μέγιστος αριθμός επανεκκινήσεων που επιτρέπεται να τρέξει ο αλγόριθμος πριν παραιτηθεί από την προσπάθεια για επίλυση. Η βασική διαφορά με την προηγούμενη μέθοδο είναι η επιλογή της μεταβλητής, τις οποίας θα αλλάξουν οι τιμές .Τα βήματα του αλγορίθμου έχουν ως εξής :

1. Αφού δοθούν οι είσοδοι για το πρόβλημα ο αλγόριθμος μπαίνει σε βρόγχο, ο οποίος έχει όριο βημάτων τον μέγιστο αριθμό επανεκκινήσεων που έχει ως είσοδο ο αλγόριθμος.

2. Γίνεται μία πλήρης αρχική ανάθεση τυχαίων τιμών στις μεταβλητές του προβλήματος.
3. Ο αλγόριθμος μπαίνει σε βρόγχο, ο οποίος έχει όριο βημάτων τον μέγιστο αριθμό βημάτων που έχει ως είσοδο ο αλγόριθμος.
4. Αν η ανάθεση τιμών που έχει γίνει είναι μία λύση που ικανοποιεί τους περιορισμούς τότε επιστρέφεται αυτή η πλήρης ανάθεση και σταματάει η λειτουργία του αλγορίθμου.
5. **Αν δεν ισχύει το προηγούμενο ,τότε επιλέγεται η μεταβλητή που έχει τις περισσότερες συγκρούσεις από το σύνολο μεταβλητών .**
6. Επιλέγεται μία τιμή που ελαχιστοποιεί τις συγκρούσεις για την μεταβλητή που επιλέχθηκε πριν.
7. Η μεταβλητή που επισκευάστηκε εισάγεται στο σύνολο των μεταβλητών ,και έτσι υπάρχει μία άλλη πλήρης ανάθεση τιμών.
8. Αν δεν έχει βρεθεί λύση και εξαντληθεί ο αριθμός των βημάτων του αλγορίθμου, τότε ο αλγόριθμος επιστρέφει στο 2.
9. Αν δεν έχει βρεθεί λύση και εξαντληθεί ο αριθμός των επανεκκινήσεων του αλγορίθμου, τότε ο αλγόριθμος επιστρέφει αποτυχία.

Η βασική διαφορά με τον προηγούμενο αλγόριθμο είναι στο βήμα 5, που επιλέγεται για επισκευή η μεταβλητή με τις περισσότερες συγκρούσεις. Με τον τρόπο αυτό γίνεται προσπάθεια να βελτιωθεί ο χρόνος επίλυσης, γιατί επιλέγοντας μία μεταβλητή που έχει τις περισσότερες συγκρούσεις, και δίνοντας προτεραιότητα στην επισκευή αυτών των μεταβλητών σε κάθε βήμα του αλγορίθμου, φτάνει ο αλγόριθμος πιο γρήγορα στην λύση. Η λογική αυτού του αλγορίθμου είναι να εξαλείψει τις πολλές συγκρούσεις από νωρίς, για να μείνουν οι μεταβλητές με τις λίγες συγκρούσεις ,αυτές δηλαδή που θέλουν λίγα βήματα επισκευής, για το τέλος.

Με τον ψευδοκώδικα που ακολουθεί περιγράφεται πλήρως η λειτουργία της Most-conflicted-Variable :

Function MIN-CONFLICTS-WR(csp,max_βήματα) επιστρέφει μία λύση ή αποτυχία.

inputs : csp, ένα πρόβλημα ικανοποίησης περιορισμών,
max_βήματα, αριθμός βημάτων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια
max_επανεκκινήσεις, αριθμός επανεκκινήσεων που επιτρέπονται πριν εγκαταλειφθεί η προσπάθεια

for j=1 to max_επανεκκινήσεις **do**
τρέχουσα ← μία πλήρης ανάθεση τιμών για το csp
for i=1 to max_ βήματα **do**
if τρέχουσα είναι λύση για το csp **then return** τρέχουσα

μεταβλητή ← επιλογή της μεταβλητής με τις περισσότερες συγκρούσεις από το VARIABLES[CSP]

τιμή ← μία τιμή u της *μεταβλητής* που ελαχιστοποιεί την CONFLICTS(μεταβλητή,u,τρέχουσα,CSP)

τίθεται μεταβλητή = τιμή στη τρέχουσα

return αποτυχία

5.4.3.Διαχείριση δομών δεδομένων από την Most-Conflicted-Variable

Η διαχείριση των δομών δεδομένων είναι σχεδόν ίδια με τον προηγούμενο αλγόριθμο .Η βασική διαφορά όσον αφορά τις καταστάσεις είναι στην κατάσταση [7] του διαγράμματος του προηγούμενου αλγορίθμου. Στον αλγόριθμο Most-Conflicts-WR στη κατάσταση αυτή γίνεται κάτι διαφορετικό. Επιλέγεται το στοιχείο που έχει την μεγαλύτερη τιμή στον conflicted_subs. Αν δύο στοιχεία έχουν την ίδια τιμή επιλέγεται τυχαία κάποιο από αυτά .

Ο τελευταίος από τους τρεις αλγόριθμους είναι φτιαγμένος με τέτοιο τρόπο ώστε να βρίσκει πιο γρήγορα την λύση .Πρακτικά μπορεί να μην ισχύει πάντα αυτό. Ίσως σε κάποια προβλήματα να μην είναι σωστό να επιλύονται οι μεταβλητές που έχουν τις περισσότερες συγκρούσεις .Υπάρχει περίπτωση βέβαια να μην λυθεί και πάλι το πρόβλημα . Η πιθανότητα να κολλήσει ο αλγόριθμος σε τοπικά μέγιστα είναι ίδια με την Min-Conflicts-WR, δηλαδή αρκετά καλύτερη από την Min-Conflicts .

5.4.4.Υλοποίηση του αλγόριθμου most-conflict-wr στο πρόγραμμα

Εδώ θα δούμε μία παρουσίαση και επεξήγηση του πως υλοποιήθηκε ο αλγόριθμος αυτός μέσα στο πρόγραμμα. Ακολουθεί το κομμάτι του κώδικα που αντιστοιχεί στον αλγόριθμο most-conflicts-wr.

```
do{
      create_List();
      create_Matrix();
      num_of_tries=0;
      do{
            count = check_for_collisions();
            if(count!=0){
                  find_max_collisions();
                  change_conflicted();
            }
            num_of_tries++;
      }while(count!=0 && num_of_tries<lim_of_tries);
      num_of_starts++;
}while(count!=0 && num_of_starts<lim_of_starts);
```

Ο κώδικας για το most-conflicts-wr είναι ο ίδιος με αυτόν για το min-conflicts-wr με τη μοναδική τους διαφορά στη συνάρτηση find_max_collisions η

οποία, σε αντίθεση με την find_collisions, επιστρέφει το μάθημα που προκαλεί τις περισσότερες συγκρούσεις ώστε να αλλάξει η θέση του στο πρόγραμμα. Σε περίπτωση που υπάρχουν περισσότερα από ένα μαθήματα που προκαλούν τις περισσότερες συγκρούσεις τότε επιλέγετε τυχαία ένα από αυτά.

Κεφάλαιο 6 : Αξιολόγηση Αλγορίθμων

6.1 Εισαγωγή

Το βασικό κομμάτι αυτής της μελέτης είναι η αξιολόγηση των αλγορίθμων που περιγράφθηκαν στο 5^{ο} κεφάλαιο. Για να γίνει μία πλήρης αξιολόγηση θα χρησιμοποιηθούν τρία προβλήματα (case studies), διαφορετικής δυσκολίας, ώστε να εξεταστούν σε βάθος οι αλγόριθμοι και να βγουν χρήσιμα συμπεράσματα για επόμενες μελέτες.

Το ενδιαφέρων στην περίπτωση της ανάλυσης απόδοσης των αλγορίθμων είναι ότι ανάλογα με τα χαρακτηριστικά του κάθε προβλήματος η σχετική απόδοση των αλγορίθμων μπορεί να διαφέρει, δηλαδή σε κάποια προβλήματα μπορεί να είναι πιο αποδοτικός ο αλγόριθμος min-conflicts ενώ σε κάποια άλλα μπορεί να είναι ο min-conflicts-wr. Επίσης η μοντελοποίηση μπορεί κι αυτή να επηρεάσει εξαιρετικά την απόδοση. Επομένως πρέπει να γίνει σαφές εδώ ότι τα πειραματικά αποτελέσματα που θα ληφθούν εδώ αφορούν μόνο την συγκεκριμένη μοντελοποίηση. Άρα καλό είναι να οριστούν κάποια κριτήρια περιγραφής των προβλημάτων ώστε να μπορούν να εξαχθούν γενικά συμπεράσματα από την πειραματική μελέτη. Παρακάτω αναλύονται τα κριτήρια αυτά.

