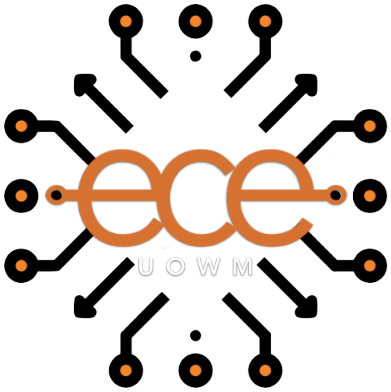


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Χρονοπρογραμματισμός ωρολογίου προγράμματος μαθημάτων (Scheduling class timetables)

Ιωάννης - Αιμίλιος Γνωστόπουλος (ΑΜ: 1155)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

9 Οκτωβρίου 2024

Περίληψη

Στα ελληνικά πανεπιστήμια παρατηρείται συχνά το φαινόμενο να καθυστερεί να αναρτηθεί το ωρολόγιο πρόγραμμα των μαθημάτων της εκάστοτε σχολής καθώς δεν υπάρχει αξιόπιστος αλγόριθμος που να δίνει άριστο αποτέλεσμα και να ικανοποιεί όλες τις απαιτήσεις που χρειάζονται. Στη συγκεκριμένη διπλωματική εργασία γίνεται μελέτη σε αλγορίθμους που ικανοποιούν σε πολύ μεγάλο βαθμό αυτό το ζήτημα και πραγματοποιούνται έλεγχοι σε αυτούς ώστε να διασφαλίζεται η ορθή λειτουργία τους καθώς και συγκρίσεις μεταξύ τους με τη βοήθεια ορισμένων μετρικών σύγκρισης. Οι αλγόριθμοι που εφαρμόστηκαν είναι Γενετικοί (GA, Timetable) και αλγόριμοι ικανοιποίησης περιορισμών (Default backtracking, LCV (Least Constraining Value), MRV (Minimum Remaining Values), Degree Heuristic, Forward Checking, Constraint Propagation). Με βάση τα αποτελέσματα και τις συγκρίσεις μεταξύ όλων των αλγορίθμων αυτός που αντιμετώπιζε πιο αποτελεσματικά το πρόβλημα είναι ο constraint propagation καθώς και στους ελαστικούς και στους σκληρούς περιορισμούς παρατηρήθηκαν λιγότερα προβλήματα, γεγονός που τον καθιστά έστω και με μικρή διαφορά την καλύτερη επιλογή.

Λέξεις κλειδιά: Ωρολόγιο Πρόγραμμα, Χρονοπρογραμματισμός, Γενετικοί Αλγόριθμοι, Αλγόριθμοι Ικανοποίησης Περιορισμών, Μετρικές Σύγκρισης.

Abstract

In Greek universities, it is often observed that the timetable of the courses of each school is delayed, as there is no reliable algorithm that gives an excellent result and satisfies all the requirements that are needed.In this particular thesis, a study is made on algorithms that satisfy this issue to a very large extent and checks are carried out on them to ensure their correct operation as well as comparisons between them with the help of certain comparison metrics. The algorithms applied are Genetic (GA, Timetable) and constraint satisfaction algorithms (Default backtracking, LCV (Least Constraining Value), MRV (Minimum Remaining Values), Degree Heuristic, Forward Checking, Constraint Propagation). Based on the results and the comparisons between all the algorithms, the one that dealt with the problem more effectively is the constraint propagation, as well as the soft and hard constraints saw fewer problems, which makes it the best choice even by a small margin.

Keywords: Class Timetable, Scheduling, Genetic Algorithms, Constraint Satisfaction Problems, Metrics.

Δήλωση Πνευματικών Δικαιωμάτων

Δήλωση Πνευματικών Δικαιωμάτων Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο ”Χρονοπρογραμματισμός ωρολογίου προγράμματος μαθημάτων” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Πλόσκα Νικόλαου αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος απεικονίσεων

Κεφάλαιο 1

Εισαγωγή

Η παρούσα διπλωματική εργασία εστιάζει στο Χρονοπρογραμματισμό Ωρολογίου Προγράμματος Μαθημάτων (Scheduling Class Timetable) για το εκάστοτε εξάμηνο ενός Πανεπιστημίου. Δεδομένης της περίπλοκης φύσης του και του πλήθους των μεταβλητών και των παραμέτρων που εμπλέκονται, αυτό το πρόβλημα εμπίπτει στην κατηγορία NP-complete, θέτοντας μια σημαντική πρόκληση στην εύρεση της καλύτερης λύσης.

1.1 Ορισμός του προβλήματος

Η πρόκληση της δημιουργίας ενός χρονοδιαγράμματος μπορεί να περιγραφεί ως ένα παζλ διαχείρισης χρόνου, όπου πρέπει να προγραμματιστεί ένας καθορισμένος αριθμός μαθημάτων, καθένα από τα οποία πιθανώς να εμπίπτει σε μία ή περισσότερες κατηγορίες θεμάτων. Κάθε τάξη απαιτεί συγκεκριμένους πόρους, όπως αίθουσες διδασκαλίας και εκπαιδευτές, οι οποίοι είναι περιορισμένοι σε διαθεσιμότητα. Επιπλέον, πρέπει να τηρούνται ορισμένοι αυστηροί κανόνες, όπως η αποφυγή συγκρούσεων προγραμματισμού για τα μαθήματα που διδάσκονται από τον ίδιο εκπαιδευτή, καθώς και λιγότερο αυστηρές οδηγίες, όπως η ανάθεση μεγάλων τάξεων σε ευρύχωρες αίθουσες.

Η αυτοματοποίηση της δημιουργίας χρονοδιαγράμματος παρουσιάζει ένα τρομερό έργο λόγω του τεράστιου όγκου δεδομένων και της ποικίλης σειράς περιορισμών που εμπλέκονται. Ταξινομημένο ως NP-complete, η εύρεση λύσης γίνεται εκθετικά πιο χρονοβόρα με αυξημένη πολυπλοκότητα δεδομένων. Κατά συνέπεια, το μοντέλο και ο αλγόριθμος που χρησιμοποιούνται για την αντιμετώπιση αυτού του προβλήματος είναι υψίστης σημασίας για την αποτελεσματικότητα του συστήματος.

Το αίνιγμα του χρονοδιαγράμματος μπορεί να αποφέρει πολλές λύσεις, με μία να ξεχωρίζει δυνητικά ως βέλτιστη. Αντίθετα, εάν το πρόβλημα έχει διατυπωθεί με τέτοιο τρόπο ώστε να μην υπάρχει βιώσιμη λύση, αυτή η ασυμφωνία πρέπει να αναγνωριστεί, ενδεχομένως μέσω εξαντλητικής διερεύνησης πιθανών λύσεων.

1.2 Κίνητρα και στόχοι υλοποίησης

Αυτή η εργασία χρησιμοποιεί τεχνολογία περιορισμών για την αντιμετώπιση του προβλήματος. Η δημιουργία χρονοδιαγραμμάτων για τα μαθήματα είναι μια προκλητική και χρονοβόρα εργασία λόγω του εκτεταμένου χώρου αναζήτησης που εμπλέκεται. Για την αντιμετώπιση αυτής της πρόκλησης, χρησιμοποιήθηκαν οι CSP (Constraint Satisfaction Problem) αλγόριθμοι, οι οποίοι είναι κατάλληλοι για την αντιμετώπιση προβλημάτων με τέτοιες πολυπλοκότητες καθώς και Genetic αλγόριθμοι.

Η υλοποίηση του προγράμματος πραγματοποιήθηκε με τη χρήση Python, μιας αντικειμενοστρεφούς γλώσσας προγραμματισμού, η οποία προσφέρει κατάλληλα εργαλεία για την επίλυση προβλημάτων χρονοπρογραμματισμού.

Συνοψίζοντας, αυτή η μελέτη πραγματεύεται τη δημιουργία χρονοδιαγραμμάτων μαθημάτων για την εύρυθμη λειτουργία των Πανεπιστημιακών Τμημάτων.

1.3 Διάρθρωση κειμένου

Στο δεύτερο κεφάλαιο παρουσιάζονται κάποιοι αλγόριθμοι οι οποίοι λύνουν το ζήτημα αυτό καθώς και η βιβλιογραφική έρευνα πάνω σε άρθρα, βιβλία, διπλωματικές άλλων ερευνητών οι οποίοι επιλύουν το πρόβλημα.

Στο τρίτο κεφάλαιο γίνεται η υλοποίηση του ζητήματος, με την αναφορά στους αλγορίθμους που χρησιμοποιήθηκαν και τους περιορισμούς που εφαρμόστηκαν για τη σωστή λειτουργία τους καθώς και τη χρήση ψευδοκώδικα για τον καθένα.

Στο τέταρτο κεφάλαιο γίνεται η σύγκριση μεταξύ των αλγορίθμων, η καταγραφή των αποτελεσμάτων και η δημιουργία γραφημάτων, ώστε να επιλεχθεί ο αλγόριθμος που προσφέρει τη βέλτιστη λύση.

Τέλος, στο πέμπτο κεφάλαιο γίνεται αναφορά των συμπερασμάτων που προκύπτουν καθώς και μελλοντικές βελτιώσεις που μπορούν να εφαρμοστούν για τη

επίλυση του προβλήματος του χρονοπρογραμματισμού.

Κεφάλαιο 2

Βιβλιογραφική ανασκόπηση

2.1 Μελέτη αλγορίθμων

Στην ενότητα αυτή, θα εμβαθύνουμε σε έρευνες που πραγματοποιήθηκαν σχετικά με το θέμα του προγραμματισμού χρονοδιαγραμμάτων, διερευνώντας τους διάφορους αλγόριθμους που χρησιμοποιούνται στη βιβλιογραφία και τα αποτελέσματα που προκύπτουν από τη χρήση τους.

2.1.1 Γενετικός αλγόριθμος

Οι Γενετικοί αλγόριθμοι ανήκουν στο κλάδο της επιστήμης υπολογιστών και αποτελούν μια μέθοδο αναζήτησης βέλτιστων λύσεων σε συστήματα που μπορούν να περιγραφούν ως μαθηματικό πρόβλημα. Είναι χρήσιμοι σε προβλήματα που περιέχουν πολλές παραμέτρους και δεν υπάρχει αναλυτική μέθοδος που να μπορεί να βρει το βέλτιστο συνδυασμό τιμών για τις μεταβλητές ώστε το υπό εξέταση σύστημα να αντιδρά όσο το δυνατόν με τον επιθυμητό τρόπο.

Το 1950, ο Turing πρότεινε την ιδέα μιας «μηχανής μάθησης» που θα μιμείται τις εξελικτικές αρχές [1]. Οι πρώτες προσπάθειες για την προσομοίωση της εξέλιξης σε υπολογιστές ξεκίνησαν το 1954 με τον Barricelli, ο οποίος χρησιμοποίησε τον υπολογιστή στο Ινστιτούτο Προηγμένων Μελετών στο Πρίνστον του Νιου Τζέρσεϊ [2, 3]. Το έργο του 1954, ωστόσο, δεν κέρδισε ιδιαίτερη προσοχή εκείνη την εποχή. Το 1957, ο Αυστραλός Fraser άρχισε να δημοσιεύει μια σειρά εργασιών για την προσομοίωση της τεχνητής επιλογής που περιλαμβάνει οργανισμούς με πολλαπλούς γενετικούς τόπους που επηρεάζουν μετρήσιμα χαρακτηριστικά [4]. Αυτή η περίοδος σηματοδότησε την αρχή της συχνότερης χρήσης του computer simulation

of evolution μεταξύ των βιολόγων, οδηγώντας σε λεπτομερείς μεθόδους που περιγράφονται στα βιβλία των Fraser και Burnell (1970) και Crosby (1973) [5, 6]. Οι προσομοιώσεις του Fraser ενσωμάτωσαν βασικά στοιχεία αυτού που θα γίνονταν στη συνέχεια σύγχρονοι γενετικοί αλγόριθμοι. Ταυτόχρονα, ο Bremermann διεξήγαγε έρευνα σε όλη τη δεκαετία του 1960 που χρησιμοποίησε επίσης populations of solutions σε προβλήματα βελτιστοποίησης, ενσωματώνοντας crossover, mutation και selection, που είναι θεμελιώδεις πτυχές των γενετικών αλγορίθμων. Οι πρωτοπόροι σε αυτόν τον τομέα περιελάμβαναν επίσης τον Friedberg et al. το έργο των οποίων συγκεντρώνεται σε μια συλλογή που επιμελήθηκε ο Fogel [7].

Παρά την προσομοίωση της εξέλιξης του παιχνιδιού από τον Barricelli το 1963, η τεχνητή εξέλιξη αναγνωρίστηκε ως μέθοδος βελτιστοποίησης μόνο με τις συνεισφορές των Rechenberg και Schwefel στη δεκαετία του 1960 και στις αρχές της δεκαετίας του 1970 [8]. Η ομάδα του Rechenberg εφάρμοσε με επιτυχία εξελικτικές στρατηγικές για την επίλυση σύνθετων προβλημάτων μηχανικής [9, 10, 11, 12]. Ταυτόχρονα, ο Fogel ανέπτυξε τον εξελικτικό προγραμματισμό (Evolutionary programming) για τη δημιουργία τεχνητής νοημοσύνης, χρησιμοποιώντας αρχικά μηχανές πεπερασμένης κατάστασης για την πρόβλεψη περιβάλλοντων και εφαρμόζοντας variation και selection για τη βελτιστοποίηση αυτών των προβλέψεων. Η έννοια των γενετικών αλγορίθμων έγινε δημοφιλής από τον Holland στις αρχές της δεκαετίας του 1970, ιδιαίτερα μέσω του βιβλίου του το 1975 ”Adaptation in Natural and Artificial Systems”. Η έρευνα του Holland, που προέρχεται από μελέτες για κυτταρικά αυτόματα στο Πανεπιστήμιο του Μίσιγκαν, εισήγαγε ένα επίσημο πλαίσιο για την πρόβλεψη της ποιότητας των επόμενων γενεών, γνωστό ως Θεώρημα Σχήματος Holland (Holland’s Schema Theorem). Το πεδίο των γενετικών αλγορίθμων πέρασε από τη θεωρητική εξερεύνηση σε ευρύτερη εφαρμογή μετά το Πρώτο Διεθνές Συνέδριο για τους Γενετικούς Αλγόριθμους στο Πίτσμπουργκ της Πενσυλβάνια, στα μέσα της δεκαετίας του 1980.

Αφού έγινε μια ιστορική αναφορά στους Γενετικούς αλγορίθμους στη συνέχεια θα μελετηθούν άρθρα τα οποία ασχολήθηκαν με το πρόβλημα του χρονοπρογραμματισμού και χρησιμοποιήθηκαν Γενετικοί αλγόριθμοι για τη λύση του. Η δημιουργία χρονοδιαγραμμάτων για τα μαθήματα πανεμιστημίου είναι ένα σύνθετο NP-Hard πρόβλημα που είναι δύσκολο να λυθεί με τη χρήση συμβατικών μεθόδων λόγω της

πολυπλοκότητάς του. Οι αλγόριθμοι Evolutionary computation (EC), και ιδιαίτερα οι γενετικοί αλγόριθμοι (GA), είναι αποτελεσματικοί στην αντιμετώπιση τέτοιων προβλημάτων. Η εργασία του Ghaemi et al. παρουσιάζει την εφαρμογή ενός GA για την ελαχιστοποίηση των σφαλμάτων στο χρονοδιάγραμμα των πανεπιστημιακών μαθημάτων [13]. Διερευνώνται δύο συγκεκριμένες προσεγγίσεις: ο Modified Genetic Algorithm (MGA) και ο Cooperative Genetic Algorithm (CGA). Σε αυτή την εργασία κατασκευάζονται προγράμματα για τα προπτυχιακά μαθήματα στη Σχολή Ηλεκτρολόγων Μηχανικών του Πανεπιστημίου Tabriz. Τα ωρολόγια προγράμματα που δημιουργούνται χωρίζονται σε:

- Χειμερινό εξάμηνο: Εξάμηνο 1, Εξάμηνο 3, Εξάμηνο 5, Εξάμηνο 7
- Εαρινό εξάμηνο: Εξάμηνο 2, Εξάμηνο 4, Εξάμηνο 6, Εξάμηνο 8

Κάθε μέρα χωρίζεται σε τέσσερις χρονικές περιόδους, καθεμία από τις οποίες αποτελείται από μια διάλεξη 90 λεπτών ακολουθούμενη από ένα διάλειμμα 30 λεπτών. Τα μαθήματα προγραμματίζονται για μία ή δύο χρονικές περιόδους την εβδομάδα.

Το πρόβλημα του χρονοδιαγράμματος ορίζεται χρησιμοποιώντας πέντε κύριες οντότητες:

- Καθηγητής: Τα μέλη ΔΕΠ που θα παραδώσουν τις διαλέξεις.
- Φοιτητής: Οι ομάδες μαθητών που παρακολουθούν τα μαθήματα.
- Μαθήματα: Τα μαθήματα που προσφέρονται κάθε εξάμηνο.
- Αίθουσα: Οι φυσικές τοποθεσίες όπου γίνονται οι διαλέξεις.
- Διδακτικές περίοδοι: Οι συγκεκριμένες χρονικές περίοδοι κατά τις οποίες προγραμματίζονται οι διαλέξεις.

Οι αλγόριθμοι MGA και CGA πραγματοποιούνται για 6 αίθουσες, 40 μαθήματα και 18 διδάσκοντες εκ των οποίων οι τέσσερις οι προϊστάμενοι των τμημάτων. Αυτές οι οντότητες πρέπει να ανατεθούν με ακρίβεια για να δημιουργήσουν ένα ολοκληρωμένο και χωρίς σφάλματα χρονοδιάγραμμα.

Ο Modified Genetic Algorithm (MGA) λειτουργεί ως εξής:

Το πρόγραμμα ξεκινά με δύο ομάδες χρωμοσωμάτων γνωστών ως 1 pop και 2 pop. Είναι τυχαίοι πληθυσμοί που δημιουργούνται χωριστά. Τα χρωμοσώματα του 1 pop και 2 pop αποτελούνται από 40 genes. Για τη δημιουργία πλήρων χρωμοσωμάτων, επιλέγονται 3 χρωμοσώματα τυχαία από 1 pop. Στη συνέχεια, κάθε υποψήφιο χρωμόσωμα απο το 1 pop συνεισφέρει μαζί με το 2 pop για την παραγωγή πλήρους χρωμοσώματος. Αυτή η μέθοδος δρα στο pop 2 με τον ίδιο τρόπο. Αυτή η μέθοδος έχει το πλεονέκτημα ότι μειώνεται το μέγεθος του πληθυσμού και τα αποτελέσματα επιτυγχάνονται σε μικρότερο χρονικό διάστημα.

Ο Cooperative Genetic Algorithm (CGA) λειτουργεί ως εξής:

Οι γενετικοί αλγόριθμοι λειτουργούν αργά σε μεγάλο χώρο αναζήτησης, επομένως για την επιτάχυνση της διαδικασίας εφαρμόζεται ο αλγόριθμος αυτός. Έτσι αρχικά χωρίζονται τα 1 pop και 2 pop σε 2 species ξεχωριστά. Στη συνέχεια συνεργάζονται για τη μείωση του κόστους και επιλέγονται 3 τυχαία χρωμοσώματα από κάθε species. Στο Σχήμα 2.1 παρουσιάζεται η αναλυτική διαδικασία και των δυό αλγορίθμων.

Το συμπέρασμα που κατέληξαν είναι ότι και ο MGA και ο CGA είναι αλγόριθμοι που βελτίωσαν σημαντικά τα αποτελέσματα και σε πολύ λίγο χρόνο. Μια διαφορά των δύο είναι ότι ο CGA αλγόριθμος ικανοποιεί σε καλύτερο βαθμό τα constraints που τους δόθηκαν συγκριτικά με τον MGA.

Σχήμα 2.1: Η μέθοδος των αλγορίθμων MGA-CGA

Step	MGA method	CGA method
1	Generate pop_1 and pop_2.	Generate $species_1$ & $species_2$
2	Produce the full population as mentioned method.	Produce the full chromosomes for each species as mentioned method.
3	Produce the weekly lecturer timetable	
4	Apply the repair functions	
5	Evaluate fitness of the chromosomes	Evaluate fitness of the obtained full chromosomes for each species
6	Repeat steps 6.1 to 6.6 until achieve to stop conditions	
6.1	Selection and Crossover	
6.2	Mutation: a) Constant μ over all generation. b) First generation has μ =0.2 when minimum cost value is fixed in 10 subsequence iteration, decrease the mutation rate. Note that minimum mutation rate is 0.02.	
6.3	Produce the full population as mentioned for pop_1 and pop_2 .	Produce the full chromosomes for each species as mentioned method
6.4	Apply the repair functions	
6.5	Evaluate fitness of the chromosomes	Evaluate fitness of the obtained full chromosomes for each species
6.6	stop condition: a) Achieve to the defined minimum cost. b) Achieve to the defined maximum iteration. c) Having unchanged minimum cost value in 15 consequence iteration with iteration number above 30.	
7	Select the best chromosome of full chromosomes	

2.1.2 Η αναζήτηση Tabu

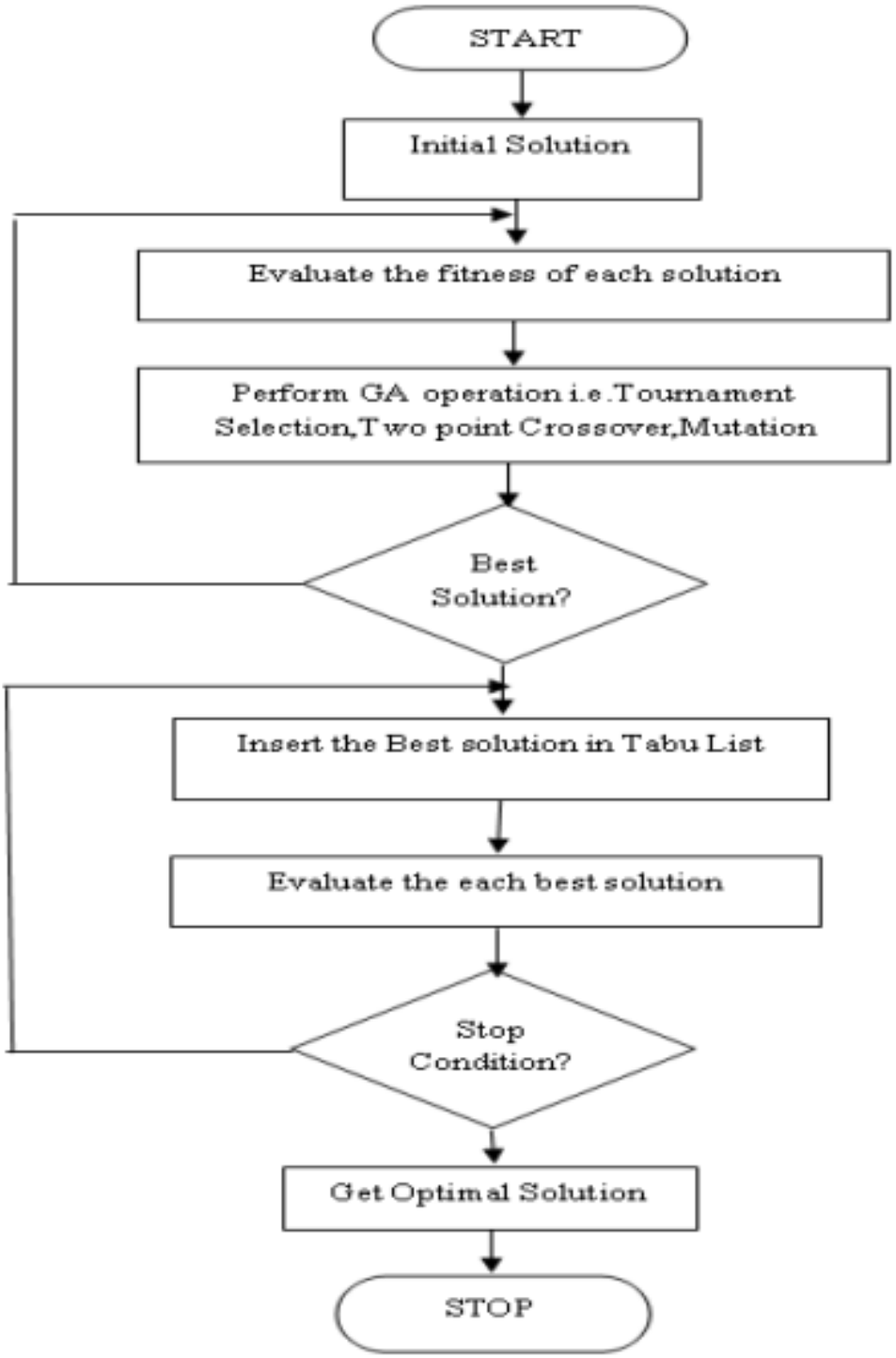
Η αναζήτηση Tabu (TS) είναι μια μεταευρετική μέθοδος αναζήτησης που χρησιμοποιεί μεθόδους τοπικής αναζήτησης (local search) οι οποίες εφαρμόζονται για μαθηματική βελτιστοποίηση. Δημιουργήθηκε από τον Glover το 1986 και επισημοποιήθηκε το 1989 [14, 15, 16].

Λειτουργεί με επαναληπτική μετάβαση από τη μια πιθανή λύση στην άλλη και σε αντίθεση με τις απλές μεθόδους τοπικής αναζήτησης, η Αναζήτηση Tabu χρησιμοποιεί μια δομή μνήμης που ονομάζεται ''tabu list'' για να παρακολουθεί τις λύσεις ή τις κινήσεις που επισκεφτήκαμε πρόσφατα. Αυτό αποτρέπει τον αλγόριθμο από το να επανεξετάσει αυτές τις λύσεις μέσα σε έναν ορισμένο αριθμό επαναλήψεων, ενθαρρύνοντας έτσι την εξερεύνηση νέων περιοχών στο χώρο λύσεων. Αυτοί οι μηχανισμοί καθιστούν τη Tabu Search μια ισχυρή και ευέλικτη μέθοδο για την επίλυση εξαιρετικά απαιτητικών προβλημάτων όπως το scheduling class timetable σε αυτή τη περίπτωση.

Οι Sonawane και Ragha χρησιμοποίησαν γενετικό αλγόριθμο (GA) και αλγόριθμο αναζήτησης Tabu για τη λύση του χρονοπρογραμματισμού ωρολογίου προγράμματος [17]. Ο GA έχει την ικανότητα να χειρίζεται έναν πολύ περίπλοκο χώρο αναζήτησης με μεγάλη πιθανότητα εύρεσης βέλτιστης λύσης και η αναζήτηση Tabu χρησιμοποιείται για την ικανότητά της να απομνημονεύει προηγούμενες κινήσεις ώστε να αποτρέψει την αναζήτηση προηγούμενων επισκέψεων, επομένως είναι ευκολότερο να επιτευχθεί η βέλτιστη λύση σε σύντομο χρονικό διάστημα. Τα μαθήματα διδάσκονται 5 μέρες την εβδομάδα σε 9 περιόδους και 45 timeslots , από τα οποία τα 40 χρησιμοποιούνται για διαλέξεις και εργαστήρια και τα υπόλοιπα για διαλείμμα. Στο Σχήμα 2.2 φαίνεται η διαδικασία δημιουργίας χρονοδιαγράμματος με τον συνδιασμό του γενετικού αλγορίθμου και της αναζήτησης Tabu.

Αυτή η διαδικασία υβριδισμού γενετικού αλγορίθμου και αναζήτησης Tabu βγάζει πολύ ικανοποιητικά αποτελέσματα καθώς εκμεταλλεύεται τα προτερήματα και των δύο ικανοποιώντας όλους τους περιορισμούς και φτάνει πιο εύκολα στη βέλτιστη λύση.

Σχήμα 2.2: Η λειτουργία των αλγορίθμων



2.1.3 Αλγόριθμος ικανοποίησης περιορισμών

Η ικανοποίηση περιορισμών (Constraint satisfaction problem) ως γενικό πρόβλημα προέκυψε στον τομέα της τεχνητής νοημοσύνης τη δεκαετία του 1970 [18]. Παρόλα αυτά όμως αργότερα και συγκεκριμένα κατά τη διάρκεια των δεκαετιών του 1980 και του 1990, αναπτύχθηκε η ενσωμάτωση περιορισμών σε μια γλώσσα

προγραμματισμού. Η πρώτη γλώσσα που επινοήθηκε ρητά με εγγενή υποστήριξη για προγραμματισμό περιορισμών ήταν η Prolog. Από τότε έχουν γίνει διαθέσιμοι και σε άλλες γλώσσες αλγόριθμοι ικανοποίησης περιορισμών, όπως η C++ ή η Java.

Η δημιουργία ωρολογίου προγράμματος μαθημάτων με τη βοήθεια αλγορίθμου ικανοποίησης περιορισμών (Constraint Satisfaction Problem) περιλαμβάνει την αντιστοίχιση χρονοθυρίδων και δωματίων σε ένα σύνολο τάξεων, ενώ επίσης ικανοποιούνται ποικίλοι περιορισμοί. Αυτοί οι περιορισμοί μπορεί να περιλαμβάνουν την αποφυγή συγκρούσεων στις ώρες διδασκαλίας μεταξύ μαθητή και δασκάλου, τη διασφάλιση ότι οι αίθουσες έχουν την κατάλληλη χωρητικότητα και εξοπλισμό και την τήρηση θεσμικών πολιτικών όπως προτιμώμενες ώρες διδασκαλίας ή συγκεκριμένες αναθέσεις αιθουσών για ορισμένα μαθήματα. Οι αλγόριθμοι ικανοποίησης περιορισμών, όπως το backtracking, local search ή ο constraint propagation, εξερευνούν συστηματικά τα πιθανά χρονοδιαγράμματα για να βρουν λύσεις που πληρούν όλες τις καθορισμένες απαιτήσεις. Με την κωδικοποίηση αυτών των περιορισμών στον αλγόριθμο, καθίσταται δυνατή η αποτελεσματική δημιουργία εφικτών χρονοδιαγραμμάτων ακόμη και για μεγάλα και πολύπλοκα προβλήματα προγραμματισμού.

Η διαδικασία ξεκινά ορίζοντας τις μεταβλητές (τάξεις, χρονοθυρίδες και δωμάτια) και τους περιορισμούς. Οι περιορισμοί ενδέχεται να προσδιορίζουν ότι δεν υπάρχουν δύο τάξεις που να μοιράζονται την ίδια αίθουσα την ίδια στιγμή και ότι οι καθηγητές δεν είναι εφικτό να κάνουν μάθημα σε 2 αίθουσες ταυτόχρονα. Ο αλγόριθμος εκχωρεί επαναληπτικά τιμές σε μεταβλητές, ελέγχοντας για παραβιάσεις περιορισμών σε κάθε βήμα. Τεχνικές όπως ο constraint propagation μπορούν να το απλοποιήσουν μειώνοντας τον αριθμό των πιθανών αναθέσεων μαθημάτων μέσω της έγκαιρης ανίχνευσης της παραβίασης των περιορισμών. Οι σύγχρονοι αλγόριθμοι ικανοποίησης περιορισμών μπορούν να χειριστούν δυναμικές αλλαγές και να ενσωματώσουν προτιμήσεις, καθιστώντας τους κατάλληλους για πραγματικές εφαρμογές σε εκπαιδευτικά ιδρύματα όπου η ευελιξία και η προσαρμοστικότητα είναι ζωτικής σημασίας.