6.2 Κριτήρια Αξιολόγησης

6.2.1 Βαθμός επιλυσιμότητας

Το βασικότερο κριτήριο για την δυσκολία των προβλημάτων, εκτός από το μέγεθος τους, είναι το περιθώριο που έχουν για να επιλυθούν ή όπως αλλιώς λέγεται "ο βαθμός επιλυσιμότητας" (feasibility). Ας πάρουμε ένα παράδειγμα για να γίνει κατανοητό αυτό το κριτήριο αξιολόγησης.

Οι ώρες λειτουργίας ενός πανεπιστημίου έχουμε ορίσει πως θα είναι από τις 09:00 έως τις 20:00, δηλαδή έντεκα ώρες την ημέρα για πέντε εργάσιμες μέρες. Άρα συνολικά θα έχουμε 55 ώρες λειτουργία του πανεπιστημίου. Εάν πολλαπλασιάσουμε αυτές τις ώρες με το πλήθος των αιθουσών θα έχουμε ένα μέγεθος, το οποίο θα το ονομάσουμε χρονοθυρίδες, και δεν θα πρέπει να το ξεπερνάει το συνολικό πλήθος των ορών των διδασκαλιών. Δηλαδή εάν έχουμε μία αίθουσα τότε θα έχουμε 55*1=55 χρονοθυρίδες, ή εάν υπάρχουν δύο αίθουσες τότε θα έχουμε 55*2=110 χρονοθυρίδες.

Αν υποθέσουμε ότι σε ένα ακαδημαϊκό εξάμηνο υπάρχουν δεκαπέντε μαθήματα τα οποία χωρίζονται σε δεκαπέντε δίωρα και δεκαπέντε τρίωρα, τότε θα είχαμε συνολικά 75 ώρες μαθημάτων. Οπότε στην περίπτωση όπου έχουμε μόνο μία αίθουσα για να γίνουν αυτά τα μαθήματα το πρόβλημα δεν θα είναι επιλύσιμο, επειδή 75 συνολικά ώρες είναι περισσότερες από τις 55 χρονοθυρίδες. Αντίθετα εάν έχουμε

δύο αίθουσες τότε το πρόβλημα θα είναι επιλύσιμο, επειδή 75 συνολικά ώρες είναι λιγότερες από τις 110 χρονοθυρίδες.

Ο τύπος που θα μας δίνει τον βαθμό επιλυσιμότητας(ΒΕ) θα είναι:

ΒΕ = Χρονοθυρίδες - Συνολικές ώρες

Όπου:

Χρονοθυρίδες = 5 * 11 * Πλήθος αιθουσών

Και

Συνολικές ώρες = (πλήθος δίωρων * 2) + (πλήθος τρίωρων * 3)

Σύμφωνα με τις τιμές που παίρνει το ΒΕ θα έχουμε τρεις πιθανές περιπτώσεις

- Εάν το ΒΕ είναι θετικό τότε το πρόβλημα μας είναι επιλύσιμο, επίσης όσο πιο μεγάλο το ΒΕ τόσο πιο εύκολα λύνεται το πρόβλημα.
- Εάν το ΒΕ είναι μηδέν τότε το πρόβλημα είναι οριακά επιλύσιμο, με κάθε αίθουσα να χρησιμοποιείται κάθε χρονική στιγμή. Άρα όσο πιο κοντά στο μηδέν βρίσκεται το ΒΕ τόσο πιο δύσκολο το πρόβλημα.
- Εάν το ΒΕ είναι αρνητικό τότε αυτό σημαίνει πως το πρόβλημα είναι μη επιλύσιμο.

Στην περίπτωση του προβλήματος μας, επειδή οι αίθουσες χωρίζονται σε δύο είδη τις θεωρητικές και τα εργαστήρια, θα έχουμε δύο διαφορετικούς βαθμούς επιλυσιμότητας. Οπότε θα έχουμε βαθμό επιλυσιμότητας για την θεωρία (ΒΕ_Θ) και βαθμό επιλυσιμότητας για τα εργαστήρια (ΒΕ_Γ).

Ο βαθμός επιλυσιμότητας, όπως τον ορίσαμε, δεν μας δίνει μια πλήρη εικόνα για το πόσο εύκολο είναι ένα πρόβλημα επειδή υπάρχουν και άλλοι παράγοντες οι οποίοι το επηρεάζουν, όπως είναι το πλήθος των μαθημάτων σε σχέση με τους καθηγητές και τα εξάμηνα, αλλά μας δίνει μια αρκετά ξεκάθαρη εικόνα για την δυσκολία του προβλήματος.

Ξέρουμε πως στο σύνολο των περιορισμών υπάρχουν δύο οι οποίοι μας περιορίζουν το όριο των διδασκαλιών ανά ημέρα, και είναι δύο για τους καθηγητές και τέσσερα για τα εξάμηνα. Από αυτό καταλαβαίνουμε πως υπάρχει κατά προέκταση ένα όριο διδασκαλιών ανά εβδομάδα για τους καθηγητές και για τα εξάμηνο, τα οποία θα είναι 10 και 20 αντίστοιχα. Για παράδειγμα εάν ένας καθηγητής είναι υπεύθυνος για 11 διδασκαλίες τότε αυτές οι έντεκα δεν θα είναι εφικτό να μοιραστούν μέσα στο Ωρολόγιο Πρόγραμμα με αποτέλεσμα να είναι αδύνατο να βρεθεί λύση. Έτσι λοιπόν όταν θα δούμε τα προβλήματα για τις προσομοιώσεις θα δείξουμε αρκετή προσοχή στο να μην ξεπερνάμε αυτά τα όριο, αλλά και σε ποια προβλήματα τα έχουμε πλησιάσει.

Όπως είπαμε και στην αρχή ο βαθμός επιλυσιμότητας μπορεί να είναι το σημαντικότερο κριτήριο αλλά δεν είναι το μοναδικό. Ένα ακόμα αρκετά σημαντικό κριτήριο είναι το μέγεθος του προβλήματος, δηλαδή το πόσες συνολικά ώρες

διδασκαλιών έχουμε. Για παράδειγμα ένα πρόβλημα με 10 μαθήματα και ένα δεύτερο με 20 μαθήματα, τα οποία έχουν τον ίδιο βαθμό επιλυσιμότητας δεν σημαίνει πως είναι το ίδιο εύκολα, αλλά λόγω του μεγαλύτερο πλήθους μεταβλητών του δεύτερου προβλήματος το κάνει δυσκολότερο από το πρώτο.

6.2.2 Υπόλοιπα κριτήρια αξιολόγησης

Υπάρχουν και τα υπόλοιπα κριτήρια αξιολόγησης τα οποία δεν αναφέρονται στην δυσκολία των προβλημάτων, αλλά χρησιμεύουν στην αξιολόγηση των αλγορίθμων. Για να έχουμε μια πιο καθαρή εικόνα για το ποιος αλγόριθμος είναι αποδοτικότερος θα χρησιμοποιήσουμε τα παρακάτω κριτήρια τα οποία θα βγουν κατά μέσω όρο από την πολλαπλή προσομοίωση των αλγορίθμων, και αυτά τα κριτήρια είναι:

- Ποσοστό επιτυχίας: το οποίο είναι %, και μα δείχνει κατά πόσο ο αλγόριθμος βρίσκει λύση.
- Μέσος χρόνος λύσης: Για τις περιπτώσεις στις οποίες ο αλγόριθμος βρήκε λύση , πόσος χρόνος χρειάστηκε σε ms.
- Μέσο πλήθος βημάτων-επανεκκινήσεων : Για όσες περιπτώσεις στις οποίες ο αλγόριθμος βρήκε λύση , σε ποιο βήμα επισκευής και σε ποιο βήμα επανεκκίνησης βρέθηκε η λύσης αυτή (π.χ. 12-3 ,δηλαδή στο 12^{o} βήμα επισκευής – 3^{o} βήμα επανεκκίνησης) .

Οι προσομοιώσεις των αλγορίθμων θα γίνονται εκατό φορές για τον κάθε έναν, οπότε το ποσοστό επιτυχίας θα μας λέει πόσες από τις εκατό αυτές φορές ο αλγόριθμος επέστρεψε λύση. Το ποσοστό δεν θα φτάνει ποτέ το 100%, στην καλύτερη περίπτωση το 99%. Με το ποσοστό θα γίνεται κατανοητό σε ποιες περιπτώσεις θα μπορεί να εφαρμοστεί επιτυχώς ο αλγόριθμος.