Οι Blanco και Khatib αναφέρουν ότι γίνεται απόπειρα εύρεσης της καλύτερης λύσης για το πρόβλημα του χρονοπρογραμματισμού με τη βοήθεια αλγορίθμου ικανοποίησης περιορισμών για τις ανάγκες του Computer Science Program του Institute

of Technology της Φλόριντα [19]. Αρχικά η διαδικασία εύρεσης του χρονοδιαγράμματος έγινε σε 2 φάσεις. Στην πρώτη φάση γίνεται η ανάθεση μαθημάτων στο διδακτικό προσωπικό το οποίο θεωρείται πολύ σημαντικό και στη δεύτερη φάση τοποθετούνται τα μαθήματα στις διαθέσιμες χρονικές περιόδους. Πραγματοποιήθηκαν αρκετές δοκιμές με διδακτικό προσωπικό έως 12 άτομα και έως 50 μαθήματα. Αυτό που έγινε αντιληπτό σε παράδειγμα με 10 καθηγητές, 40 μαθήματα και 32 χρονικές περιόδους για να τοποθετηθούν τα μαθήματα (9 απο τις οποίες αφορούν νυχτερινό ωράριο) και για προπτυχιακές και για μεταπτυχιακές σπουδές, είναι ότι επειδή τα μαθήματα των μεταπτυχιακών σπουδών τοποθετούνται κυρίως απογευματινές ώρες ξεκινώντας πρώτα με τη τοποθέτηση αυτών η βέλτιστη λύση βρίσκεται σε πολύ μικρότερο χρόνο όπως φαίνεται και στο Σχήμα 2.3.

Σχήμα 2.3: Αποτελέσματα δημιουργίας ωρολογίου προγράμματος

	Undergraduate courses first	Graduate courses first
Day time slots first	15 minutes	10 seconds
Evening time slots first	16 minutes	15 seconds

Αυτό συμβαίνει διότι οι απογευματινές περίοδοι είναι λιγότερες απο τις πρωινές και έτσι τοποθετούνται με μεγαλύτερη ευκολία τα μαθήματα. Με τη χρήση του αλγορίθμου ικανοποίησης περιορισμών σε αυτό το άρθρο και τα αποτελέσματα που πάρθηκαν είναι φανερό ότι ικανοιούνται όλοι οι περιορισμοί που ζητήθηκαν και διαπιστώθηκε ότι ο χωρισμός σε 2 φάσεις ήταν πιο αποδοτικός και πιο εύκολα αντιμετωπίσιμος σε περίπτωση σφάλματος.

2.1.4 Βελτιστοποίηση σμήνους σωματιδίων

Η βελτιστοποίηση σμήνους σωματιδίων (Particle swarm optimization-είναι μια υπολογιστική μέθοδος που βελτιστοποιεί ένα πρόβλημα προσπαθώντας επαναληπτικά να βελτιώσει μια πιθανή λύση. Επιλύει ένα πρόβλημα έχοντας έναν πληθυσμό πιθανών λύσεων, στη συγκεκριμένη περίπτωση μετατοπιζόμενα σωματίδια, και μετακινώντας αυτά τα σωματίδια στον χώρο αναζήτησης σύμφωνα με έναν απλό μαθηματικό τύπο πάνω από τη θέση και την ταχύτητα του σωματιδίου. Η κίνηση κάθε

σωματιδίου επηρεάζεται από την πιο κοντινή του θέση, αλλά επίσης καθοδηγείται προς τις πιο γνωστές θέσεις στον χώρο αναζήτησης, οι οποίες ενημερώνονται καθώς εντοπίζονται καλύτερες θέσεις από άλλα σωματίδια. Αυτό αναμένεται να οδηγήσει το σμήνος προς τις καλύτερες λύσεις.

Το PSO αρχικά αποδόθηκε στους Kennedy et al. και προοριζόταν αρχικά για την προσομοίωση της κοινωνικής συμπεριφοράς, ως μια αναπαράσταση της κίνησης των οργανισμών σε ένα κοπάδι πουλιών ή ένα κοπάδι ψαριών [20, 21]. Ο αλγόριθμος απλοποιήθηκε και παρατηρήθηκε ότι εκτελούσε βελτιστοποίηση. Το βιβλίο των Kennedy και Eberhart περιγράφει πολλές φιλοσοφικές πτυχές του PSO και της νοημοσύνης του σμήνους [22]. Μια εκτενής έρευνα για τις εφαρμογές PSO γίνεται από την Poli[23]. Το 2017, δημοσιεύτηκε μια ολοκληρωμένη ανασκόπηση σε θεωρητικές και πειραματικές εργασίες για το PSO από τους Bonyadi και Michalewicz[24].

Το PSO είναι μία μετευρετική μέθοδος καθώς κάνει ελάχιστες έως καθόλου υποθέσεις σχετικά με το πρόβλημα που βελτιστοποιείται και μπορεί να πραγματοποιήσει αναζήτηση σε πολύ μεγάλους χώρους πιθανών λύσεων. Ωστόσο, μεταευρετικές τεχνικές όπως το PSO δεν εγγυώνται ότι θα βρεθεί ποτέ μια βέλτιστη λύση.

Στο άρθρο το οποίο γράφτηκε απο τους Chen and Shih [25], αρχικά χρησιμοποιούνται δύο εξισώσεις για τον υπολογισμό της ταχύτητας και θέσης του σωματιδίου οι οποίες είναι:

$$V_{id}^{t+1} = V_{id}^{t} + c_1 \cdot \text{RAND1} \cdot (P_{id} - X_{id}) + c_2 \cdot \text{RAND2} \cdot (P_{gd} - X_{id})$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1}$$

Γίνεται χρήση 2 τύπων PSO, της έκδοσης inertia weight και της έκδοσης constriction. Ο πρώτος τύπος χρησιμοποιείται για να εξισορροπήσει την ικανότητα συνολικής αναζήτησης (global search) και την ικανότητα τοπικής αναζήτησης (local search) αυξάνοντας τις πιθανότητες για την εύρεση της βέλτιστης λύσης. Για το λόγο αυτό προστίθεται η τιμή w στη παραπάνω εξίσωση:

$$V_{id}^{t+1} = w * V_{id}^{t} + c_1 \cdot \text{RAND1} \cdot (P_{id} - X_{id}) + c_2 \cdot \text{RAND2} \cdot (P_{gd} - X_{id})$$

Ο δεύτερος τύπος χρησιμοποιείται για να επιτευχθεί όσο τον δυνατόν πιο σταθερή σύγκλιση και αυτό επιτυγχάνεται μόνο όταν η μέγιστη ταχύτητα περιορίζεται

στο εύρος της μέγιστης θέσης:

$$V_{max} = X_{max}$$

Έτσι οι Bratton και Kennedy αργότερα το όρισαν ως Standard PSO (SPSO) [26] και η πρώτη εξίσωση μετατρέπεται :

$$V_{id}^{t+1} = K * [V_{id}^{t} + c_1 \cdot \text{RAND1} \cdot (P_{id} - X_{id}) + c_2 \cdot \text{RAND2} \cdot (P_{gd} - X_{id})]$$

όπου

$$K = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4}}$$

Για τις εξισώσεις:

- V_{id} - είναι συνιστώσα της ταχύτητας του i-οστού σωματιδιου στην d-οστή διάσταση.
- $_{id}$ - είναι συνιστώσα της θέσης του i-οστού σωματιδιου στην d-οστή διάσταση.
- Το c_1 είναι ο γνωστικός παράγοντας μάθησης.
- Το c_2 είναι ο παράγοντας κοινωνικής μάθησης.
- P_{id} - είναι συνιστώσα της θέσης του P_{best} (Personal best) του i-οστού σωματιδιου στην d-οστή διάσταση.
- P_{gd} - είναι συνιστώσα της θέσης του G_{best} (Global best) στην d-οστή διάσταση.
- Rand() είναι ένας τυχαίος αριθμός ανάμεσα στο [0, 1].
- K είναι ο παράγοντας συστολής και φ το άθροισμα των c_1 και c_2.

Στη έρευνα αυτή χρησιμοποιήθηκαν 16 καθηγητες, 10 μαθητές και 10 αίθουσες και οι διδακτικές ώρες είναι 40 απο 8 ώρες τη μέρα, οι οποίες χωρίζονται χωρίζονται σε 20 διδακτικές περιόδους καθώς αποδείχθηκε ότι είναι πιο αποδοτικό με αυτό το τρόπο.Οι λύσεις που βρέθηκαν μπόρεσαν να ικανοποιήσουν τις προτιμήσεις των καθηγητών και των μαθητών ως προς το πρόγραμμα. Όποιο πρόβλημα δημιουργήθηκε μεταξύ των προγραμμάτων των δασκάλων, των προγραμμάτων των μαθημάτων ή των προγραμμάτων των εκάστοτε αιθουσών αντιμετωπίστηκε με επιτυχία. Επιπλέον, η πρόωρη σύγκλιση αποφεύχθηκε με την προσθήκη ενός μηχανισμού τοπικής

αναζήτησης (local search). Γίνεται αντιληπτό ότι ο αλγόριθμος PSO είναι ένα πολυ χρήσιμο εργαλείο στη δημιουργία χρονοδιαγραμμάτων κάθως τα αποτελέσματα που προέκυψαν είναι αρκετά ενθαρυντικά και αποτελώντας μία πολύ καλή εναλλακτική στη λύση αυτού του δύσκολου προβλήματος.

2.1.5 Ο αλγόριθμος GRASP

Η Άπληστη Τυχαία Διαδικασία Αναζήτησης (επίσης γνωστή ως GRASP), αναπτύχθηκε από τους Moura και Scaraficci (2010) και είναι ένας μεταευρετικός αλγόριθμος που εφαρμόζεται συνήθως σε προβλήματα βελτιστοποίησης, όπως ο χρονοπρογραμματισμός ωρολογίου προγράμματος [27]. Ο αλγόριθμος GRASP επαναλαμβάνει τρία πολύ βασικά βήματα. Αρχικά επιλέγει τυχαία κάποιο μάθημα και ορίζει τους πόρους που διαθέτει, στη συνέχεια προσθέτει το δεύτερο μάθημα και το ταξινομεί σε σχέση με το πρώτο, και έπειτα προχωράει στα υπόλοιπα μαθήματα με τον ίδιο τρόπο. Η αναπτυσσόμενη λίστα μαθημάτων ταξινομεί τα μαθήματα που έχουν μεγαλύτερο "βάρος" με την έννοια ότι διαφορετικά κριτήρια λαμβάνονται υπόψη κάθε φορά. Αυτό το στάδιο ονομάζεται "άπληστο" (greedy). Η προσθήκη κάθε επόμενου μαθήματος θα πρέπει να προσαρμόζεται στη υπάρχουσα λίστα και να ταξινομείται μαζί με τα υπόλοιπα μαθήματα. Το δεύτερο στάδιο βελτιώνει τη λίστα κατάταξης, χρησιμοποιώντας μια τοπική διαδικασία αναζήτησης (local search), η οποία συγκρίνει γειτονικά μαθήματα και τα ταξινομεί ανάλογα. Αυτή η διαδικασία συνεχίζεται μέχρι να μην υπάρχουν άλλες βελτιώσεις. Στο τελικό στάδιο χρησιμοποιείται μία στρατηγική για να αναδείξει τις βέλτιστες λύσεις, οι οποίες στη συνέχεια χρησιμοποιούνται σαν "οδηγοί" στην τελική λύση. Αυτός ο κύκλος επαναλαμβάνεται, έως ότου τοποθετηθούν όλα τα μαθήματα στη λίστα.

Στη διπλωματική της Ελένης Λεβέντη, που αφορά τη μελέτη και δημιουργία εφαρμογής για να κατασκευαστεί ωρολόγιο πρόγραμμα μαθημάτων στο Πανεπιστήμιο Δυτικής Μακεδονίας [28]. Αρχικά επιλύονται 3 προβλήματα διαβαθμισμένης δυσκολίας για τη δημιουργία προγράμματος, ο αριθμός μαθημάτων είναι μεταβλητός.

- Εύκολο - 3 αίθουσες, 10 καθηγητες και ώρες διδασκαλίας 42 και οι χρονοθυρίδες είναι 11 τη μέρα (11 * 5 = 55). Άρα 55 * 3 =165 πιθανές χρονοθυρίδες.
- Μέτριο - 6 αίθουσες , 25 καθηγητες και ώρες διδασκαλίας 100 και οι χρονο-

θυρίδες είναι 11 τη μέρα (11 * 5 = 55). Άρα 55 * 6 =330 πιθανές χρονοθυρίδες.

- Δύσκολο - 8 αίθουσες , 40 καθηγητες και ώρες διδασκαλίας 160 και οι χρονοθυρίδες είναι 11 τη μέρα (11 * 5 = 55). Άρα 55 * 8 =440 πιθανές χρονοθυρίδες.

Τοποθετούνται τα δεδομένα είτε απο το πληκτρολόγιο είτε από έτοιμο αρχείο. Στη συνέχεια με τη συνάρτηση sorting(), ταξινομούνται οι χρονοθυρίδες σε αύξουσα ή φθίνουσα σειρά και στη συνέχεια επιλέγεται ένας ευρετικός αλγόριθμος από τους 7 διαθέσιμους :

1. Αύξουσα ταξινόμηση συνολικών διαθέσιμων χρονοθυρίδων διδασκαλίας κάθε καθηγητή.

2. Αύξουσα ταξινόμηση συνολικών ωρών διδασκαλίας κάθε καθηγητή.

3. Συνδυασμός 1 και 2. Δηλαδή, αύξουσα ταξινόμηση σύμφωνα με τον αριθμό που προκύπτει αν αφαιρέσουμε τις συνολικές ώρες διδασκαλίας από τις συνολικές διαθέσιμες κάθε καθηγητή.

4. Αύξουσα ταξινόμηση εξαμήνου, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 1ου εξαμήνου, ανά καθηγητή.

5. Φθίνουσα ταξινόμηση εξαμήνων, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 10ου εξαμήνου.

6. Αύξουσα ταξινόμηση εξαμήνων, δηλαδή έχουν προτεραιότητα καταχώρησης τα μαθήματα του 1ου εξαμήνου.

7. Επιλέγει τυχαία τη σειρά που θα καταχωρηθούν οι καθηγητές στο πρόγραμμα.

Αφού γίνει η επιλογή με ποιον ευρετικό θα λυθεί, ο χρήστης καλείται να επιλέξει ανάμεσα σε ακόμα 2 ευρετικούς αλγορίθμους. Στη πρώτη περίπτωση, η εφαρμογή επιχειρεί να καταχωρήσει στο ωρολόγιο πρόγραμμα πρώτα όλα τα μαθήματα για τον πρώτο καθηγητή, στη συνέχεια το ίδιο για τον επόμενο ενώ στη δεύτερη περίπτωση η εφαρμογή επιχειρεί να καταχωρήσει στο ωρολόγιο πρόγραμμα, το πρώτο μάθημα του πρώτου καθηγητή, στη συνέχεια το πρώτο μάθημα του επόμενου καθηγητή. Αφού ολοκληρωθεί και αυτό το βήμα εμφανίζονται τα αποτελέσματα.

Αυτό που παρατηρείται απο την επίλυση του εύκολου προβλήματος είναι ότι οι ευρετικοί αλγόριθμοι 1 και 2 εκτελούνται σε μικρότερο χρόνο σε σχέση με τους υπόλοιπους και ειδικότερα από τον 7 ευρετικό αλγόριθμο καθώς η συνάρτηση του καλείται πάρα πολλές φορές μέχρι να καταχωρηθούν όλα τα μαθήματα.

Όσο αφορά το μέτριο πρόβλημα, οι χρόνοι εκτέλεσης των ευρετικών αλγορίθμων αυξήθηκαν, σε σχέση με το εύκολο πρόβλημα, και αυτό είναι λογικό λόγω του μεγαλύτερου πλήθους δεδομένων που περιέχει.

Τέλος στο δύσκολο πρόβλημα, παρατηρούμε αντίστοιχα με πριν ότι οι χρόνοι εκτέλεσης αυξήθηκαν, εφόσον το πλήθος δεδομένων γίνεται πολύ μεγαλύτερο. Το θέμα που προκύπτει είναι ότι κανένας ευρετικός αλγόριθμος δε κατάφερε να λύσει το πρόβλημα στο 100% οπότε προστέθηκαν παραπάνω προσομοιώσεις με πολύ περισσότερες ώρες διδασκαλίας και παρατηρήθηκε σημαντική βελτίωση με το ποσοστό ανάθεσης να φτάνει το 97.99%.

Το συμπέρασμα που προκύπτει είναι ότι πρόγραμμα είναι σε θέση να καλύψει πλήρως τις απαιτήσεις του μέτριου προβλήματος με μεσαίο αριθμό καθηγητών. Στην περίπτωση επιλογής μεγάλου αριθμού καθηγητών παρατηρήθηκε ότι για την πλήρη καταχώρηση των μαθημάτων με βάση τις δηλωθείσες ώρες των καθηγητών, απαιτείται οι ώρες αυτές να είναι τουλάχιστον οι τετραπλάσιες σε σύγκριση με τις ώρες διδασκαλίας.

Κεφάλαιο 3

Υλοποίηση

3.1 Περιορισμοί του προβλήματος

Μετά την ανάλυση αρκετών αλγορίθμων που λύνουν το πρόβλημα του χρονοπρογραμματισμού σε αυτό το κεφάλαιο θα γίνει ανάλυση των αλγορίθμων που χρησιμοποιήθηκαν σε αυτήν την εργασία. Οι αλγόριθμοι αυτοί είναι ο Γενετικός αλγόριθμος (Genetic algorithms) και ο αλγόριθμος ικανοποίησης περιορισμών(Constraint satisfaction problem). Οι αλγόριθμοι CSP ειναι 6 ειδών (Default backtracking, Lcv (Least Constraining Value), Mrv (Minimum remaining values), Degree Heuristic, Forward Checking, Constraint Propagation) και 2 οι γενετικοί (GA,Timetable).

Για τη σωστή λειτουργία των αλγορίθμων εφαρμόστηκαν ορισμένοι περιορισμοί, μερικοί απο τους οποίους είναι αυστηροί (Hard constraints) που σημαίνει ότι δεν επιτρέπεται να παραβιαστούν και άλλοι είναι ελαστικοί (Soft constraints) και βοηθούν ώστε να γίνει σύγκριση μεταξύ των αλγορίθμων.

Σε αυτή τη περίπτωση οι αυστηροί περιορισμοί είναι οι εξής:

- Μαθήματα του ίδιου έτους (διαλέξεις ή εργαστήρια) να μην συμπίπτουν την ίδια ώρα.
- Να μην γίνεται ανάθεση μαθημάτων σε αίθουσες με αριθμό μαθητών μεγαλύτερο από τη χωρητικότητα της εκάστοτε αίθουσας.
- Να μην διδάσκονται ταυτόχρονα μαθήματα από τον ίδιο καθηγητή.

Οι ελαστικοί περιορισμοί είναι οι εξής:

- Να μην διδάσκονται περισσότερα από 3 Μαθήματα του ίδιου έτους (διαλέξεις ή εργαστήρια) μέσα στη διάρκεια 1 μέρας.

- Να μην γίνονται περισσότερες από 1 διαλέξεις για το ίδιο μάθημα μέσα στη διάρκεια 1 μέρας.
- Ποσοστά χρήσης αιθουσών.

Η χρήση περιορισμών είναι απαραίτητη καθώς αυξάνουν τις πιθανότητες εύρεσης βέλτιστης λύσης για τον κάθε αλγόριθμο.

3.2 Ανάλυση αλγορίθμων CSP

Για την κατανόηση των CSP αλγορίθμων που χρησιμοποιήθηκαν σε αυτήν την εργασία θα εξηγηθούν ορισμένες μεταβλητές:

- **VARIABLES** : μαθήματα (διαλέξεις ή εργαστήριο).
- **DOMAINS** : οι πιθανές εβδομαδιαίες χρονικές περίοδοι.
- **CONSTRAINTS** : οι περιορισμοί του προβλήματος.
- **ROOMS** : οι αίθουσες διδασκαλίας οι οποίες περιέχουν την ονομασία της αίθουσας και τη χωρητικότητα τους.
- **DAYS** : οι 5 μέρες τις εβδομάδας.
- **INSTRUCTORS** : οι καθηγητές του πανεπιστημίου.
- **SUBJ_TEACH** : κάθε μάθημα και οι καθηγητές που το διδάσκουν.
- **SUBJECTS** : οι τίτλοι των μαθημάτων, η προσέλευση των φοιτητών στο εκάστοτε μάθημα και ο αριθμός των εργαστηρίων.
- **SPECIALITIES** : ο διαχωριμός μαθημάτων σε 5 έτη και τα μαθήματα που αντιστοιχούν στο κάθε έτος.

Οι αλγόριθμοι που χρησιμοποιήθηκαν ορισμένοι αυτούσια και ορισμένοι με αλλαγές φαίνονται στον παρακάτω σύνδεσμο: `https://github.com/vrshchk/timetableCSP`

3.2.1 Default Backtracking

Το Backtracking είναι ένας αναδρομικός αλγόριθμος αναζήτησης που χρησιμοποιείται για την εύρεση λύσεων σε CSP (Constraint satisfaction problems) [29]. Αρχικά πραγματοποιείται εκχώρηση χρονοθυρίδων κατά την οποία επιλέγεται ένα μάθημα και αντιστοιχείται σε μια διαθέσιμη χρονοθυρίδα. Στη συνέχεια γίνεται έλεγχος περιορισμών που σημαίνει ότι η εκχώρηση μαθημάτων δεν παραβιάζει κανέναν περιορισμό, το επόμενο στάδιο είναι η αναδρομική αναζήτηση όπου επαναλαμβάνεται η ίδια διαδικασία για το επόμενο μάθημα και σε περίπτωση που δεν μπορεί να γίνει έγκυρη ανάθεση για ένα μάθημα, επιστρέφει στο προηγούμενο και δοκιμάζεται η επόμενη διαθέσιμη θέση.

Δημιουργήθηκε ένας ψευδοκώδικας για τον αλγόριθμο default backtracking απεικόνηση 3.1 όπου φαίνεται η ευρετική συνάρτηση (heuristic) η οποία επιλέγει όλα τα μαθήματα τα οποία δεν έχουν τοποθετηθεί σε κάποια χρονοθυρίδα, ώστε στη συνέχεια στη συνάρτηση backtracking να πραγματοποιηθούν οι απαραίτητοι έλεγχοι για τη δημιουργία του χρονοδιαγράμματος.

Απεικόνιση 3.1: Pseudocode Backtracking

```
FUNCTION backtracking(assignment, csp, heuristic):
    SET rnd_domains_back TO copy of csp[DOMAINS]
    WHILE True:
        IF assignment is complete then
            RETURN assignment
        SHUFFLE rnd_domains_back
        FOR each value IN rnd_domains_back do
            SET class TO heuristic(assignment)
            ASSIGN value TO assignment[class]
            SET class._room TO getRoom(csp, assignment, class, value)
            INCREMENT counter
            IF assignment is consistent WITH csp[CONSTRAINTS] then
                BREAK
             ELSE:
```

```
                UNASSIGN assignment[class]
                RESET class._room
    RETURN FAILURE

FUNCTION heuristic(assignment):
    Initialize empty list res
    For each i in csp[VARIABLES]:
        If assignment[i] is None:
            Append i to res
    RETURN res element at index generated by random integer
    between 0 and length of res minus 1
```

3.2.2 Least Constraining Value - LCV

Ο ευρετικός αλγόριθμος LCV επιλέγει την τιμή που θα αφήσει τον μέγιστο αριθμό επιλογών για τις υπόλοιπες μεταβλητές του προβλήματος και είναι ιδιαίτερα χρήσιμος γιατί μεγιστοποιεί τη μελλοντική ευελιξία [29]. Για παράδειγμα, κατά την αντιστοίχιση χρονοθυρίδων σε μαθήματα, ο LCV θα έδινε προτεραιότητα σε χρονοθυρίδες που αφήνουν τις περισσότερες διαθέσιμες επιλογές για τον προγραμματισμό άλλων μαθημάτων, μειώνοντας τις πιθανές εμφάνισεις προβλημάτων. Επίσης ο ευρετικός αυτός αλγόριθμος, βοηθά στην πιο αποτελεσματική πλοήγηση στον χώρο αναζήτησης, αυξάνοντας έτσι την πιθανότητα εύρεσης ενός έγκυρου και βέλτιστου χρονοδιαγράμματος.

Δημιουργήθηκε ένας ψευδοκώδικας για τον ευρετικό αλγόριθμο LCV απεικόνηση 3.2 ο οποίος στη συνάρτηση lcv_heuristic επιλέγεται κάθε φορά ο καθηγητής με τα λιγότερα μαθήματα ως ευρετική μέθοδος και στη συνέχεια τοποθετούνται τα μαθήματα στα κατάλληλα timeslots και rooms ώστε να κατασκευαστεί το χρονοδιάγραμμα.

Απεικόνιση 3.2: Pseudocode LCV

```
FUNCTION backtracking_lcv(assignment, csp, heuristic):
    SET rnd_domains_back TO copy of csp[DOMAINS]
```

```
    WHILE True:
        IF assignment is complete then
            RETURN assignment
        SHUFFLE rnd_domains_back
        FOR each value IN rnd_domains_back do
            SET class TO heuristic(assignment)
            ASSIGN value TO assignment[class]
            SET class._room TO getRoom(csp, assignment, class, value)
            INCREMENT counter
            IF assignment is consistent WITH csp[CONSTRAINTS] then
                BREAK
             ELSE:
                UNASSIGN assignment[class]
                RESET class._room
    RETURN FAILURE

FUNCTION lcv_heuristic(assignment):
    Initialize empty dictionary teach
    For each i in csp[VARIABLES]:
        Set teach[i._teacher] to 0
    Initialize empty list res
    For each i in csp[VARIABLES]:
        If assignment[i] is None:
            Increment teach[i._teacher] by 1
            Append i to res
    Sort res by the value of teach[l._teacher] for each element l
    Return the first element of res
```

3.2.3 Minimum Remaining Values - MRV

Ο ευρετικός αλγόριθμος MRV δίνει προτεραιότητα στη μεταβλητή με τις λιγότερες επιτρεπτές τιμές που απομένουν, που σημαίνει ότι επιλέγει πρώτα την εργασία ή τη μεταβλητή που είναι πιο περιορισμένη και έχει τη λιγότερη ευελιξία [29].

Εστιάζοντας στις πιο περιορισμένες μεταβλητές, το MRV στοχεύει να μειώσει την πολυπλοκότητα του προβλήματος από νωρίς, αποφεύγοντας πιθανά προβλήματα και αδιέξοδα αργότερα στη διαδικασία προγραμματισμού. Αυτή η προσέγγιση ενισχύει την αποτελεσματικότητα της εύρεσης λύσης, καθιστώντας την ένα πολύτιμο εργαλείο για τη βελτιστοποίηση των χρονοδιαγραμμάτων.

Δημιουργήθηκε ένας ψευδοκώδικας για τον ευρετικό αλγόριθμο MRV απεικόνηση 3.3 ο οποίος στη συνάρτηση find_mrv επιλέγεται κάθε φορά το μάθημα με τα λιγότερα πιθανά timeslots διαθέσιμα, ώστε να ξεκινήσει από το πιο απαιτητικό πρόβλημα και να βρεθεί πιο εύκολα η λύση. Στη συνέχεια στη συνάρτηση mrv_backtracking παίρνοντας τα αποτελέσματα από τη find mrv και την ικανοποίηση όλων των περιορισμών κατασκευάζεται το χρονοδιάγραμμα.

Απεικόνιση 3.3: Pseudocode MRV

```
FUNCTION mrv_backtracking(assignment, csp):
  Set global variables counter and mrv_domains
  Copy csp[DOMAINS] to rnd_domains_m
  If assignment is complete:
    Return assignment
  Set class as the result of find_mrv(assignment, csp)
  Shuffle rnd_domains_m
  For each value in rnd_domains_m:
    If value is in mrv_domains[class]:
        Set assignment[class] as value
        Increment counter by 1
        Call add_to_mrv_domains(assignment, csp, class)
        Set class._room as the result of getRoom(csp,assignment,
        class,value)
        If assignment is consistent with csp[CONSTRAINTS]:
            Set result as the result of mrv_backtracking(assignment,
            csp)
            If result is not FAILURE:
                Return result
```

```
        Set assignment[class] as None
         Call undo(assignment, csp)
  Return FAILURE

_____________________________________________________
Function find_mrv(assignment, csp):
  Set min_val to the result of select_unassigned_variable(csp
  [VARIABLES], assignment)
  Set min_domain to the result of domain_len(mrv_domains[min_val])

  For each variable i in csp[VARIABLES]:
    If assignment[i] is None:
        If domain_len(mrv_domains[i]) is less than min_domain:
            Set min_val to i
            Set min_domain to domain_len(mrv_domains[min_val])
    Return min_val
```

3.2.4 Degree Heuristic

Ο ευρετικός αλγόριθμος Degree heuristic στον προγραμματισμό είναι μια μέθοδος που χρησιμοποιείται για την επιλογή της μεταβλητής που είναι πιο πιθανό να προκαλέσει αποτυχία, προκειμένου να αποτύχει όσο το δυνατόν νωρίτερα [30]. Στο πλαίσιο ενός προβλήματος δημιουργίας χρονοδιαγράμματος, μπορεί να επιλεγεί πρώτα ένα μάθημα που μπορεί να διδαχθεί μόνο από καθηγητή που είναι και ο μόνος που μπορεί να διδάξει αρκετά ακόμα μαθήματα.

Δημιουργήθηκε ένας ψευδοκώδικας για τον ευρετικό αλγόριθμο Degree Heuristic απεικόνηση 3.4 ο οποίος στη συνάρτηση choose_values επιστρέφει το μάθημα με το μεγαλύτερο degree value (το μάθημα με τους περισσότερους περιορισμούς) που δεν έχει καταχωρηθεί ακόμα και στη συνέχεια στη συνάρτηση degree_backtracking γίνονται όλοι οι απαραίτητοι έλεγχοι για να τοποθετηθούν όλα τα μαθήματα σε χρονοθυρίδες και να κατασκευαστεί το χρονοδιάγραμμα.

Απεικόνιση 3.4: Pseudocode Degree Heuristic

```
Function degree_backtracking(assignment, csp):
    Copy csp[DOMAINS] to rnd_domains_d
    While True:
        If assignment is complete:
            Return assignment
        Randomly shuffle rnd_domains_d
        For each value in rnd_domains_d:
            Set class to the result of choose_value(assignment,csp)
            Assign value to assignment[class]
            Update values using update_values(assignment,csp)
            Set class._room to the result of getRoom(csp,
            assignment, class, value)
            Increment counter by 1
            If assignment is consistent with csp[CONSTRAINTS]:
                Break
            Else:
                Set assignment[class] to None
                Set class._room to None
                Update values using update_values(assignment,csp)
    Return FAILURE


Function choose_value takes in parameters assignment and csp:
    Access global variable degree_values
    Sort degree_values in descending order based on the second
    element of each sublist
    For each element 'i' in degree_values:
        If the value of the key 'i[0]' in assignment is None:
            Return 'i[0]'
```

3.2.5 Forward Checking

Ο αλγόριθμος Forward Checking στο χρονοπρογραμματισμό εξετάζει προληπτικά τα υπόλοιπα μη εκχωρημένα μαθήματα και αφαιρεί οποιαδήποτε μαθήματα θα μπορούσαν να προκαλέσουν πρόβλημα με την τρέχουσα εκχώρηση [29].

Δημιουργήθηκε ένας ψευδοκώδικας για τον αλγόριθμο Forward Checking απεικόνηση 3.5 ο οποίος στη συνάρτηση forward_checking επιλέγει μια μη εκχωρημένη μεταβλητή και ελέγχει εάν καταλήγει σε μια συνεπή εκχώρηση. Αν το κάνει, συνεχίζει με το επόμενο μάθημα, εάν δεν το κάνει, αναιρεί την εκχώρηση και δοκιμάζει άλλο μάθημα. Συνεχίζει αυτή τη διαδικασία μέχρι να βρεθεί λύση.

Απεικόνιση 3.5: Pseudocode Forward Checking

```
Function forward_checking(assignment, csp):
    Declare global variables var_domains and counter
    Make a copy of csp[DOMAINS] and assign it to rnd_domains_f
    While True:
        If assignment is complete:
            Return assignment
        Select an unassigned variable and assign it to class
        Shuffle rnd_domains_f
        For each value in rnd_domains_f:
          If value is in the domain of class:
            Assign value to class in assignment
            Update domains based on the current assignment and class
            Assign a room to class based on the current assignment and
            value
            Increment counter by 1
            If the current assignment is consistent with the constraints:
                Break the loop
              Else:
                Remove the assignment of class
                Remove the room of class
```

```
                Undo the assignment
    If no valid assignment is found, Return FAILURE
```

3.2.6 Constraint Propagation

Ο αλγόριθμος Constraint Propagation επιλέγει κάποιο μάθημα που δεν έχει επιλεχθεί και του δίνει κάποια διαθέσιμη χρονοθυρίδα που δεν παραβιάζει κανέναν περιορισμό [29]. Εάν η ανάθεση οδηγήσει σε μελλοντικό πρόβλημα παραβίασης περιορισμών, . Η διαδικασία συνεχίζεται μέχρι να βρεθεί ένα έγκυρο πρόγραμμα.