Ο μέσος χρόνος λύσης υπολογίζεται από τον χρόνο που χρειάζεται μία επιτυχής προσομοίωση, δηλαδή από την στιγμή όπου θα αρχικοποιηθεί ο πίνακας Matrix έως τη στιγμή όπου θα επιστραφεί λύση. Είναι αρκετά σημαντικό να ξέρουμε ποιός αλγόριθμος επιστρέφει πιο γρήγορα λύση. Ο μέσος χρόνος υπολογίζεται σε ms.

Το μέσο πλήθος βημάτων-επανεκκινήσεων υπολογίζεται από τα βήματα και τις επανεκκινήσεις που έγιναν σε κάθε επιτυχής προσομοίωση. Οπότε στον min-conflicts θα έχουμε μόνο τον μέσω όρο βημάτων ενώ στους άλλους δύο αλγορίθμους θα έχουμε και τον μέσω όρο επανεκκινήσεων. Για παράδειγμα μπορεί να έχουμε 12,5-3,1 δηλαδή 12,5 βήματα και 3,1 επανεκκινήσεις κατά μέσο όρο. Έτσι σε συνδυασμό με τον μέσο χρόνο θα έχουμε μια σαφή εικόνα για την ταχύτητα των βημάτων των αλγορίθμων.

6.3 Πειραματικά Αποτελέσματα

6.3.1 Εύκολο Πρόβλημα

Το πρώτο πρόβλημα θα είναι το πιο εύκολο για να λυθεί, επειδή αποτελείτε από 21 μαθήματα χωρισμένα σε 47 διδασκαλίες, μοιρασμένα σε τρία διαφορετικά εξάμηνα και 12 καθηγητές. Το μέγεθος του προβλήματος θα γίνει κατανοητό, συγκριτικά με τα επόμενα προβλήματα, πως είναι αρκετά μικρό.

Οι συνολικές ώρες των διδασκαλιών για την θεωρία είναι 80 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχουν δύο ή περισσότερες αίθουσες θεωρίας, δίνοντας μας έτσι BE_Θ=(2*55)-80 = 30, άρα από την πλευρά της θεωρίας το πρόβλημα είναι επιλύσιμο. Οι συνολικές ώρες των διδασκαλιών για τα εργαστήρια είναι 40 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχει μια ή περισσότερες αίθουσες εργαστηρίου, δίνοντας μας έτσι BE_E=(1*55)-40 = 15, άρα και από την πλευρά του εργαστηρίου το πρόβλημα είναι επιλύσιμο.

Όπως αναφέραμε υπάρχουν και άλλοι παράγοντες για να είναι το πρόβλημα επιλύσιμο. Σε αυτό το πρόβλημα έχουμε στην από 2 έως 5 διδασκαλίες για κάθε καθηγητή, και για τα εξάμηνα έχουμε 14 έως 17 διδασκαλίες.

Ακολουθεί ο πίνακας των διδασκαλιών του προβλήματος:

Αρ. Μαθήματος (sub_id)	Καθηγητής (prof)	Εξάμηνο (sem)	Είδος Μαθήματος (kind)	Χρόνος (hours)
0	1	1	0	1
1	2	1	0	0
1	2	1	1	0
2	3	1	0	1
2	3	1	1	0
3	4	1	0	0
3	4	1	0	1
3	4	1	1	0
4	5	1	0	1
4	5	1	1	1
5	1	1	0	0
5	1	1	1	1
6	6	1	0	1
6	6	1	0	0
7	2	3	0	1
7	2	3	1	0
7	2	3	1	0
8	7	3	0	0
8	7	3	0	1
9	8	3	0	1
9	8	3	1	0

10	4	3	0	1
10	4	3	0	1
11	9	3	0	1
11	9	3	0	0
11	9	3	1	1
12	10	3	0	0
12	10	3	1	1
13	6	3	0	0
13	6	3	1	0
14	7	5	0	0
14	7	5	0	1
15	3	5	0	1
15	3	5	1	0
16	11	5	0	0
16	11	5	1	0
17	12	5	0	1
17	12	5	1	1
17	12	5	1	1
18	10	5	0	1
18	10	5	0	1
19	5	5	0	0
19	5	5	0	1
19	5	5	1	0
20	1	5	0	1
21	8	5	0	0
21	8	5	1	1

Πίνακας 6.1 : Μορφή της εισόδου του εύκολου προβλήματος.

Για τα αποτελέσματα που ακολουθούν έγιναν 100 προσομοιώσεις για κάθε αλγόριθμο και υπολογίσθηκαν οι μέσοι όροι. Επίσης θα έχουμε αποτελέσματα και από τις δύο διαφορετικές αρχικοποιήσεις όπως αναφέραμε στο 4^{ο} Κεφάλαιο.

Ακολουθεί ο πίνακας των αποτελεσμάτων:

Αλγόριθμος	Όριο Βημάτων	Όριο Επαναλήψεων	Αίθουσες Θεωρίας-Εργαστηρίου	ΒΕ-Θ / ΒΕ-Ε	Ποσοστό Επιτυχίας	Μέσος Χρόνος Λύσης	Μέσο πλήθος βημάτων-επανεκκινήσεων
MIN-CONFLICTS	40	0	3 - 2	85 / 70	99%	22.98	28.34
	35	0	3 - 2	85 / 70	95%	22.65	28.37
	30	0	3 - 2	85 / 70	79%	16.74	26.69
	60	0	2 - 1	30 / 15	78%	21.46	47.85
	70	0	2 - 1	30 / 15	96%	22.64	49.57

MIN-CONFLICTS-GS	20	0	3 - 2	85 / 70	99%	612.48	4.24
	30	0	2 - 1	30 / 15	77%	415.15	13.33
MIN-CONFLICTS-WR	30	5	3 - 2	85 / 70	99,5%	24.1	27.37 - 0.38
	28	4	3 - 2	85 / 70	94%	32.27	26.25 - 0.58
	26	4	3 - 2	85 / 70	77%	43	25.27 - 1.03
	50	5	2 - 1	30 / 15	99%	49.56	43.13 - 1.939
	45	4	2 - 1	30 / 15	81%	35.44	40.85 - 1.83
	40	4	2 - 1	30 / 15	69%	42.39	38.35 - 2.347
MIN-CONFLICTS-WR-GS	5	2	3 - 2	85 / 70	99.5%	716.43	2.95 - 0.16
	10	4	2 - 1	30 / 15	74%	867.05	7.51 - 1.10
MOST-CONFLICTS-WR	25	5	3 - 2	85 / 70	99,5%	25.58	23.25 - 1.515
	25	4	3 - 2	85 / 70	99%	21.63	23.21 - 0.47
	23	4	3 - 2	85 / 70	85%	36.54	22.14 - 1.142
	40	5	2 - 1	30 / 15	98%	28.75	35.31 - 0.73
	40	4	2 - 1	30 / 15	95%	28.55	35.61 - 0.705
	35	4	2 - 1	30 / 15	81%	34.62	32.85 - 2.26
MOST-CONFLICTS-WR-GS	5	2	3 - 2	85 / 70	95%	734.71	3.063 - 0.18
	10	4	2 - 1	30 / 15	76%	766.88	7.166 - 0.855

Πίνακας 6.2 : Αποτελέσματα των τριών αλγορίθμων για το εύκολο πρόβλημα.

Παρατηρήσεις για τα αποτελέσματα:

- Για τον Min-Conflicts βλέπουμε πως με το κατάλληλο πλήθος βημάτων βρίσκει σχεδόν πάντα λύση. Όταν έχουμε 3 – 2 δωμάτια τότε με όριο βημάτων 40 ή και 35 βλέπουμε πως ο αλγόριθμος είναι πολύ αποδοτικός. Αλλά παρατηρούμε πως όταν πλησιάσουμε των μέσο όρο

βημάτων, ο οποίος είναι 28.3, και βάλουμε όριο τα 30 βήματα τότε ο αλγόριθμος έχει μία αρκετά μεγάλη πτώση στο ποσοστό επιτυχίας στο 77%. Αντίστοιχα όταν έχουμε 2 -1 δωμάτια παρατηρήθηκε πως χρειάστηκαν αρκετά περισσότερα βήματα, με το όριο να τίθεται στα 70 και 60. Είναι πολύ ενδιαφέρον να παρατηρήσουμε πως το μέσο πλήθος βημάτων είναι αρκετά μικρότερο από το όριο βημάτων, το οποίο συμβαίνει επειδή όταν έχουμε ένα αρκετά μικρό Βαθμό Επιλυσιμότητας ο αλγόριθμος είναι εύκολο να κολλήσει σε ένα τοπικό βέλτιστο και λόγω της τυχαιότητας τον κινήσεων να χρειάζεται αρκετά βήματα για να ξεκολλήσει. Τέλος βλέπουμε πως ο αλγόριθμος είναι πολύ γρήγορος και χρειάζεται μόνο 22 ms για την εκτέλεση του.