Δημιουργήθηκε ένας ψευδοκώδικας για τον αλγόριθμο Constraint Propagation απεικόνηση 3.6 ο οποίος στη συνάρτηση constraint_propagation επιλέγει ένα μάθημα που δεν έχει χρησιμοποιηθεί ακόμα και πραγματοποιείται έλεγχος αν στις διαθέσιμες χρονοθυρίδες παραβιάζονται οι περιορισμοί του προβλήματος, αν ναι αφαιρεί το μάθημα που έχει επιλεγεί να γίνει ανάθεση και ξεκινάει την ίδια διαδικασία με το επόμενο έως ότου ολοκληρωθεί η κατασκευή του ωρολογίου προγράμματος χωρίς κάποιο σφάλμα.

Απεικόνιση 3.6: Pseudocode Constraint Propagation

```
Function constraint_propagation(assignment, csp):
    Declare global variables var_domains and counter
    Make a copy of csp[DOMAINS] and assign it to rnd_domains_p
    While True:
        If assignment is complete:
            Return assignment
        Select an unassigned variable and assign it to class
        Shuffle rnd_domains_p
        For each value in rnd_domains_p:
            If value is in the domain of class:
                Assign value to class in assignment
                Update domains based on the current assignment and class
                If check_for_zero(csp) returns True:
                  Assign a room to class based on the current assignment
```

```
                and value
                Increment counter by 1
                If the current assignment is consistent with
                the constraints:
                  Break the loop
                Else:
                  Remove the assignment of class
                  Remove the room of class
                  Undo the assignment
              Else:
                Remove the assignment of class
    If no valid assignment is found, Return FAILURE
```

3.3 Ανάλυση γενετικών αλγορίθμων

Οι Γενετικοί αλγόριθμοι που χρησιμοποιήθηκαν σε αυτή τη εργασία είναι δύο: ο Timetable και ο GA. Αρχικά θα μελετηθεί ο Timetable και θα εξηγηθεί ο τρόπος που λειτουργεί καικαι στη συνέχεια θα ακολουθηθεί η ίδια διαδικασία και για τον GA αλγόριθμο.

3.3.1 Ο γενετικός αλγόριθμος Timetable

Για τη δημιουργία χρονοδιαγράμματος χρησιμοποιούνται οι παρακάτω μεταβλητές και συναρτήσεις:

- η συνάρτηση **addRoom** (roomId, roomname, capacity) η οποία αποτελείται απο το id του κάθε δωματίου, την ονομασία του και τη χωρητικότητα του.
- η συνάρτηση **addProfessor** (professorId, professorName, preferedroom=0, preferedtime=0) η οποία αποτελείται απο το id του κάθε καθηγητή, το όνομά του, την αίθουσα που προτιμάει και τη χρονοθυρίδα που προτιμάει.
- η συνάρτηση **addCourse** (courseId, courseCode, course, professorIds,numStu) η οποία αποτελείται απο το id του κάθε μαθήματος, μια συντομογραφία του

εκάστοτε μαθήματος, την ονομασία του, τους καθηγητές που το διδάσκουν και τη προσέλευση φοιτητών στο κάθε μάθημα.

- η συνάρτηση **addGroup** (groupId, groupname, courseIds) η οποία αποτελείται από το id το οποίο δηλώνει το έτος, την ονομασία για το κάθε έτος και τα id των μαθημάτων κατανεμημένα σε όλα τα έτη.

- η συνάρτηση **addTimeslot** (timeslotId, timeslot) η οποία αποτελείται απο το id για τη εκάστοτε χρονοθυρίδα και οι χρονοθυρίδες για κάθε μέρα της εβδομάδος.

Ο αλγόριθμος που χρησιμοποιήθηκε που είναι φτιαγμένος (άπο java σε python) φαίνεται στον σύνδεσμο: `https://github.com/janhavibhalerao/class-scheduler-ai/tree/master`

Selection

Αρχικά ο αλγόριθμος πραγματοποιεί τη διαδικασία της επιλογής (selection) κατά την οποία επιλέγει ένα άτομο (individual) από τον κύριο πληθυσμό (main population) για να προχωρήσει σε διασταύρωση (crossover) στον γενετικό αλγόριθμο. Η επιλογή αυτή γίνεται με τη δημιουργία ενός τουρνουά (tournament) και με την τυχαία επιλογή ενός ατόμου (individual). Στη συνέχεια επιλέγεται το fittest individual για να προχωρήσει στη διαδικασία του crossover.

Crossover

Ο σκοπός της μεθόδου crossoverPopulation είναι η δημιουργία ενός νέου πληθυσμού (Population) συνδυάζοντας τις γενετικές πληροφορίες των γονικών ατόμων (parent) μέσω διασταύρωσης. Αυτό βοηθά στη δημιουργία νέων λύσεων (offsprings) που μπορεί να έχουν καλύτερο fitness κληρονομώντας ευεργετικά χαρακτηριστικά και από τους δύο γονείς. Αυτή η μέθοδος διασφαλίζει επίσης ότι ένας συγκεκριμένος αριθμός, αυτών με το υψηλότερο fitness (elite individuals) μεταβιβάζονται απευθείας στην επόμενη γενιά χωρίς να υποστούν διασταύρωση (crossover).

Mutation

Ο σκοπός της μεθόδου mutatingPopulation είναι να εισαγάγει γενετική ποικιλότητα στον πληθυσμό αλλάζοντας τυχαία τα γονίδια των ατόμων (individuals). Αυτό βοηθά στην εξερεύνηση νέων λύσεων και εμποδίζει τον γενετικό αλγόριθμο να κολλήσει στο τοπικό βέλτιστο. Αυτή η μέθοδος διασφαλίζει ότι ένας συγκεκριμένος αριθμός, αυτών με το υψηλότερο fitness (elite individuals) μεταβιβάζονται απευθείας στην επόμενη γενιά χωρίς να υποστούν μετάλλαξη.

Ψευδοκώδικας του αλγορίθμου Timetable

Με τη δημιουργία ψευδοκώδικα 3.7 φαίνεται αναλυτικά η λειτουργία του Γενετικού αλγορίθμου:

Απεικόνιση 3.7: Pseudocode Timetable

```
FUNCTION selectFromPopulation(population):
    CREATE tournament WITH size tournSize
    SHUFFLE population
    FOR i FROM 0 TO tournSize - 1:
        ADD population[i] TO tournament
    RETURN fittestIndividual FROM tournament
-----------------------------------------------
FUNCTION crossoverPopulation(population):
    CREATE newPopulation WITH size population.size()
    FOR i FROM 0 TO population.size() - 1:
        SET parent1 TO i-th fittest individual FROM population
        IF crossoverConditionMet AND i > count:
            CREATE offspring
            INIT offspring WITH parent1's chromosome length
            SET parent2 TO selectFromPopulation(population)
            FOR j FROM 0 TO parent1.chromosomeLength - 1:
                IF randomValue < 0.5:
                    SET offspring[j] TO parent1[j]
```

```
                ELSE:
                    SET offspring[j] TO parent2[j]
            ADD offspring TO newPopulation
        ELSE:
            ADD parent1 TO newPopulation
    RETURN newPopulation
-----------------------------------------
FUNCTION mutatePopulation(population, schedule):
    CREATE newPopulation WITH size popSize
    SET bestFitness TO fitness OF fittestIndividual FROM population
    FOR i FROM 0 TO population.size() - 1:
        SET individual TO  i-th fittest individual FROM population
        CREATE randomIndividual
        INIT randomIndividual WITH schedule
        FOR j FROM 0 TO individual.chromosomeLength - 1:
            IF i > count AND randomValue < rateOfMutation:
                SET j-th individual OF population TO randomIndividual
        ADD individual TO newPopulation
    RETURN newPopulation
```

3.3.2 Ο Γενετικός αλγόριθμος GA

Για τη δημιουργία χρονοδιαγράμματος χρησιμοποιήθηκαν σε αυτήν την εργασία ορισμένες μεταβλητές:

- **ROOMS** : οι αίθουσες διδασκαλίας οι οποίες περιέχουν την ονομασία της αίθουσας και τη χωρητικότητά τους.
- **MEETING TIMES** : οι 5 μέρες τις εβδομάδας και οι χρονοθυρίδες τους.
- **INSTRUCTORS** : οι καθηγητές του πανεπιστημίου.
- **SUBJECT** : τα id των μαθημάτων, οι τίτλοι των μαθημάτων, η προσέλευση των φοιτητών στο εκάστοτε μάθημα και οι καθηγητές που τα διδάσκουν.

- **SPECIALITIES** : ο διαχωριμός μαθημάτων σε 5 έτη και τα μαθήματα που αντιστοιχούν στο κάθε έτος.

Ο αλγόριθμος που χρησιμοποιήθηκε φαίνεται στον σύνδεσμο: `https://github.com/vrshchk/timetableGA`

Evolve

Η συνάρτηση evolve πραγματοποιεί crossover στον πληθυσμό που δόθηκε και στη συνέχεια γίνεται mutate στο αποτέλεσμα που προκύπτει από το crossover.

Crossover

Η συνάρτηση _Crossover_population δημιουργεί έναν άδειο πληθυσμό (population) ,επιλέγει δύο χρονοδιαγράμματα (schedules) και τα συνδιάζει ώστε να προκύψει ένα καινούργιο χρονοδιάγραμμα μέσο της συνάρτησης _crossover_schedule και επιστρέφει το νέο πληθυσμό. Η συνάρτηση Crossover schedule δημιουργεί ένα νέο χρονοδιάγραμμα και για κάθε μάθημα αυτού επιλέγει τυχαία κάποιο μάθημα απο τα δυο χρονοδιαγράμματα της crossover population και το καταχωρεί στο νέο και το επιστρέφει.

Select

Η συνάρτηση _select_for_change δημιουργεί έναν άδειο πληθυσμό (population) και επιλέγει τυχαία χρονοδιαγράμματα και τα προσθέτει στο νέο population. Στη συνέχεια τα ταξινομεί με βάση τη συνάρτηση health_rate (όπου ελέγχονται όλοι οι περιορισμοί που χρησιμοποιήθηκαν) σε φθίνουσα σειρά και επιστρέφει τον πληθυσμό μετά την ταξινόμηση.

Mutate

Η συνάρτηση _mutate καλεί τη συνάρτηση _mutate_one στη οποία δημιουργείται ένα νέο χρονοδιάγραμμα και για κάθε ανάθεση μαθήματος (το οποίο περιλαμβάνει την ώρα διδασκαλίας του, τον εκπαιδευτή του, την προσέλευση φοιτητών καθώς και το έτος του) πραγματοποιείται mutation με τη βοήθεια της μεταβλητής mutation_possibility και γίνεται αντικατάσταση του μαθήματος στο νέο χρονοδιάγραμμα. Τέλος επιστρέφει το νέο χρονοδιάγραμμα.

Ψευδοκώδικας του γενετικού αλγορίθμου GA

Δημιουργήθηκε ο παρακάτω ψευδοκώδικας 3.8 για τη πλήρη κατανόηση της λειτουργίας του αλγορίθμου και των συναρτήσεων που προαναφέρθηκαν.

Απεικόνιση 3.8: Pseudocode GA

```
FUNCTION evolve(population):
    RETURN mutate(crossover_population(population))
-------------------------------------------------------
FUNCTION crossover_population(pop):
    INITIALIZE crossover_pop AS new Population with size 0

    FOR i FROM 0 TO INDIVIDUALS_TO_THE_NEXT_ROUND - 1:
        ADD pop.schedules[i] TO crossover_pop.schedules

    SET i TO INDIVIDUALS_TO_THE_NEXT_ROUND

    WHILE i < INDIVIDUALS:
        SET schedule1 TO select_for_change(pop).schedules[0]
        SET schedule2 TO select_for_change(pop).schedules[0]
        ADD crossover_schedule(schedule1, schedule2) TO
        crossover_pop.schedules
        INCREMENT i BY 1

    RETURN crossover_pop
-------------------------------------------------------------------
FUNCTION mutate(population):
    FOR i FROM INDIVIDUALS_TO_THE_NEXT_ROUND TO INDIVIDUALS - 1:
        CALL mutate_one(population.schedules[i])

    RETURN population
-------------------------------------------------------------------
FUNCTION crossover_schedule(schedule1, schedule2):
```

```
    CALL new Evaluation().initialize() TO INITIALIZE crossoverSchedule

    FOR i FROM 0 TO LENGTH(crossoverSchedule.get_classes()) - 1:
        IF random() > 0.5:
            SET crossoverSchedule.get_classes()[i] TO
            schedule1.classes[i]
        ELSE:
            SET crossoverSchedule.get_classes()[i] TO
            schedule2.get_classes()[i]

    RETURN crossoverSchedule
-------------------------------------------------------------------
FUNCTION mutate_one(mutateSchedule):
    CALL new Evaluation().initialize() TO INITIALIZE schedule

    FOR i FROM 0 TO LENGTH(mutateSchedule.get_classes()) - 1:
        IF MUTATION_POSSIBILITY > random():
            SET mutateSchedule.get_classes()[i] TO
            schedule.get_classes()[i]

    RETURN mutateSchedule
-------------------------------------------------------------------
FUNCTION select_for_change(pop):
    INITIALIZE change_pop AS new Population with size 0

    SET i TO 0

    WHILE i < INDIVIDUALS_TO_CHANGE:
        ADD pop.schedules[random_range(0, INDIVIDUALS)] TO
        change_pop.schedules
        INCREMENT i BY 1
```

```
SORT change_pop.schedules BY healthRate DESCENDING

RETURN change_pop
```

Κεφάλαιο 4

Υπολογιστική μελέτη

4.1 Εισαγωγή στα αποτελέσματα και στις μετρικές

Τα δεδομένα του προβλήματος για κάθε αλγόριθμο είναι τα εξής:

Αίθουσες: Οι αίθουσες ειναι μέχρι 15 και το μέγεθος της κάθε αίθουσας είναι σταθερό.

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

Διδακτικές Περίοδοι: Οι διδακτικές περίοδοι αφορούν τις ώρες στις οποίες θα γίνονται τα μαθήματα και είναι μη μεταβλητό μέγεθος, στη συγκεκριμένη εργασία με τη παραδοχή ότι τα μαθήματα διεξάγονται από τις 9:00 μέχρι τις 21:00 και κάθε μάθημα είτε είναι εργαστήριο είτε είναι διάλεξη διαρκεί δύο ώρες, αυτό σημαίνει ότι υπάρχουν 5*6=30 διαθέσιμες χρονοθυρίδες για το κάθε μάθημα.

Μαθήματα και εργαστήρια: Τα χαρακτηριστικά του κάθε μαθήματος είναι το όνομά του, το έτος που διδάσκεται, αν είναι μάθημα επιλογής και από ποιον καθηγητή διδάσκεται. Τα μαθήματα είναι ένα μεταβλητό μέγεθος, καθώς είναι πιθανόν να αλλάξει είτε ο καθηγητής είτε να αλλάξουν οι ώρες διδασκαλίας, ακόμη και να προστεθεί ή να αφαιρεθεί ένα μάθημα.

Καθηγητές: Πρόκειται για το εκπαιδευτικό προσωπικό που διδάσκει τα μαθήματα και ο μέγιστος αριθμός καθηγητών που χρησιμοποιήθηκαν στα παραδείγματα ήταν 15. Τα χαρακτηριστικά των καθηγητών είναι το ονοματεπώνυμο, οι διαθέσιμες ώρες διδασκαλίας και τα μαθήματα που διδάσκουν.

Στον Πίνακα 4.1 παρουσιάζονται τα δεδομένα για καθένα απο τα 20 παραδείγματα που εφαρμόστηκαν σε όλους τους αλγορίθμους ξεχωριστά.

Πίνακας 4.1: 20 παραδείγματα εκτέλεσης

Παραδείγματα					
Α/Α παραδείγματος	Αριθμός μαθημάτων	Αριθμός διαλέξεων	Αριθμός εργαστηρίων	Αριθμός καθηγητών	Αριθμος αίθουσων
1	20	40	30	11	8
2	20	40	20	9	7
3	25	50	30	9	10
4	15	22	11	6	6
5	20	40	0	6	8
6	25	40	19	8	8
7	25	45	16	8	6
8	30	50	20	9	9
9	30	30	10	5	5
10	20	20	30	5	4
11	25	30	30	6	7
12	25	30	30	8	12
13	25	25	25	8	9
14	20	20	20	9	9
15	20	20	20	9	5
16	20	30	20	9	5
17	25	35	20	10	6
18	25	35	25	11	13
19	25	35	25	12	14
20	20	20	15	12	15

Ωστόσο, για να εξεταστεί πόσο καλός είναι ένας αλγόριθμος υπάρχουν διάφορες μετρικές οι οποίες βοηθούν στην αντιμετώπιση του ζητήματος του χρονοπρογραμματισμού. Σε αυτήν την εργασία χρησιμοποιήθηκαν τρεις μετρικές σύγκρισης, οι οποίες είναι:

- Ποινή (penalty) για τη διδασκαλία τριών ή περισσότερων μαθήματων είτε είναι εργαστήρια είτε διαλέξεις απο το ίδιο έτος την ίδια μέρα. Εξαιρούνται τα μαθήματα επιλογής τα οποία δεν προσμετρώνται.
- Ποινή όταν οι διαλέξεις ενός μαθήματος κάποιου έτους ξεπερνούν τη μία μέσα στην ίδια μέρα.
- Ποσοστό χρήσης αιθουσών (Utilization rate).

Στο κεφάλαιο αυτό θα γίνει παρουσίαση των αποτελεσμάτων απο κάθε μετρική και αξιολόγηση του κάθε αλγορίθμου με σκοπό την εύρεση του καλύτερου αν αυτό είναι εφικτό.

4.2 Αποτελέσματα για κάθε μετρική

Αφού εκτελεστούν όλοι οι αλγόριθμοι για κάθε ένα από τα 20 παραδείγματα που προαναφέρθηκαν, θα πρέπει να χρησιμοποιηθούν οι τρεις μετρικές της προηγούμενης υποενότητας. Αρχικά πρέπει να γίνει ο υπολογισμός ποινής για τη πρώτη μετρική. Έτσι έχουμε ακόλουθη σχέση:

$$p(\mu_i) = \begin{cases} 0, & \text{p}(\mu_i) < 4 \\ 1, & \text{p}(\mu_i) = 4 \\ 4, & \text{p}(\mu_i) = 5 \\ 7, & \text{p}(\mu_i) \geq 6 \end{cases} \tag{4.1}$$

$$tp(e) = \sum p(\mu) \tag{4.2}$$

Για τον κάθε αλγόριθμο:

- μ_i - πόσα μαθήματα παραβιάζονται σε μια μέρα [μ_1, μ_2, μ_3, μ_4, μ_5].
- p(μ) - συνολική ποινή για 1 μέρα.
- e - παράδειγμα και αποτελείται από 5 μέρες [μ_1, μ_2, μ_3, μ_4, μ_5].
- tp(e) - συνολική ποινή για ένα παράδειγμα.

Έτσι με αυτό το τρόπο γίνεται ο υπολογισμός της ποινής για την πρώτη μετρική.

Με βάση τα παραπάνω δημιουργείται ο συνολικός Πίνακας 4.2 με όλες τις πιθανές παραβιάσεις σε κάθε παράδειγμα σε όλους τους αλγορίθμους:

Αυτό που παρατηρούμε είναι ότι στο τρίτο παράδειγμα υπάρχουν αρκετά σφάλ-

Πίνακας 4.2: Ποινή για 3 ή περισσότερα μαθήματα τη μέρα ενός έτους

Ποινές για κάθε αλγόριθμο								
Α/Α παραδείγματος	Default backtracking	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	1	6	17	6	11	4	14	3
2	7	7	4	6	6	3	6	9
3	12	11	21	22	18	18	23	21
4	1	1	1	0	0	0	1	1
5	0	1	0	0	0	0	1	0
6	1	1	0	0	0	2	2	0
7	0	3	1	5	0	0	1	1
8	1	5	2	6	1	1	5	2
9	0	0	0	0	0	0	0	0
10	2	0	5	0	4	0	1	1
11	0	1	1	1	0	0	2	1
12	1	1	1	1	1	2	2	1
13	0	0	0	1	0	0	0	0
14	1	1	0	0	0	0	0	0
15	0	2	0	0	0	0	1	1
16	0	0	0	0	0	0	1	2
17	0	1	1	1	0	0	1	1
18	5	1	2	3	3	2	4	2
19	1	4	1	1	6	6	1	4
20	0	0	0	0	0	0	0	0
Μ.Ο.	2.1	2.3	2.85	2.65	2.5	1.9	3.3	2.5

ματα σε όλους τους αλγορίθμους και ο λόγος είναι ότι σε αυτό το παράδειγμα βρίσκεται ο μεγαλύτερος αριθμός διαλέξεων και εργαστηρίων καθώς και λιγότερα μαθήματα επιλογής συγκριτικά με τα υπόλοιπα.

Στη συνέχεια με τον ίδιο τρόπο υπολογίζουμε τις ποινές για τη δεύτερη μετρική (Ποινή όταν οι διαλέξεις ενός μαθήματος κάποιου έτους ξεπερνούν τη μία μέσα στη ίδια μέρα) χρησιμοποιώντας την εξής σχέση:

$$p(\mu_i) = \begin{cases} 0, & \text{p}(\mu_i) < 2 \\ 2, & \text{p}(\mu_i) = 2 \\ 6, & \text{p}(\mu_i) \geq 3 \end{cases} \tag{4.3}$$

$$tp(e) = \sum p(\mu) \tag{4.4}$$

Για τον κάθε αλγόριθμο:

- μ_i - πόσα μαθήματα παραβιάζονται σε μια μέρα [μ_1, μ_2, μ_3, μ_4, μ_5].
- p(μ) - συνολική ποινή για 1 μέρα.
- e - παράδειγμα και αποτελείται από 5 μέρες [μ_1, μ_2, μ_3, μ_4, μ_5].
- tp(e) - συνολική ποινή για ένα παράδειγμα.

Με βάση τα παραπάνω δημιουργείται ο συνολικός Πίνακας **4.3** με όλες τις πιθανές παραβιάσεις σε κάθε παρέδειγμα σε όλους τους αλγορίθμους.

Παρατηρώντας τα αποτελέσματα αυτό που γίνεται εμφανές είναι ότι σε ορισμένα παραδείγματα δεν υπάρχει κάποιο σφάλμα και ο λόγος είναι ότι δεν διδάσκονται πολλές διαλέξεις σε όλα τα μαθήματα. Αυτό δεν αναιρεί το γεγονός ότι σε γενική εικόνα οι αλγόριθμοι δίνουν ενα ικανοποιητικό αποτέλεσμα.

Εν συνεχεία στην τρίτη μετρική χρησιμοποιούμε το Utilization rate το οποίο είναι το γινόμενο του Frequency rate με το Occupancy rate. Το Frequency rate δείχνει συχνά ότι χρησιμοποιείται ένα δωμάτιο σε σχέση με τον συνολικό αριθμό των ωρών κατά τις οποίες είναι διαθέσιμο. Το Occupancy rate είναι το ποσοστό που μετράει τον βαθμό στον οποίο ένα δωμάτιο είναι πλήρως κατειλημμένο σε σχέση με τη συνολική χωρητικότητά του. Το occupancy rate βρίσκεται διαιρώντας τα άτομα που

Πίνακας 4.3: Ποινή για 2 ή περισσότερες διαλέξεις τη μέρα για το ίδιο μάθημα

Ποινές για κάθε αλγόριθμο								
Α/Α παραδείγματος	Default backtracking	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	8	12	8	8	8	4	2	6
2	10	12	8	2	6	6	4	4
3	0	4	6	16	12	8	14	10
4	4	0	2	6	4	0	2	2
5	6	6	8	6	8	6	12	6
6	8	4	6	8	4	4	8	4
7	6	10	4	12	10	2	8	2
8	8	6	6	14	6	4	12	6
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	2	0	0	0	0	0	2	0
12	0	0	2	2	2	2	2	2
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	2	2	2	4	2	0	2	0
17	0	0	6	0	0	4	6	2
18	0	4	2	2	2	2	4	2
19	4	2	4	4	6	4	4	4
20	0	0	0	0	0	0	0	0
Μ.Ο.	2.9	3.1	3.2	4.2	3.5	2.3	4.1	2.5

παρίστανται σε κάθε μάθημα αυτής της αίθουσας με τη συνολική χωριτηκότητά της και στη συχέχεια εξαγοντας το μέσο όρο. Αφού υπολογιστούν τα παραπάνω έχουμε τη σχέση: **Utilization rate = Frequency rate * Occupancy rate**

Με αυτή τη σχέση υπολογίζουμε το Utilization rate όλων των αιθουσών σε κάθε αλγόριθμο ξεχωριστά και δημιουργείται ο συνολικός Πίνακας **4.4** με τα ποσοστά χρήσης.

Πίνακας 4.4: Ποσοστό χρήσης αιθουσών κατά μέσο όρο

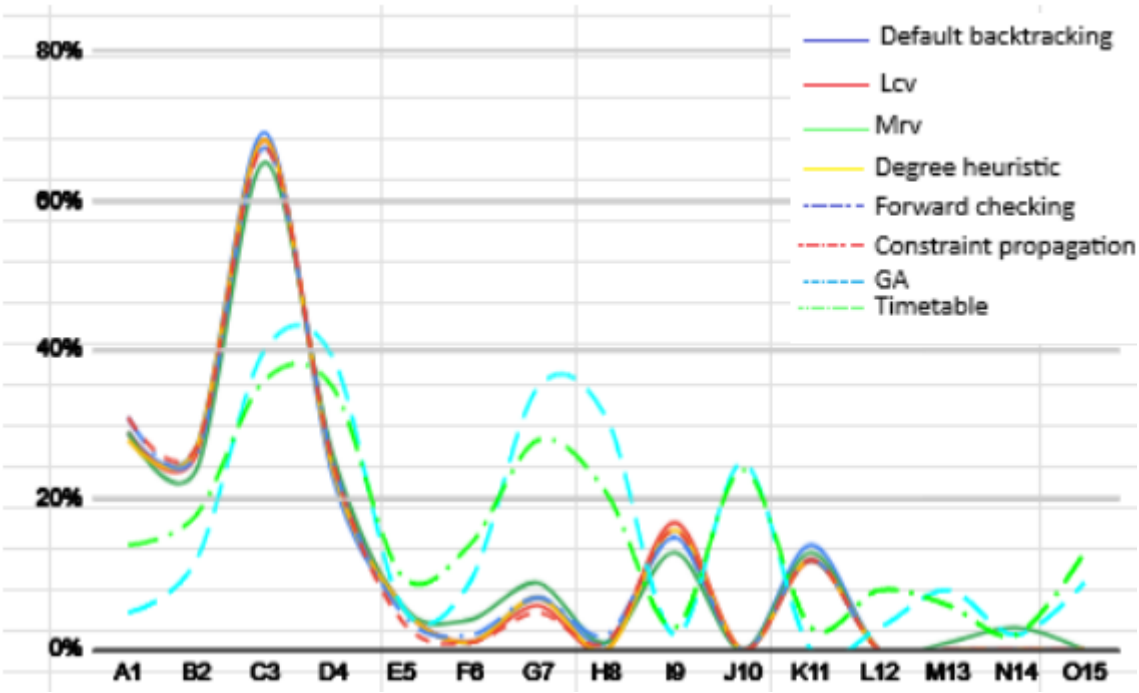
Utilization rate for each algorithm								
Α/Α παραδείγματος	Default backtr.	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	26.00%	25.88%	25.63%	26.13%	25.38%	26.00%	24.25%	25.38%
2	25.71%	25.71%	25.86%	25.71%	26.00%	25.57%	24.29%	24.57%
3	25.50%	25.60%	25.20%	25.40%	25.60%	25.50%	23.20%	23.70%
4	16.67%	16.67%	17.00%	16.50%	16.67%	16.83%	16.00%	16.50%
5	15.38%	15.38%	15.38%	15.38%	15.50%	15.50%	15.00%	24.00%
6	22.25%	22.00%	22.38%	22.38%	22.50%	22.25%	21.38%	21.88%
7	30.67%	30.67%	30.33%	30.67%	30.83%	30.67%	29.83%	30.50%
8	24.22%	24.56%	24.67%	24.56%	24.44%	24.44%	22.89%	23.33%
9	24.40%	24.60%	24.40%	24.00%	25.00%	24.80%	23.80%	23.80%
10	38.25%	38.25%	37.50%	38.50%	38.00%	38.50%	36.00%	38.00%
11	26.57%	26.71%	26.43%	26.71%	26.57%	27.43%	25.43%	25.86%
12	16.25%	16.33%	16.08%	16.83%	16.25%	16.25%	14.83%	15.00%
13	17.78%	17.89%	17.78%	17.89%	17.67%	17.78%	16.33%	16.56%
14	14.22%	15.00%	14.33%	14.33%	14.33%	14.22%	13.22%	13.22%
15	25.00%	24.80%	25.20%	24.80%	25.00%	25.00%	24.40%	24.40%
16	31.80%	31.40%	31.20%	31.20%	31.60%	31.40%	30.20%	31.60%
17	28.67%	28.67%	28.67%	28.67%	28.50%	28.50%	27.67%	28.50%
18	15.00%	15.08%	15.00%	15.15%	15.00%	15.08%	13.69%	14.31%
19	13.93%	14.00%	13.71%	14.07%	14.00%	13.86%	13.00%	13.21%
20	7.60%	7.60%	7.60%	7.60%	7.60%	7.60%	6.20%	6.47%
Μ.Ο.	22.29%	22.34%	22.22%	22.30%	22.32%	22.36%	21.08%	22.04%

Παρατηρείται ότι όλοι οι αλγόριθμοι έχουν περίπου τον ίδιο μέσο όρο χρήσης στις αίθουσες γεγονός που δυσκολεύει τη επιλογή κάποιου ως τον ιδανικό που θα λύσει το πρόβλημα. Αν θα μπορούσαμε όμως να ξεχωρίσουμε κάτι το οποίο δε φαίνεται με την πρώτη ματιά είναι ότι οι αλγόριθμοι που ανήκουν στους CSP (Constraint Satisfaction Problem) γεμίζουν τις αίθουσες με τη σειρά σε μεγάλο βαθμό που ση-

μαίνει ότι σε περίπτωση που υπάρχουν πολλές αίθουσες, κάποια ή κάποιες μπορεί να μην χρησιμοποιηθούν. Αντίθετα όμως στους γενετικούς αλγορίθμους παρατηρούμε ότι τα μαθήματα απλώνονται σε όλες τις αίθουσες με πιο τυχαίο τρόπο με αποτέλεσμα να χρησιμοποιούνται σχεδόν σε όλες τις περιπτώσεις όλες οι αίθουσες. Αυτός ειναι κι ο λόγος που κατά μέσο όρο τα ποσοστά χρήσης βγαίνουν παρόμοια διότι στη μία περίπτωση υπάρχουν μεγάλα ποσοστά χρήσης στις αρχικές αίθουσες και μικρότερα έως μηδενικά στις τελευταίες ενώ στη άλλη περίπτωση είναι πιο ομοιόμορφα κατανεμημένα.

Για τον λόγο αυτό στο Σχήμα 4.1 γίνεται η γραφική απεικόνηση κατανομής αιθουσών του κάθε αλγορίθμου όπου φαίνεται η διαφορά των γενετικών και αλγορίθμων ικανοποίησης περιορισμών.