Όταν στις προσομοιώσεις χρησιμοποιήσουμε την άπληστη αρχικοποίηση, όπου στον πίνακα είναι τα δύο τελευταία αποτελέσματα για τον Min-Conflicts-GS (GS = greedy start, δηλ. άπληστη αρχικοποίηση), παρατηρούμε πως το μέσο πλήθος βημάτων που γίνονται είναι πολύ μικρότερο από όταν έχουμε τυχαία αρχικοποίηση, και επειδή το πρόβλημα είναι αρκετά εύκολο σε μερικές περιπτώσεις είχαμε λύση του προβλήματος από την αρχικοποίηση. Όταν το πρόβλημα έχει έναν αρκετά μεγάλο βαθμό επιλυσιμότητας ο min-conflicts με άπληστη αρχικοποίηση είναι πολύ αποδοτικός αλλά ‘όταν το BE μικραίνει τότε το ποσοστό επιτυχία του μικραίνει αρκετά. Το μεγάλο μειονέκτημα του αλγορίθμου αυτού είναι πως ο μέσος χρόνος λύσης είναι πολύ μεγαλύτερος, το οποίο συμβαίνει επειδή η άπληστη αρχικοποίηση είναι πολύ χρονοβόρα σε σχέση με την τυχαία.

- Για τον αλγόριθμο Min-Conflicts-WR στις προσομοιώσεις πήραμε το όριο των βημάτων να είναι κοντά στον μέσο πλήθος βημάτων που βρήκαμε στην πιο επιτυχημένη εκτέλεση του min-conflicts. Αφού λοιπόν στα παραδείγματα με 3 – 2 δωμάτια είχαμε μέσο πλήθος βημάτων 28.3 και με 2 – 1 δωμάτια είχαμε μέσο πλήθος βημάτων περίπου 48.5 βήματα, τότε θα για 3 - 2 δωμάτια πήραμε από 30 έως 26 βήματα και για τα 2 – 1 δωμάτια 50 έως 45 βήματα.

  Τα αποτελέσματα μας δείχνουν πως ο Min-Conflicts-WR είναι πολύ αποτελεσματικός με ένα σχετικά χαμηλό όριο βημάτων σε συνδυασμό με τις επαναλήψεις, φτάνοντας έως και 99,5% στην καλύτερη του περίπτωση για το πρόβλημα με 3 – 2 αίθουσες και 99% για 2 – 1 αίθουσες. Αυτό είναι πολύ ενδιαφέρον γιατί με το ίδιο όριο βημάτων ο Min-Conflicts είναι πολύ λιγότερο αποδοτικός από τον Min-Conflicts-WR, έτσι βλέπουμε την σημασία των επανεκκινήσεων. Υπάρχει όμως ένα μειονέκτημα και αυτό είναι πως όσο μεγαλύτερο είναι το μέσο πλήθος επανεκκινήσεων τόσο μεγαλύτερος είναι και ο μέσος χρόνος και αυτό γιατί η ανάθεση τιμών στη αρχικοποίηση είναι χρονοβόρα.

  Στην περίπτωση που θα χρησιμοποιήσουμε την άπληστη αρχικοποίηση βλέπουμε πως τα βήματα που χρειάζονται είναι πολύ λίγα, για παράδειγμα στην περίπτωση με τις 3 – 2 αίθουσες έχουμε μέσο πλήθος βημάτων 1,95, το οποίο είναι πολύ μικρό. Επίσης βλέπουμε πως σαν αλγόριθμος είναι και αυτός αρκετά αποδοτικός στην περίπτωση του εύκολου προβλήματος. Παρ’ όλα αυτά ο μέσος

χρόνος εκτέλεσης είναι πάλι πολύ υψηλός, καθώς σε μερικές περιπτώσεις χρειάζονται επανεκκινήσεις, με αποτέλεσμα να ξεπερνάει το ένα δευτερόλεπτο σε κάποιες προσομοιώσεις και τον μέσο χρόνο να βρίσκεται στα 867.05 ms στην περίπτωση των 2 – 1 αιθουσών.

- Όπως ήταν αναμενόμενο από τον αλγόριθμο Most-Conflicts-WR βλέπουμε πως το μέσω πλήθος βημάτων που κάνει είναι μικρότερο από αυτό των δύο προηγούμενων αλγορίθμων, επειδή όπως ξέρουμε ο Most-Conflicts-WR είναι ο πιο γρήγορος στο να βρει λύση. Αυτό δεν φαίνεται μόνο από το μέσω πλήθος βημάτων αλλά και από τον μέσο χρόνο λύσης του προβλήματος ο οποίος είναι μικρότερος από τους δύο προηγούμενους αλγορίθμους.

  Όταν χρησιμοποιούμε την άπληστη αρχικοποίηση στα ίδια παραδείγματα με αυτά που χρησιμοποιήσαμε στον προηγούμενο αλγόριθμο (Min-Conflicts-WR-GS) παρατηρούμε πως όταν έχουμε την πιο εύκολη περίπτωση με τις 3 – 2 αίθουσες το ποσοστό επιτυχία είναι μικρότερο για τον Most-Conflicts-WR-GS, το οποίο είναι λογικό καθώς και η άπληστη αρχικοποίηση και ο Most-Conflicts οδηγούνται εύκολα σε τοπικά βέλτιστα. Αλλά όταν έχουμε την δυσκολότερη περίπτωση των 2 – 1 αιθουσών τα αποτελέσματα είναι καλύτερα.

  Τέλος συμπεραίνουμε πως ο αλγόριθμος είναι πολύ αποδοτικός και σε συνδυασμό με το ότι είναι ο ταχύτερος, σε χρόνο και βήματα, μπορούμε να πούμε πως είναι και ο καταλληλότερος για την επίλυση του εύκολου προβλήματος.

6.3.2 Μέτριο Πρόβλημα

Για το μέτρο πρόβλημα θα προσθέσουμε ένα ακόμα εξάμηνο, θα επιβαρύνουμε περισσότερο τα εξάμηνο και τους καθηγητές και θα έχουμε περισσότερες αίθουσες.

Το πρώτο πρόβλημα είναι μέτριας δυσκολίας, και αποτελείτε από 29 μαθήματα χωρισμένα σε 69 διδασκαλίες, μοιρασμένα σε τέσσερα διαφορετικά εξάμηνα και 15 καθηγητές.

Οι συνολικές ώρες των διδασκαλιών για την θεωρία είναι 112 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχουν τρεις ή περισσότερες αίθουσες θεωρίας, δίνοντας μας έτσι BE_Θ=(3*55)-112 = 53, άρα από την πλευρά της θεωρίας το πρόβλημα είναι επιλύσιμο. Οι συνολικές ώρες των διδασκαλιών για τα εργαστήρια είναι 62 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχει δύο ή περισσότερες αίθουσες εργαστηρίου, δίνοντας μας έτσι BE_E=(2*55)-62 = 48, άρα και από την πλευρά του εργαστηρίου το πρόβλημα είναι επιλύσιμο.

Σε αυτό το πρόβλημα έχουμε στην από 2 έως 7 διδασκαλίες για κάθε καθηγητή, και για τα εξάμηνα έχουμε 17 έως 18 διδασκαλίες.

Ακολουθεί ο πίνακας των διδασκαλιών του προβλήματος:

Αρ. Μαθήματος (sub_id)	Καθηγητής (prof)	Εξάμηνο (sem)	Είδος Μαθήματος (kind)	Χρόνος (hours)
0	1	1	0	1
1	2	1	0	0
1	2	1	1	0
2	3	1	0	1
2	3	1	1	0
3	4	1	0	0
3	4	1	0	1
3	4	1	1	0
4	5	1	0	1
4	5	1	1	1
5	1	1	0	0
5	1	1	1	1
6	6	1	0	1
6	6	1	0	0
7	2	1	0	1
7	2	1	1	0
7	2	1	1	0
8	7	3	0	0
8	7	3	0	1
9	8	3	0	1
9	8	3	1	0
10	4	3	0	1
10	4	3	0	1
11	9	3	0	1
11	9	3	0	0
11	9	3	1	1
12	10	3	0	0
12	10	3	1	1
13	6	3	0	0
13	6	3	1	0
14	7	3	0	0
14	7	3	0	1
15	3	3	0	1
15	3	3	1	0
16	11	5	0	0
16	11	5	1	0
17	12	5	0	1
17	12	5	1	1
17	12	5	1	1
18	10	5	0	1
18	10	5	0	1
19	5	5	0	0
19	5	5	0	1

19	5	5	1	0
20	1	5	0	1
21	8	5	0	0
21	8	5	1	1
22	13	5	0	1
22	13	5	0	1
22	13	5	1	0
22	13	5	1	0
23	1	7	0	0
23	1	7	1	0
24	14	7	0	1
24	14	7	0	1
24	14	7	1	0
24	14	7	1	0
25	3	7	0	0
25	3	7	0	1
25	3	7	1	0
26	5	7	0	0
26	5	7	0	1
27	6	7	0	1
27	6	7	1	1
28	15	7	0	0
28	15	7	1	0
28	15	7	1	1
29	7	7	0	1
29	7	7	1	1

Πίνακας 6.3 : Μορφή της εισόδου του μέτριου προβλήματος.

Όπως και στο προηγούμενο πρόβλημα για τα αποτελέσματα που ακολουθούν έγιναν 100 προσομοιώσεις για κάθε αλγόριθμο και υπολογίσθηκαν οι μέσοι όροι. Επίσης θα έχουμε αποτελέσματα και από τις δύο διαφορετικές αρχικοποιήσεις όπως αναφέραμε στο 4^{ο} Κεφάλαιο.

Ακολουθεί ο πίνακας των αποτελεσμάτων:

Αλγόριθμος	Όριο Βημάτων	Όριο Επαναλήψεων	Αίθουσες Θεωρίας-Εργαστηρίου	ΒΕ-Θ / ΒΕ-Ε	Ποσοστό Επιτυχίας	Μέσος Χρόνος Λύσης	Μέσο πλήθος βημάτων-επανεκκινήσεων
MIN-CONFLICTS	70	0	4 - 3	108 / 103	99%	59.87	48.424
	60	0	4 - 3	108 / 103	96%	69.68	47.03
	50	0	4 - 3	108 / 103	69%	65.202	44.695
	90	0	3 - 2	53 / 48	98%	72.01	59.08

	80	0	3 - 2	53 / 48	92%	57.02	55.48
	65	0	3 - 2	53 / 48	82%	56.75	54.85
MIN-CONFLICTS-GS	50	0	4 - 3	108 / 103	99%	2631.58	10.16
	65	0	3 - 2	53 / 48	90%	2202.08	16.42
MIN-CONFLICTS-WR	50	5	4 - 3	108 / 103	99%	103.12	44.18 - 0.494
	45	5	4 - 3	108 / 103	96%	130.104	43.373 - 1.36
	45	4	4 - 3	108 / 103	84%	130.881	43.41 - 1.107
	60	5	3 - 2	53 / 48	99%	103.283	54.19 - 0.535
	55	5	3 - 2	53 / 48	91%	112.52	51.21 - 1.065
	50	4	3 - 2	53 / 48	63%	139.98	49.302 - 1.41
MIN-CONFLICTS-WR-GS	30	2	4 - 3	108 / 103	99%	2809.15	10.36 – 0.05
	30	4	3 - 2	53 / 48	99%	2549.68	13.708 - 0.303
MOST-CONFLICTS-WR	40	5	4 - 3	108 / 103	99%	80.212	36.96 - 0.717
	40	2	4 - 3	108 / 103	85%	55.94	36.68 - 0.22
	35	5	4 - 3	108 / 103	70%	97.78	33.81 - 1.27
	45	5	3 - 2	53 / 48	99.5%	89.98	42.21 - 1.01
	50	2	3 - 2	53 / 48	90%	69.22	43.64 - 0.266
	40	10	3 - 2	53 / 48	92%	145.07	38.95 - 2.55
MOST-CONFLICTS-WR-GS	28	2	4 - 3	108 / 103	99%	2850.62	10.45 – 0,11
	30	2	3 - 2	53 / 48	95%	2709.36	14.20 - 0.234

Πίνακας 6.4 : Αποτελέσματα των τριών αλγορίθμων για το μέτριο πρόβλημα.

Παρατηρήσεις για τα αποτελέσματα:

- Αν και το πρόβλημα έγινε αρκετά μεγαλύτερο και πιο περίπλοκο φαίνεται πως ο αλγόριθμος Min-Conflicts είναι αρκετά αποδοτικός, αν και χρειάζεται ένα μεγάλο πλήθος βημάτων για να πετύχει το μέγιστο ποσοστό επιτυχίας. Επίσης παρατηρούμε την διαφορά στον μέσο χρόνο που χρειάζεται για την λύση αυτού του προβλήματος ο αλγόριθμος σε σχέση με το προηγούμενο. Όπως είναι λογικό όσο πλησιάζουμε το μέσο πλήθος βημάτων τόσο λιγότερο αποτελεσματικό είναι ο αλγόριθμος.

  Όταν στις προσομοιώσεις χρησιμοποιήσουμε την άπληστη αρχικοποίηση, όπου το όριο βημάτων είναι το ίδιο με το μικρότερο που βάλαμε στις προσομοιώσεις με τυχαία αρχικοποίηση, μπορούμε να παρατηρήσουμε ότι είναι πολύ πιο αποτελεσματικός ο αλγόριθμος και χρειάζεται πολύ λιγότερα βήματα. Για παράδειγμα όταν έχουμε 4 - 3 αίθουσες και το όριο βημάτων είναι 65, με τυχαία αρχικοποίηση ο αλγόριθμος είναι 69% επιτυχής και χρειάζεται κατά μέσο όρο 44.495 βήματα, ενώ με άπληστη αρχικοποίηση είναι 99% επιτυχής και χρειάζεται κατά μέσο όρο 10.16 βήματα. Αλλά όπως ήταν αναμενόμενο ο μέσος χρόνος για να βρει λύση είναι πολύ μεγάλος στα 2631.58.

- Για τον αλγόριθμο Min-Conflicts-WR όπως και στο προηγούμενο πρόβλημα μπορούμε να θέσουμε ένα μικρότερο όριο βημάτων από αυτό για τον Min-Conflicts, επειδή έχουμε τις επανεκκινήσεις. Παρατηρούμε πως και αυτός ο αλγόριθμος είναι πολύ αποδοτικός αλλά ο μέσος χρόνος για να βρει λύση είναι σχεδόν διπλάσιος από αυτόν που χρειάζεται ο Min-Conflicts. Με την άπληστη αρχικοποίηση έχουμε πάλι πολύ καλά αποτελέσματα με μικρότερα όρια, καθώς και αρκετά μεγάλο χρόνο για να βρεθεί λύση

- Για τον αλγόριθμο Most-Conflicts-WR βλέπουμε και πάλι πως είναι πολύ αποδοτικός και χρειάζεται μικρότερα όρια βημάτων και επανεκκινήσεων. Είναι πολύ ενδιαφέρον να παρατηρήσουμε πως στη περίπτωση που έχουμε 4 - 3 αίθουσες, ο αλγόριθμος είναι πιο αποδοτικός με τα όρια των βημάτων/επανεκκινήσεων να είναι 40/5 με ποσοστό επιτυχίας 99%, και βλέπουμε πως όταν μειώσουμε το όριο επανεκκινήσεων στις 2 το ποσοστό πέφτει στο 85% αλλά όταν μειώσουμε το όριο των βημάτων στα 35 το ποσοστό πέφτει στο 70%. Οπότε καταλαβαίνουμε πως για τον Most-Conflicts-WR είναι αρκετά πιο σημαντικό να έχει ένα μεγάλο όριο βημάτων παρά επανεκκινήσεων.

  Για το πρόβλημα με 3 – 2 αίθουσες ο αλγόριθμος είναι πιο αποδοτικός με τα όρια των βημάτων/επανεκκινήσεων να είναι 45/5 με ποσοστό επιτυχίας 99,5%. Στις δύο επόμενες προσομοιώσεις είναι ενδιαφέρον να δούμε την ισορροπία μεταξύ των δύο ορίων, στις οποίες έχουμε περίπου το ίδιο ποσοστό επιτυχίας, με την πρώτη να έχει μεγάλο όριο βημάτων και μικρό όριο επανεκκινήσεων στα 50/2 και την δεύτερη το αντίθετο 40/10. Όπως είναι αναμενόμενο η

προσομοίωση με το μεγάλο πλήθος επανεκκινήσεων έχει και το μεγαλύτερο μέσο χρόνο λύσης.

Χρησιμοποιώντας την άπληστη αρχικοποίηση ακόμα και στο μέτριο πρόβλημα κατά τις ξεχωριστές προσομοιώσεις υπήρχαν περιπτώσεις όπου το πρόβλημα ήταν λυμένο από την αρχικοποίηση. Για την άπληστη αρχικοποίηση αποφεύγουμε να βάζουμε μεγάλο όριο επανεκκινήσεων επειδή τότε θα έκανε τον μέσο χρόνο επίλυσης πολύ μεγάλο, για αυτό στις προσομοιώσεις παίρνουμε το πολύ όριο 3.