Σχήμα 4.1: Ποσοστό χρήσης αιθουσών



Οι 2 γενετικοί αλγόριθμοι GA και Timetable φαίνεται ότι δεν έχουν πολύ υψηλά ποσοστά χρήσης αιθουσών συγκριτικά με τους αλγορίθμους περιορισμών, βέβαια αυτό δε σημαίνει ότι είναι καλύτεροι διότι όσο πιο μεγάλα ποσοστά χρήσης τόσο τόσο πιο αποδοτικές οι αίθουσες. Άρα καταλήγουμε στο συμπέρασμα ότι οι αλγόριθμοι περιορισμών προσφέρουν καλύτερες λύσεις σε σχέση με τους γενετικούς.

4.3 Γραφήματα για τον μέσο όρο της εκάστοτε μετρικής

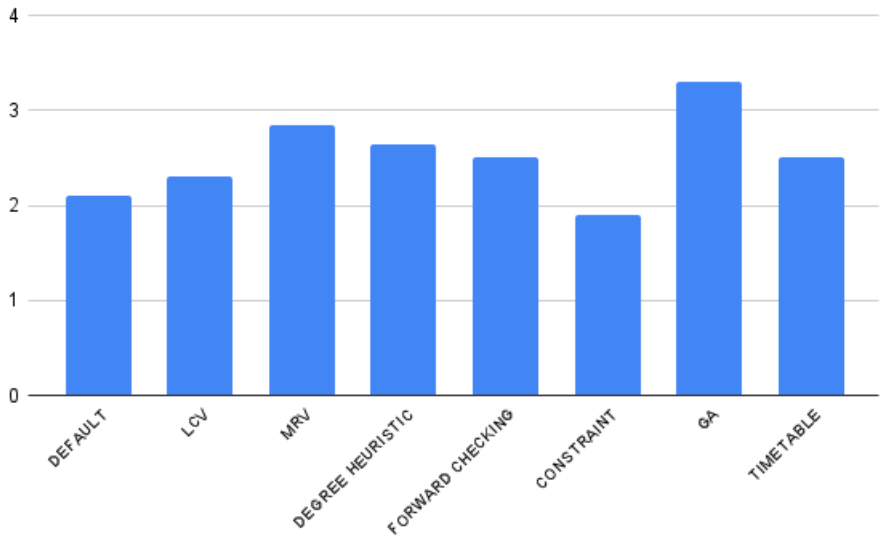
Σε αυτή την υποενότητα θα γίνει υπολογισμός και απεικόνηση σε γράφημα για τον μέσο όρο των ποινών κάθε αλγορίθμου από όλα τα παραδείγματα και σχολιασμός ώστε να αξιολογηθεί ποιος αλγόριθμος λύνει το πρόβλημα του χρονοπρογραματισμού με τον καλύτερό δυνατό τρόπο έχοντας λίγοτερα σφάλματα. Έτσι με τον υπολογισμό του μέσου όρου δημιουργείται ο συνολικός Πίνακας **4.5** για κάθε αλγόριθμο στην πρώτη μετρική.

Πίνακας 4.5: Μέσος όρος για κάθε αλγόριθμο-1 Μετρική

Μέσος όρος για κάθε αλγόριθμο-1 Μετρική	
Algorithms	Average Penalties
Default backtracking	2.1
Lcv	2.3
Mrv	2.85
Degree heuristic	2.65
Forward checking	2.5
Constraint Propagation	1.9
GA	3.3
Timetable	2.5

Στο Σχήμα 4.2 παρουσιάζεται ένα γράφημα στηλών με τον μέσο όρο σφαλμάτων κάθε αλγορίθμου, όπου στον άξονα x είναι οι αλγόριθμοι και στον y βρίσκονται ο μέσος όρος των penalties.

Σχήμα 4.2: Μέσος όρος για τη πρώτη μετρική



Μεταξύ των αλγορίθμων που μελετήθηκαν, ο constraint propagation ξεχώρισε περισσότερο για τη μείωση των ποινών γεγονός που τον καθιστά καλύτερο και ικανό να κατασκευάσει ένα πιο αποδοτικό εβδομαδιαίο πρόγραμμα με τα μαθήματα για κάθε εξάμηνο να είναι μοιρασμένα ομοιόμορφα.

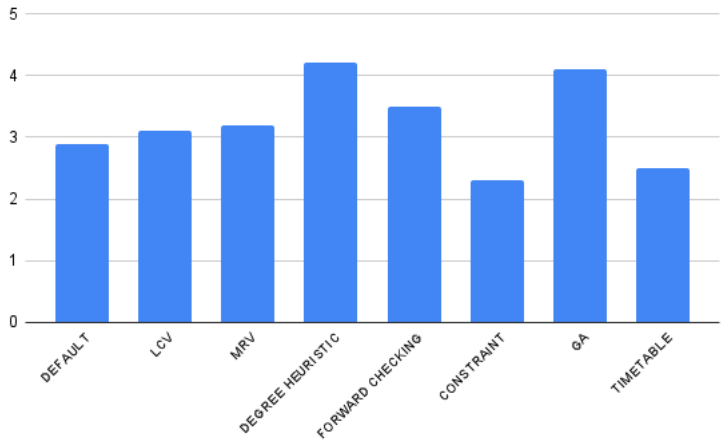
Στη συνέχεια προχωράμε στον υπολογισμό του μέσου όρου για την επόμενη μετρική (Ποινή όταν οι διαλέξεις ενός μαθήματος κάποιου έτους ξεπερνούν τη μία μέσα στη ίδια μέρα) και τη γραφική απεικόνησή του. Έτσι με τον υπολογισμό του μέσου όρου δημιουργείται ο συνολικός Πίνακας **4.6** για κάθε αλγόριθμο στη δεύτερη μετρική.

Πίνακας 4.6: Μέσος όρος για κάθε αλγόριθμο-2 Μετρική

Μέσος όρος για κάθε αλγόριθμο-2 Μετρική	
Algorithms	Average Penalties
Default backtracking	2.9
Lcv	3.1
Mrv	3.2
Degree heuristic	4.2
Forward checking	3.5
Constraint Propagation	2.3
GA	4.1
Timetable	2.5

Στο Σχήμα 4.3 παρουσιάζεται ένα γράφημα στηλών με τον μέσο όρο σφαλμάτων κάθε αλγορίθμου, όπου στον άξονα x είναι οι αλγόριθμοι και στον y βρίσκονται ο μέσος όρος των penalties.

Σχήμα 4.3: Μέσος όρος για τη δεύτερη μετρική



Παρομοίως με το γράφημα της προηγούμενης μετρικής ο αλγόριθμος με τις λιγό-

τερες ποινές είναι ο constraint propagation. Άρα αυτό μας οδηγεί στο συμπέρασμα ότι ο constraint propagation ικανοποιεί μέχρι στιγμής όλα τα constraints που έχουν εφαρμοστεί με τον καλύτερο δυνατό τρόπο.

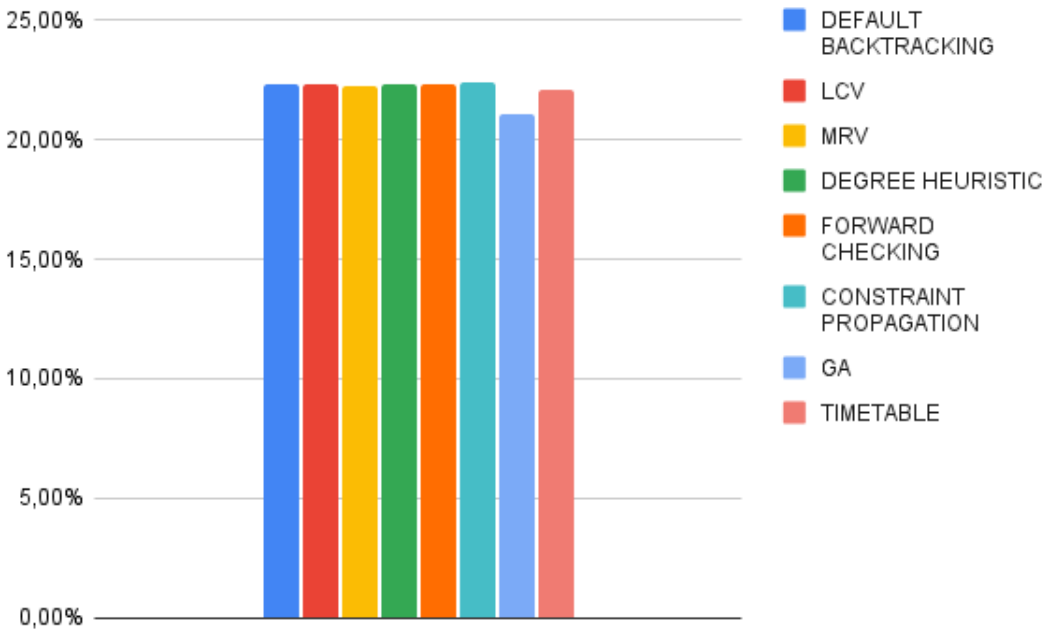
Με την ολοκλήρωση των προηγούμενων γραφημάτων γίνεται υπολογισμός του μέσου όρου για το ποσοστό χρήσης των αιθουσών για κάθε αλγόριθμο. Έτσι δημιουργείται ο συνολικός Πίνακας **4.7** με τα ποσοστά χρήσης για κάθε αλγόριθμο.

Πίνακας 4.7: Μέσος όρος ποσοστών χρήσης για κάθε αλγόριθμο-3 Μετρική

Μέσος όρος ποσοστών χρήσης για κάθε αλγόριθμο-3 Μετρική	
Algorithms	Average Penalties
Default backtracking	22.29%
Lcv	22.34%
Mrv	22.22%
Degree heuristic	22.30%
Forward checking	22.32%
Constraint Propagation	22.36%
GA	21.08%
Timetable	22.04%

Στο Σχήμα 4.4 παρουσιάζεται ένα γράφημα στηλών με τον μέσο όρο του utilization rate κάθε αλγορίθμου όπου στον άξονα x είναι οι αλγόριθμοι και στον y βρίσκονται τα ποσοστά χρήσης.

Σχήμα 4.4: Μέσος όρος για τη τρίτη μετρική

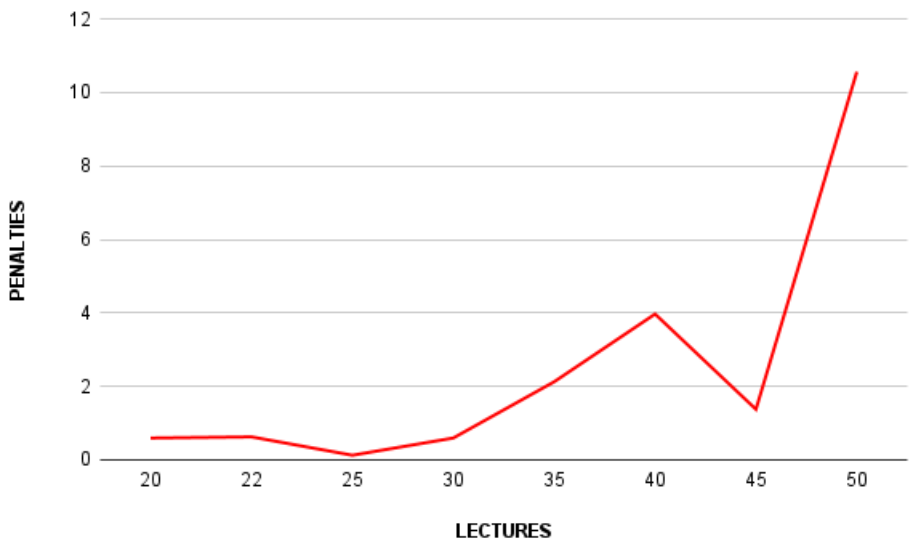


Αυτό που γίνεται αντιληπτό είναι ότι ο γενετικοί αλγόριθμοι έχουν ελαφρώς μικρότερο ποσοστό και συγκεκριμένα ο GA έχει το μικρότερο ποσοστό χρήσης κατά μέσο όρο, το οποίο μεταφράζεται ότι τα μαθήματα είναι κατανεμημένα σε όλες τις αίθουσες με ελαφρώς καλύτερο τρόπο, ενώ στους αλγορίθμους CSP (Constraint propagation problem) τοποθετούνται με τη σειρά στις αίθουσες με αποτέλεσμα ορισμένες να χρησιμοποιούνται απο ελάχιστα έως καθόλου. Είναι ένα λογικό αποτέλεσμα αν αναλογιστούμε ότι οι γενετικοί αλγόριθμοι προσπαθούν να βρουν τη καλύτερη δυνατή λύση σε αντίθεση με τους CSP που τοποθετούν τιμές στην πρώτη διαθέσιμη κενή θέση.

4.3.1 Μελέτη των penalties με βάση τις διαλέξεις

Σε αυτή την υποενότητα γίνεται μελέτη των ποινών για τις πρώτες δύο μετρικές με βάση τις διαλέξεις που χρησιμοποιήθηκαν σε όλα τα παραδείγματα, για να διαπιστωθεί αν τα αποτελέσματα που προκύπτουν είναι ανάλογα του πλήθους των διαλέξεων. Στην περίπτωση που υπήρχαν παραδείγματα με ίδιες διαλέξεις χρησιμοποιήθηκε ο μέσος όρος αυτών για τον υπολογισμό. Στο Σχήμα 4.5 παρουσιάζεται ένα γράφημα όπου στον άξονα x είναι οι διαλέξεις και στον y βρίσκονται ο μέσος όρος των penalties για τη μετρική: Ποινή για τη διδασκαλία τριών ή περισσότερων μαθήματων είτε είναι εργαστήρια είτε διαλέξεις απο το ίδιο έτος την ίδια μέρα.

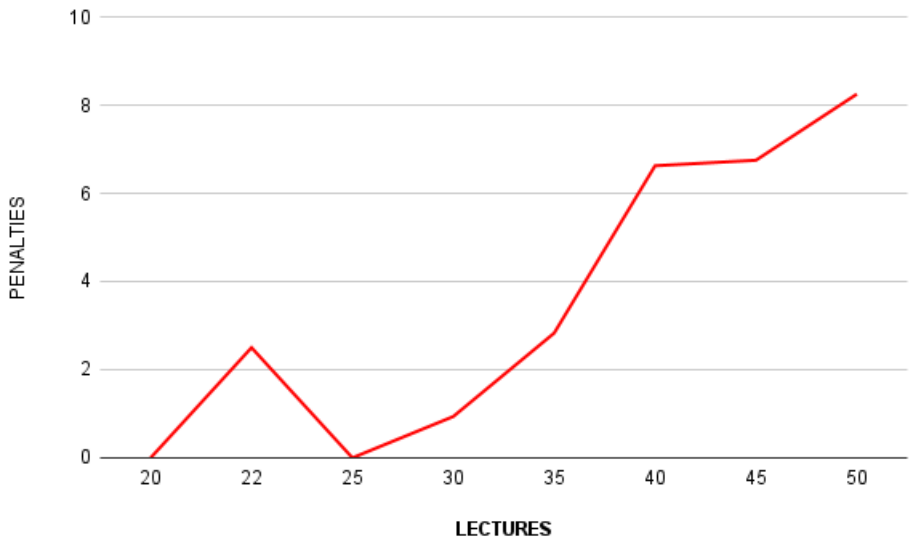
Σχήμα 4.5: Μέσος όρος με βάση τις διαλέξεις 1-Μετρική



Παρατηρώντας το γράφημα γίνεται αντιληπτό ότι υπάρχει μία απότομη πτώση των ποινών απο τις 40 διαλέξεις στις 45 και αυτό είναι λογικό καθώς στο συγκεκριμένο παράδειγμα αρκετά μαθήματα και συνεπώς διαλέξεις ανήκουν σε επιλογής μαθήματα τα οποία δεν προσμετρώνται στη καταγραφή σφαλμάτων. Παρ'όλα αυτά καταλήγουμε στο ασφαλές συμπέρασμα ότι οι ποινές σε όλους τους αλγορίθμους αυξάνονται ανάλογα με τον αριθμό διαλέξεων που εφαρμόζονται κάθε φορά.

Μετά το υπολογισμό και τη γραφική απεικόνηση για την παραπάνω μετρική, εφαρμόζουμε την ίδια λογική για τη μελέτη της δεύτερης μετρικής (Ποινή όταν οι διαλέξεις ενός μαθήματος κάποιου έτους ξεπερνούν τη μία μέσα στη ίδια μέρα) στο ίδιο ζήτημα. Στο Σχήμα 4.6 παρουσιάζεται ένα γράφημα όπου στον άξονα x είναι οι διαλέξεις και στον y βρίσκονται ο μέσος όρος των penalties για τη μετρική.