6.3.3 Δύσκολο Πρόβλημα

Το τελευταίο και δυσκολότερο πρόβλημα θα είναι και το πιο ρεαλιστικό από άποψη πλήθους μαθημάτων, με πολύ βεβαρυμμένο πρόγραμμα για τα εξάμηνα και τους καθηγητές. Θα προσθέσουμε ένα ακόμα εξάμηνο, περισσότερα μαθήματα ανά καθηγητή και εξάμηνο ώστε να πλησιάσουμε περισσότερο τα όρια, και θα επιδιώξουμε να έχουμε ένα όσο το δυνατών μικρότερο BE.

Στο πρόβλημα αυτό θα έχουμε 39 μαθήματα χωρισμένα σε 98 διδασκαλίες. Από τα πέντε εξάμηνα τα δύο θα έχουν 19 διδασκαλίες ενώ τα υπόλοιπα τρία θα έχουν 20, δηλαδή θα βρίσκονται στο όριο. Στους καθηγητές υπάρχει ένα καθηγητή ο οποίος βρίσκεται στο όριο των 10 διδασκαλιών, και οι υπόλοιποι έχουν ένα μεγάλο πλήθος με το μικρότερο να είναι στις 5 διδασκαλίες.

Οι συνολικές ώρες των διδασκαλιών για την θεωρία είναι 156 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχουν τρεις ή περισσότερες αίθουσες θεωρίας, δίνοντας μας έτσι BE_Θ=(3*55)-156 = 9, άρα από την πλευρά της θεωρίας το πρόβλημα είναι επιλύσιμο αλλά σχεδόν στο όριο. Οι συνολικές ώρες των διδασκαλιών για τα εργαστήρια είναι 95 και για να είναι επιλύσιμο το πρόβλημα χρειαζόμαστε περισσότερες χρονοθυρίδες, οπότε θα υποθέσουμε ότι υπάρχει δύο ή περισσότερες αίθουσες εργαστηρίου, δίνοντας μας έτσι BE_E=(2*55)-95 = 15, άρα και από την πλευρά του εργαστηρίου το πρόβλημα είναι επιλύσιμο.

Ακολουθεί ο πίνακας των διδασκαλιών του προβλήματος:

Αρ. Μαθήματος (sub_id)	Καθηγητής (prof)	Εξάμηνο (sem)	Είδος Μαθήματος (kind)	Χρόνος (hours)
0	1	1	0	1
1	2	1	0	0
1	2	1	1	0
2	3	1	0	1
2	3	1	1	0
3	4	1	0	0
3	4	1	0	1
3	4	1	1	0

4	5	1	0	1
4	5	1	1	1
5	1	1	0	0
5	1	1	1	1
6	6	1	0	1
6	6	1	0	0
7	2	1	0	1
7	2	1	1	0
7	2	1	1	0
8	7	1	0	0
8	7	1	0	1
9	8	3	0	1
9	8	3	0	1
9	8	3	1	0
10	4	3	0	1
10	4	3	0	1
11	9	3	0	1
11	9	3	0	0
11	9	3	1	1
12	10	3	0	0
12	10	3	0	1
12	10	3	1	1
13	6	3	0	0
13	6	3	1	0
14	7	3	0	0
14	7	3	0	1
15	3	3	0	1
15	3	3	1	0
16	11	3	0	0
16	11	3	1	1
16	11	3	1	0
17	12	5	0	1
17	12	5	1	1
17	12	5	1	1
18	10	5	0	1
18	10	5	0	1
19	5	5	0	0
19	5	5	0	1
19	5	5	1	0
20	1	5	0	1
21	8	5	0	0
21	8	5	1	1
22	13	5	0	1
22	13	5	0	1
22	13	5	1	0
	13	5	1	0

22	14	5	0	1
23	14	5	0	1
23	14	5	1	0
23	14	5	1	0
23	1	7	0	0
24	1	7	1	0
24	3	7	0	0
25	3	7	0	1
25	3	7	1	0
25	5	7	0	0
26	5	7	0	1
26	6	7	0	1
27	6	7	1	1
27	15	7	0	0
28	15	7	1	0
28	15	7	1	1
28	7	7	0	1
29	7	7	1	1
29	4	7	0	1
30	4	7	1	0
30	4	7	1	0
30	11	7	0	0
31	11	7	0	1
31	11	7	1	1
31	13	9	0	1
32	13	9	0	0
32	13	9	1	0
32	13	9	1	1
32	13	9	1	1
32	15	9	0	1
33	15	9	0	1
33	7	9	0	0
34	7	9	1	0
34	7	9	1	1
34	15	9	0	0
35	15	9	0	1
35	5	9	0	1
36	5	9	0	1
36	5	9	1	0
36	12	9	0	0
37	12	9	1	1
37	2	9	0	1
38	2	9	1	1
38	1	9	0	1
39				

Πίνακας 6.5 : Μορφή της εισόδου του δύσκολου προβλήματος.

Επειδή το πρόβλημα αυτό είναι το σημαντικότερο τα αποτελέσματα των τριών αλγορίθμων δεν θα βρίσκονται στον ίδιο πίνακα αλλά θα έχει ο καθένα τον δικό του επειδή θα είναι ποιο ενδιαφέρων να δούμε τα αποτελέσματα από ένα μεγαλύτερο εύρος προσομοιώσεων από αυτό που είχαμε στα προηγούμενα, πιο εύκολα, προβλήματα.

Για να έχουμε μία καλύτερη εικόνα της απόδοσης των αλγορίθμων σε κάθε προσομοίωση κρατήσαμε σταθερό τον αριθμό των αιθουσών με 3 θεωρίας και 2 εργαστηρίου. Έτσι όχι μόνο είχαμε ένα μεγαλύτερο εύρος πειραματικών αποτελεσμάτων σε ένα πιο σταθερό πρόβλημα, αλλά διατηρήσαμε και τη δυσκολία του προβλήματος σταθερή με βαθμό επιλυσιμότητας για τις αίθουσες θεωρίας 9, και για τις αίθουσες εργαστηρίου 15, οι οποίοι είναι αρκετά μικροί συγκριτικά με το μέγεθος του προβλήματος.

Ακολουθεί ο πίνακας των αποτελεσμάτων για τον Min-Conflicts:

Αλγόριθμος	Όριο Βημάτων	Όριο Επαναλήψεων	Αίθουσες Θεωρίας-Εργαστηρίου	ΒΕ-Θ / ΒΕ-Ε	Ποσοστό Επιτυχίας	Μέσος Χρόνος Λύσης	Μέσο πλήθος βημάτων-επανεκκινήσεων
MIN-CONFLICTS	500	0	3 - 2	9 / 15	15%	454.5	285.375
	700	0	3 - 2	9 / 15	20%	566.75	355.1
	1000	0	3 - 2	9 / 15	25%	758.063	477.96
	1200	0	3 - 2	9 / 15	27%	811.63	440.93
	1500	0	3 - 2	9 / 15	29%	719.31	454.53
	2000	0	3 - 2	9 / 15	30%	766.8	484.63
MIN-CONFLICTS-GS	200	0	3 - 2	9 / 15	1%	5287	68

Πίνακας 6.6 : Αποτελέσματα του min-conflicts για το δύσκολο πρόβλημα.

Παρατηρήσεις για τα αποτελέσματα του αλγορίθμου Min-Conflicts:

Η πρώτη παρατήρηση για τον αλγόριθμο είναι το μεγάλο όριο βημάτων που χρειάζεται. Συγκρίνοντάς το με το όριο των βημάτων που χρησιμοποιήσαμε για το μέτριο πρόβλημα βλέπουμε μια πολύ μεγάλη διαφορά. Οι προσομοιώσεις ξεκίνησαν με το όριο να βρίσκεται στα 500 βήματα και το ποσοστό επιτυχίας στο 15% και έφτασαν το όριο των 2000 βημάτων με 30%. Οπότε καταλαβαίνουμε πως ο Min-Conflicts δεν είναι καθόλου αποδοτικός για ένα δύσκολο πρόβλημα.

Είναι πολύ ενδιαφέρον να παρατηρήσουμε πως μεταξύ των περιπτώσεων όπου έχουμε για όριο βημάτων 1200, 1500 και 2000 το ποσοστό επιτυχίας διαφέρει πολύ λίγο. Έτσι λοιπόν καταλαβαίνουμε πως από κάποιο σημείο και μετά δεν έχει σημασία πόσο μεγάλο είναι το όριο των βημάτων, επειδή όπως ήταν αναμενόμενο ο αλγόριθμος κολλάει σε τοπικά βέλτιστα.

Μια ενδιαφέρουσα πληροφορία που παίρνουμε, κυρίως για τις προσομοιώσεις με επανεκκινήσεις είναι πως, αν έχουμε ένα αρκετά μεγάλο όριο βημάτων, με 4 επανεκκινήσεις θα έχουμε πολύ καλά αποτελέσματα. Για παράδειγμα στην προσομοίωση με 1000 όριο βημάτων, βλέπουμε πως λύνει ένα στα τέσσερα προβλήματα (25%), άρα με 4 επανεκκινήσεις θα μπορεί να λύσει κάθε προσομοίωση.

Τέλος, όταν χρησιμοποιούμε την άπληστη αρχικοποίηση ο αλγόριθμος δεν μπορεί να βρει λύση σχεδόν ποτέ, με την μοναδική φορά που βρήκε αποτέλεσμα να χρειαστεί λίγα βήματα με 68, αλλά πολύ χρόνο στα 5.3 δευτερόλεπτα.

Ακολουθεί ο πίνακας των αποτελεσμάτων για τον Min-Conflicts-WR:

Αλγόριθμος	Όριο Βημάτων	Όριο Επαναλήψεων	Αίθουσες Θεωρίας-Εργαστηρίου	ΒΕ-Θ / ΒΕ-Ε	Ποσοστό Επιτυχίας	Μέσος Χρόνος Λύσης	Μέσο πλήθος βημάτων-επανεκκινήσεων
MIN-CONFLICTS-WR	1000	2	3 - 2	9 / 15	50%	1275.18	560.13 - 0.34
	1000	5	3 - 2	9 / 15	85%	3077.14	711.66 - 1.423
	700	5	3 - 2	9 / 15	81%	2354.83	573.1 - 1.419
	500	5	3 - 2	9 / 15	64%	2161.02	471.61 - 1.687
	400	5	3 - 2	9 / 15	57%	1575.11	346.59 - 1.719
	400	10	3 - 2	9 / 15	85%	2330.06	359.11 - 2.953
	400	15	3 - 2	9 / 15	92%	3457.92	361.46 - 4.758
	300	20	3 - 2	9 / 15	81%	2936.32	281.70 - 5.197

Πίνακας 6.7 : Αποτελέσματα του min-conflicts-wr για το δύσκολο πρόβλημα.

Παρατηρήσεις για τα αποτελέσματα του αλγορίθμου Min-Conflicts-WR:

Στις προσομοιώσεις ξεκινήσαμε με ένα μεγάλο όριο βημάτων και μικρό όριο επανεκκινήσεων, και στη συνέχεια αυτό αναστρέφεται. Από το ποσοστό επιτυχίας παρατηρούμε πως ο Min-Conflicts-WR τα καταφέρνει πολύ καλύτερα από το Min-Conflicts λόγω των επανεκκινήσεων.

Βλέπουμε πως ο αλγόριθμος έχει την καλύτερη απόδοσή του στις περιπτώσεις όπου το όριο βημάτων είναι 1000 με 5 επανεκκινήσεις, και με 400 βήματα και 15 επανεκκινήσεις. Παρατηρούμε ότι όταν χρειάζεται πολλές επανεκκινήσεις έχει

μεγαλύτερο μέσω χρόνο λύσης, αλλά παρά τις απαιτήσεις σε χρόνο οι πολλές επανεκκινήσεις μας προσφέρουν μεγαλύτερη αποδοτικότητα.

Τέλος χρησιμοποιώντας την άπληστη αρχικοποίηση ο αλγόριθμος δεν μπόρεσε να λύσει το πρόβλημα, οπότε καταλαβαίνουμε πως αυτή η αρχικοποίηση δυσκολεύει πολύ τον αλγόριθμο.

Ακολουθεί ο πίνακας των αποτελεσμάτων για τον Most-Conflicts-WR:

Αλγόριθμος	Όριο Βημάτων	Όριο Επαναλήψεων	Αίθουσες Θεωρίας-Εργαστηρίου	ΒΕ-Θ / ΒΕ-Ε	Ποσοστό Επιτυχίας	Μέσος Χρόνος Λύσης	Μέσο πλήθος βημάτων-επανεκκινήσεων
MOST-CONFLICTS-WR	400	2	3 - 2	9 / 15	23%	979.739	327.93 - 0.695
	500	2	3 - 2	9 / 15	34%	1016.91	376.35 - 0.488
	300	5	3 - 2	9 / 15	32%	1451.97	261.88 - 1.781
	400	5	3 - 2	9 / 15	47%	1562.23	344.63 - 1.531
	500	5	3 - 2	9 / 15	53%	2170.98	419.83 - 1.867
	500	10	3 - 2	9 / 15	76%	3161.04	414.867 - 3.065
	500	15	3 - 2	9 / 15	94%	3884.47	427.96 - 3.914
	400	15	3 - 2	9 / 15	79%	3541.49	362.99 - 4.417
	300	20	3 - 2	9 / 15	80%	4561.36	286.75 - 8.035

Πίνακας 6.8 : Αποτελέσματα του most-conflicts-wr για το δύσκολο πρόβλημα.

Παρατηρήσεις για τα αποτελέσματα του αλγορίθμου Most-Conflicts-WR:

Επειδή στο το προηγούμενο μέτριο πρόβλημα παρατηρήσαμε πως για τον Most-Conflicts-WR δεν είναι πολύ απαραίτητο να έχει μεγάλο όριο επανεκκινήσεων στις πρώτες προσομοιώσεις χρησιμοποιήσαμε όριο 2. Αλλά όπως γίνεται κατανοητό από το ποσοστό επιτυχίας αυτό δεν είναι καθόλου αποδοτικό, το οποίο δικαιολογείται εάν σκεφτεί κανείς την μέθοδο που χρησιμοποιεί αυτός ο αλγόριθμος προσπαθώντας να διορθώσει το μάθημα με τις περισσότερες παραβιάσεις περιορισμών με αποτέλεσμα να οδηγείται συχνά σε τοπικά βέλτιστα.

Για να έχει μεγαλύτερες πιθανότητες να βρει λύση χρησιμοποιήσαμε περισσότερες επανεκκινήσεις. Επίσης βάλαμε τα όρια να είναι παρόμοια με αυτά στον Min-Conflicts-WR ώστε να είναι πιο εύκολη η σύγκριση. Παρατηρούμε πως όταν έχουν τα ίδια όριο ο Most-Conflicts-WR τείνει να είναι λιγότερο αποδοτικός, αλλά πιο γρήγορος και με μικρότερο μέσω πλήθος βημάτων και επανεκκινήσεων.

6.3.4 Γενικά συμπεράσματα

Εκτός από τις παρατηρήσεις που έγιναν για τα αποτελέσματα των πειραμάτων, υπάρχουν και κάποια γενικά συμπεράσματα τα οποία βγήκαν από το σύνολο των αποτελεσμάτων και από τα τρία προβλήματα. Και αυτά είναι:

- Από το σύνολο των πειραματικών αποτελεσμάτων μπορούμε να συμπεράνουμε πως οι καλύτερες μέθοδοι είναι οι δύο αλγόριθμοι που έχουν επανεκκινήσεις. Για να πετύχουμε την ίδια απόδοση με τον Min-Conflicts χρειάζονται πολλά βήματα, αν και μερικές φορές αυτό δεν μας το εξασφαλίζει.
- Παρατηρήσαμε επίσης πως κατά την εκτέλεση των αλγορίθμων αυτό που ήταν πιο χρονοβόρο είναι η αρχική ανάθεση των τιμών στα μαθήματα και πως τα βήματα ήταν λιγότερο χρονοβόρα ακόμα και στο δύσκολο πρόβλημα όπου το μέσο πλήθος βημάτων ήταν μέγιστο.
- Κάτι ακόμα που γίνεται κατανοητό από το σύνολο των προβλημάτων είναι πως όταν δεν χρησιμοποιείται κάποια άπληστη μέθοδος για την λήψη των αποφάσεων οι αλγόριθμοι γενικότερα λειτουργούν πιο αποδοτικά. Χαρακτηριστικό παράδειγμα είναι η διαφορά στο ποσοστό επιτυχία μεταξύ των Min-Conflicts-WR και Most-Conflicts-WR στο τελευταίο πρόβλημα.

Κεφάλαιο 7^{ο} : Συμπεράσματα και Μελλοντική Εργασία

7.1.Συμπεράσματα

Αφού εξετάστηκαν πλήρως οι αλγόριθμοι τοπικής αναζήτησης στο πρόβλημα του εβδομαδιαίου ακαδημαϊκού προγράμματος, μπορούν να βγουν κάποια συμπεράσματα όσον αφορά την χρησιμότητα τους.

Οι αλγόριθμοι τοπικής αναζήτησης γίνεται κατανοητό, από τα πειραματικά αποτελέσματα στο κεφάλαιο 6, πως είναι πολύ αποδοτικοί σε προβλήματα τα οποία δεν έχουν μεγάλο μέγεθος, για παράδειγμα εάν πάρουμε την περίπτωση του τμήματος μηχανικών πληροφορικής και τηλεπικοινωνιών, οι αλγόριθμοι είναι κατάλληλοι. Αν αναλογιστεί κιόλας κανείς ότι οι πόροι της σχολής αυτής (δηλαδή οι αίθουσες σύμφωνα με την μοντελοποίηση μας) είναι αρκετές και περισσεύουν μάλιστα, τότε το πρόβλημα είναι εύκολα επιλύσιμο και σε πολύ μικρό χρόνο. Αντίθετα θα αντιμετωπίζονταν δυσκολίες στις περιπτώσεις των μεγάλων πανεπιστημίων, όπως αυτή του Αριστοτελείου, λόγω των πλήθους των τμημάτων και του περιορισμένου χώρου. Ακόμα και σε μια δύσκολη όμως περίπτωση σαν και αυτήν ο αλγόριθμος θα έχει πιθανότητες να βρει λύση έχοντας ένα μεγάλο όριο βημάτων και επανεκκινήσεων, έχοντας σχετικά μικρές απαιτήσεις στον χρόνο.

7.2.Μελλοντική εργασία

Η παρούσα μελέτη θα βοηθούσε την προσπάθεια ανάπτυξης μίας εφαρμογής σίγουρα. Για να μπορέσει όμως να λειτουργήσει σωστά και να είναι πρακτική αυτή η εφαρμογή πρέπει μελλοντικά να ληφθούν και άλλοι παράμετροι υπ' όψιν, όπως :

- *Βελτιστοποίηση.* Πρέπει να γνωρίζει ο κατασκευαστής αν θα γίνεται βελτιστοποίηση του προβλήματος σε περίπτωση που βρεθεί λύση. Με τον όρο αυτό εννοείται η εύρεση της καλύτερης λύσης, αν βρεθεί λύση. Για παράδειγμα πόσο καλά μοιρασμένα είναι τα μαθήματα τόσο για τους καθηγητές όσο και για τους φοιτητές ή να μην αλλάζουν οι αίθουσες ανά εξάμηνο ώστε να μην υπάρχει κάποια καθυστέρηση μεταξύ των μαθημάτων.

- *Εισαγωγή περισσότερων περιορισμών*. Κάτι που επίσης μπορεί να δυσκολέψει το πρόβλημα είναι να μπουν περισσότεροι περιορισμοί. Για παράδειγμα , να συμπεριλαμβάναμε τις χωρητικότητες των αιθουσών και να τις συγκρίναμε με το πλήθος των φοιτητών του κάθε εξαμήνου. Επίσης ,οι περιορισμοί που παραλήφθηκαν εσκεμμένα είναι οι ήπιοι (soft) ή αλλιώς προτιμήσεις(preferences). Άρα κάτι που θα έκανε το πρόβλημα του εβδομαδιαίου προγράμματος ακόμα πιο ρεαλιστικό θα ήταν να χρησιμοποιούνταν και αυτοί. Ένα παράδειγμα είναι κάποιος καθηγητής να έχει την προτίμηση να κάνει όλες τις διδασκαλίες ενός μαθήματος σε μια ημέρα ή να υπάρχουν προτιμήσεις

σχετικά με τις πρωινές και απογευματινές ώρες για την διεξαγωγή κάποιων μαθημάτων.

- *Πιο αυστηρή μοντελοποίηση*. Στην παρούσα μελέτη έγιναν κάποιες παραδοχές για να διευκολυνθεί η κωδικοποίηση του προβλήματος. Για να πλησιάσουμε όμως μια πιο ρεαλιστική εφαρμογή θα πρέπει να ληφθούν υπόψη και άλλες παράμετροι. Ξανά χαρακτηριστικό παράδειγμα είναι ο περιορισμός χωρητικότητας των αιθουσών, ο αριθμός των φοιτητών του κάθε έτους, κ.τ.λ.

- *Αλληλεπίδραση με τον χρήστη*. Το σωστό σε μία τέτοια εφαρμογή θα ήταν ο πιθανός χρήστης να μπορεί να αλληλεπιδρά με το σύστημα. Δηλαδή καθώς θα έχει τοποθετήσει τα δεδομένα του προβλήματος να μπορεί να μπορεί να ρυθμίσει, στην εφαρμογή, την βαρύτητα των περιορισμών στην εφαρμογή, ή αν θέλει να ληφθούν υπόψη όλες οι παράμετροι.

- *Δυναμικό περιβάλλον*. Τελευταίο αλλά ίσως και το δυσκολότερο στην υλοποίηση θα ήταν η εφαρμογή να είναι *δυναμική*, δηλαδή να μπορεί το πρόγραμμα να αλλάζει αν συμβεί κάτι απρόοπτο. Το απρόοπτο στην περίπτωση του ακαδημαϊκού προγράμματος είναι να μην έρθει ένας καθηγητής λόγω καιρού, να γίνει κατάληψη, να χαθεί το μάθημα λόγω κάποιου τεχνικού προβλήματος, κ.τ.λ. Η δυσκολία εδώ είναι η έγκαιρη ενημέρωση της εφαρμογής και όσων εμπλέκονται έμμεσα σε αυτή ,δηλαδή των φοιτητών και των καθηγητών.

Βιβλιογραφία

[1] Markus Bohlin. Constraint satisfaction by local search. *2002*

[2] James F. Blakesley, Keith S. Murray, Frederick H. Wolf and Dagmar Murray. Academic Scheduling. *1998*

[3] A. Schaerf. A Survey of Automated Timetabling. *1999*

[4] D Abramson. Constructing School Timetables using Simulated Annealing: Aequential and Parallel Algorithms. *1991*

[5] Stuart Russel and Peter Norvig. Artificial Intelligence, A Modern Approach. *2003*

[6] Kyriakos Zervoudakis and Panagiotis Stamatopoulos. A Generic Object-Oriented Constraint-Based Model for University Course Timetabling. *2001*

[7] Panagiotis Stamatopoulos, Efstratios Viglas and Serafeim Karaboyas. Nearly Optimun Timetable Construction Through CLP and Intelligent Search. *1998*

[8] Harikleia Frangouli, Vassilis Harmadas and Panagiotis Stamatopoulos. UTSE : Construction of Optimum Timetables for University Courses – A CLP Based Approach. *1995*

[9] George M. White and Junhan Zhang. Generating Complete University Timetables by Combining Tabu Search with Constraint Logic. *1998*

[10] Martin Henz and Jörg Würtz. Using Oz for College Timetabling. *1995*

[11] Christelle Guéret, Narendra Jussien, Patrice Boizumault and Christian Prins. Building University timetables using Constraint Logic Programming.*1996*

[12] Hans-Joachim Goltz. Combined Automatic and Interactive Timetabling Using Constraint Logic Programming. *2000*

[13] Hans-Joachim Goltz, Georg Küchler and Dirk Matzle. Constraint-Based Timetabling for Universities. *1998*

[14] Hans-Joachim Goltz. Reducing Domains for Search in CLP(FD) and Its Application to Job-Shop Scheduling. *1995*

[15] Hans-Joachim Goltz and Ulrich John. Methods for Solving Practical Problems of Jop-Shop Scheduling Modeled in CPL(FD). *1996*

[16] P.Boizumault, Y. Delon and L.Peridy. Constraint Logic Programming for Examination Timetabling. *1995*

[17] Dons Banks, Peter Van Beek and Amnon Meisels. A Heuristic Incremental Modeling Approach to Course Timetabling. *1998*

[18] F.P.M. Dignum, W.P.M. Nuijten and L.M.A. Janssen. Solving a Time Tabling Problem by Constraint Satisfaction. *1995*

[19] Peter Wilke, Matthias Gröbner and Norbert Oster. A Hybrid Genetic Algorithm for School Timetabling *2002*

[20] Slim Abdennadher and Michael Marte. University Timetabling using Constraint Handling Rules. *2000*

[21] Luìs Paulo Reis and Eugénio Oliveira. A Language for Specifying Complete Timetabling Problems. *2000*

[22] Maria Kambi and David Gilbert. Timetabling in Constraint Logic Programming. *1996*

[23] Claude Le Pape. Three mechanisms for managing resource constraints in a library for constraint-based schedule. *1995*

[24] Rina Dechter and Itay Meiri. Experimental Evaluation of Preprocessing Algorithms for Constraint Satisfaction Problems.*1994*