Σχήμα 4.6: Μέσος όρος με βάση τις διαλέξεις 2-Μετρική



Όπως και στο προηγούμενο γράφημα παρατηρείται μία πτώση στις 25 διαλέξεις βέβαια αυτη τη φορά και ο λόγος βρίσκεται στο γεγονός ότι στο συγκεκριμένο παράδειγμα δε διδάχθηκαν αρκτετές διαλέξεις στο κάθε μάθημα με αποτέλεσμα να μην παρουσιάζονται σφάλματα. Ένα δεύτερο κοινό που έχουν τα γραφήματα των δύο μετρικών είναι η εκθετική αύξηση των ποινών σε συνάρτηση με τη αύξηση των διαλέξεων,αυτό δείχνει όλοι αλγόριθμοι λειτουργούν με αρκετά παρόμοιο τρόπο στη δημιουργία του ωρολογίου προγράμματος.

4.4 Υπολογισμός τυπικής απόκλισης και δημιουργία κανονικής κατανομής

Σε αυτή την υποενότητα θα ασχοληθούμε με τον υπολογισμό τυπικής απόκλισης (Standard deviation) και στη συνέχεια τη δημιουργία κανονικής κατανομής με την υπόθεση ότι όλες οι τιμές ανήκουν σε μια κανονική κατανομή. Στη στατιστική, η τυπική απόκλιση (SD, εκπροσωπούμενη επίσης από το ελληνικό γράμμα σίγμα σ) είναι ένα μέτρο που χρησιμοποιείται για να υπολογιστεί το ποσό της μεταβολής ή της διασποράς ενός συνόλου τιμών δεδομένων. Μια χαμηλή τυπική απόκλιση υποδηλώνει ότι τα σημεία των δεδομένων τείνουν να είναι κοντά στο μέσο όρο (που ονομάζεται επίσης η αναμενόμενη τιμή) του συνόλου, ενώ μία υψηλή τυπική απόκλιση υποδεικνύει ότι τα στοιχεία απλώνονται πάνω από ένα ευρύτερο φάσμα των τιμών. Ο τύπος της τυπικής απόκλισης είναι ο εξής:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{4.5}$$

όπου σ τυπική απόκλιση, N ο αριθμός των penalties για κάθε αλγόριθμο από όλα τα παραδείγματα, xi το κάθε penalty απο τα παραδείγματα και $\overline{x}$ ο μέσος όρος των penalties.

Με βάση τα παραπάνω και με τη βοήθεια του πίνακα **4.5** που περιέχει το μέσο όρο penalties για όλους τους αλγορίθμους θα γίνει υπολογισμός της τυπικής απόκλισης για τη πρώτη μετρική και δημιουργείτε ο συνολικός Πίνακας **4.8** με τη τυπική απόκλιση για κάθε αλγόριθμο. Με την ολοκλήρωση του υπολογισμού της τυπικής

Πίνακας 4.8: Τυπική απόκλιση για κάθε αλγόριθμο

Τυπική απόκλιση για κάθε αλγόριθμο	
Algorithms	Standard deviation
Default backtracking	3.552612086
Lcv	2.921787485
Mrv	5.724140938
Degree heuristic	5.081183037
Forward checking	4.696022953
Constraint Propagation	4.140938109
GA	5.639148872
Timetable	4.817730411

απόκλισης περνάμε στη δημιουργία μίας κανονικής κατανομής (normal distribution). Η κανονική κατανομή, που συχνά αναφέρεται ως κατανομή Gauss, είναι μια θεμελιώδης έννοια στη στατιστική και τη θεωρία πιθανοτήτων. Περιγράφει την κατανομή μιας συνεχούς τυχαίας μεταβλητής όπου τα δεδομένα τείνουν να συγκεντρώνονται γύρω από τον μέσο όρο σε μια συμμετρική καμπύλη σε σχήμα καμπάνας. Πολλά φυσικά και κοινωνικά φαινόμενα παρουσιάζουν κανονική κατανομή, όπως τα ύψη των ατόμων σε έναν πληθυσμό, οι βαθμολογίες σε τυποποιημένα τεστ και τα σφάλματα στις μετρήσεις. Η κανονική κατανομή είναι απαραίτητη σε διάφορους τομείς, συμπεριλαμβανομένων των στατιστικών, των οικονομικών, της μηχανικής και των κοινωνικών επιστημών, καθώς παρέχει τη βάση για στατιστικές συμπερασμάτων, δοκιμές υποθέσεων και μοντελοποίηση φαινομένων του πραγματικού κόσμου.

Στο δικό μας ζήτημα του χρονοπρογραμματισμού ωρολογίου προγράμματος με τη υπόθεση ότι τα δεδομένα μας και όλες οι τιμές ανήκουν σε μια κανονική κατανομή και με τη βοήθεια του μέσου όρου και της τυπικής απόκλισης θα πραγματοποιηθεί σύγκριση μεταξύ των αλγορίθμων. Η παρακάτω μαθηματική σχέση χρησιμοποιείται για τη δημιουργία κανονικής κατανομής:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{4.6}$$

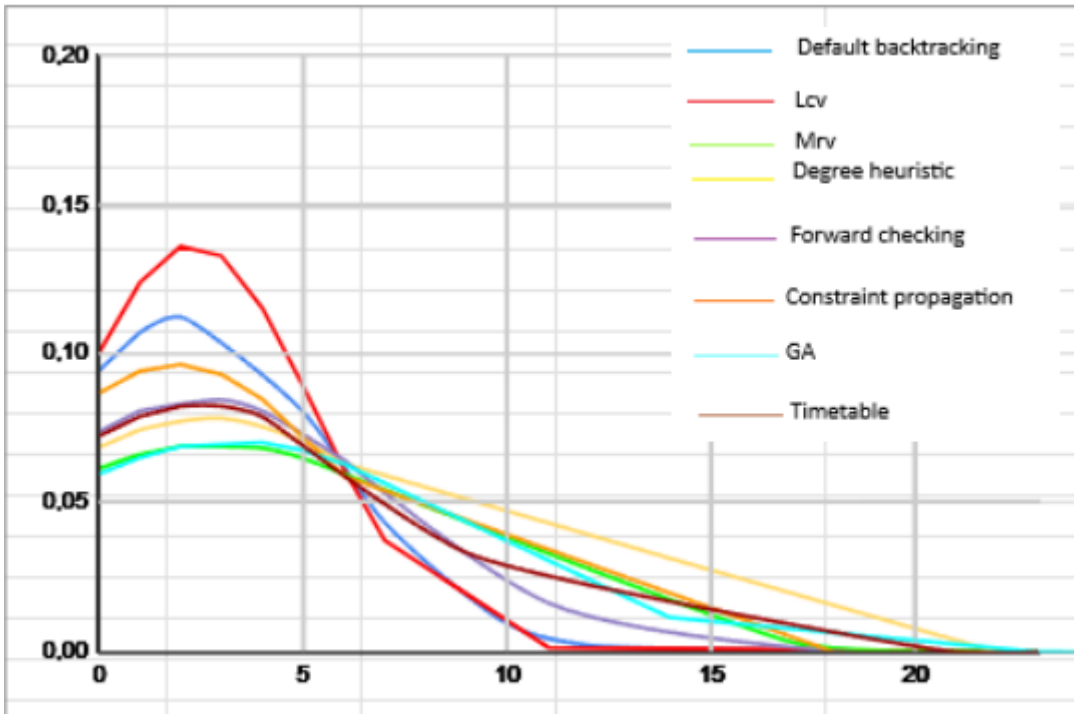
, όπου σ η τυπική απόκλιση, μ ο μέσος όρος και x τα penalties κάθε αλγορίθμου. Με αυτόν το τρόπο δημιουργείτε ο συνολικός Πίνακας **4.9** με τις τιμές της κανονικής κατανομής για κάθε αλγόριθμο.

Τέλος θα σχεδιαστεί το γράφημα κανονικής κατανομής για τη πρώτη μετρική (Ποινή για τη διδασκαλία τριών ή περισσότερων μαθήματων είτε είναι εργαστήρια είτε διαλέξεις από το ίδιο έτος την ίδια μέρα) για να αποτυπωθεί με μεγαλύτερη σαφήνεια ποιός αλγόριθμος είναι καλύτερος. Στο Σχήμα 4.7 ο άξονας x δείχνει τις τιμές των penalties και στο άξονα y τις τιμές για την κανονική κατανομή.

Πίνακας 4.9: Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου

Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου								
Examples	Default backtracking	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	0.009	0.061	0.003	0.063	0.016	0.084	0.011	0.082
2	0.043	0.037	0.068	0.063	0.064	0.093	0.063	0.033
3	0.002	0.001	0.000	0.000	0.000	0.000	0.000	0.000
4	0.107	0.123	0.066	0.068	0.073	0.086	0.065	0.078
5	0.094	0.123	0.061	0.068	0.073	0.086	0.065	0.072
6	0.107	0.123	0.061	0.068	0.073	0.096	0.068	0.072
7	0.094	0.132	0.066	0.070	0.073	0.086	0.065	0.078
8	0.107	0.089	0.068	0.063	0.080	0.094	0.067	0.082
9	0.094	0.100	0.061	0.068	0.073	0.086	0.059	0.072
10	0.112	0.100	0.064	0.068	0.080	0.086	0.065	0.078
11	0.094	0.123	0.066	0.074	0.073	0.086	0.068	0.078
12	0.107	0.123	0.066	0.074	0.080	0.096	0.068	0.078
13	0.094	0.100	0.061	0.074	0.073	0.086	0.059	0.072
14	0.107	0.123	0.061	0.068	0.073	0.086	0.059	0.072
15	0.094	0.135	0.061	0.068	0.073	0.086	0.065	0.078
16	0.094	0.100	0.061	0.068	0.073	0.086	0.065	0.082
17	0.094	0.123	0.066	0.074	0.073	0.086	0.065	0.078
18	0.080	0.123	0.068	0.078	0.084	0.096	0.070	0.082
19	0.107	0.115	0.066	0.074	0.064	0.059	0.065	0.078
20	0.094	0.100	0.061	0.068	0.073	0.086	0.059	0.072

Σχήμα 4.7: Κανονική κατανομή-1 μετρική



Όπως είναι φανερό απο σχήμα η καμπύλη που αντιστοιχεί στον Lcv αλγόριθμο (κόκκινη) είναι η μεγαλύτερη το οποίο σημαίνει ότι στη συγκεκριμένη μετρική ο Lcv δίνει πιο κάλα αποτελέσματα συγκρικά με τους υπόλοιπους. Αυτό έχει και εξήγηση όμως διότι ο μέσος όρος στο συγκεκριμένο αλγόριθμο είναι 2.3 όπως φαίνεται και στο πίνακα **4.5** και η τυπική απόκλιση είναι 2.9 σύμφωνα με το πίνακα **4.8**, έτσι επειδή ξέρουμε ότι όσο πιο μικρή είναι η απόκλιση αυτών των δύο μεταξύ τους τόσο κάλυτερά τα αποτελέσματα, καταλήγουμε ότι ο Lcv σε αυτή τη μετρική φαίνεται να είναι ο πιο αξιόπιστος.

Με τον ίδιο τρόπο προχωράμε στον υπολογισμό της τυπικής απόκλισης για τη δεύτερη μετρική (Ποινή όταν οι διαλέξεις ενός μαθήματος κάποιου έτους ξεπερνούν τη μία μέσα στη ίδια μέρα) και δημιουργείται ο Πίνακας **4.10** με την τυπική απόκλιση για κάθε αλγόριθμο.

Στη συνέχεια πραγματοποιείται ο υπολογισμός των τιμών για τη δημιουργία κανονικής κατανομής και φτιάχνεται ο συγκετρωτικός Πίνακας **4.11** με τη βοήθεια του μέσου όρου και τυπικής απόκλισης.

Πίνακας 4.10: Τυπική απόκλιση για κάθε αλγόριθμο-2 μετρική

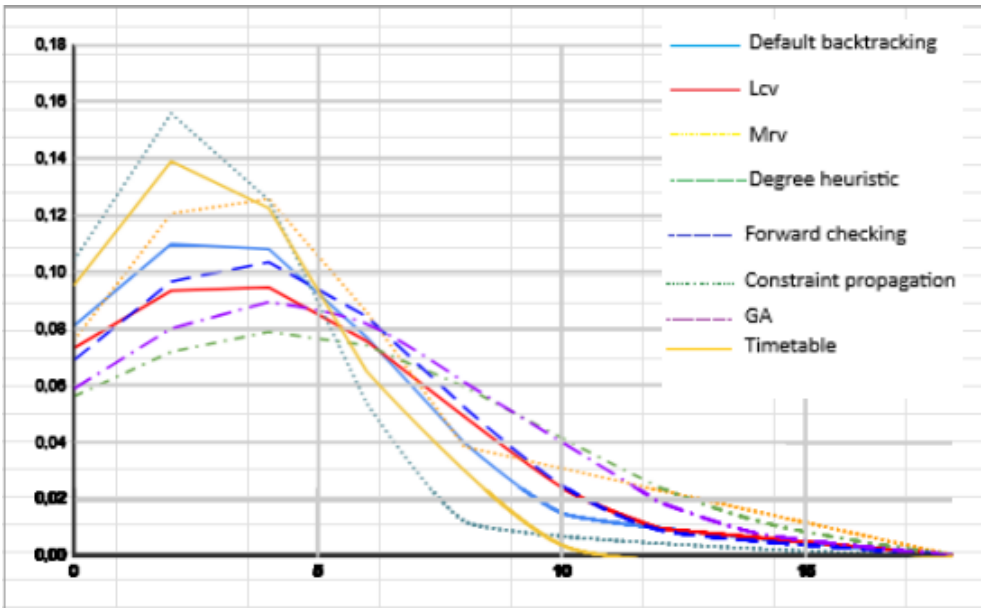
Τυπική απόκλιση για κάθε αλγόριθμο-2 μετρική	
Algorithms	Standard deviation
Default backtracking	3.522857692
Lcv	4.128208481
Mrv	3.071087584
Degree heuristic	5.063803438
Forward checking	3.831998242
Constraint Propagation	2.536055786
GA	4.470958922
Timetable	2.819107735

Πίνακας 4.11: Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου-1 μετρική

Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου-1 μετρική								
Examples	Default backtracking	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	0.039	0.009	0.038	0.059	0.052	0.125	0.079	0.065
2	0.014	0.009	0.038	0.071	0.084	0.054	0.089	0.122
3	0.080	0.094	0.085	0.052	0.008	0.012	0.007	0.004
4	0.107	0.072	0.120	0.073	0.103	0.104	0.079	0.139
5	0.076	0.075	0.038	0.073	0.052	0.054	0.018	0.065
6	0.039	0.094	0.085	0.059	0.103	0.125	0.060	0.122
7	0.076	0.023	0.125	0.024	0.024	0.156	0.060	0.139
8	0.039	0.075	0.085	0.012	0.084	0.125	0.018	0.065
9	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095
10	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095
11	0.109	0.072	0.075	0.055	0.068	0.104	0.079	0.095
12	0.080	0.072	0.120	0.071	0.096	0.156	0.079	0.139
13	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095
14	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095
15	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095
16	0.109	0.093	0.120	0.078	0.096	0.104	0.079	0.095
17	0.080	0.072	0.085	0.055	0.068	0.125	0.081	0.139
18	0.080	0.094	0.120	0.071	0.096	0.156	0.089	0.139
19	0.107	0.093	0.125	0.078	0.084	0.125	0.089	0.122
20	0.080	0.072	0.075	0.055	0.068	0.104	0.058	0.095

Με την ολοκλήρωση όλων των απαραίτητων υπολογισμών δημιουργείται το σχήμα της κανονικής κατανομής για τη δεύτερη μετρική 4.8 ο άξονας x δείχνει τις τιμές των penalties και στο άξονα y τις τιμές για τη κανονική κατανομή. Σε αυτή τη περί-

Σχήμα 4.8: Κανονική κατανομή-2 μετρικη



πτωση παρατηρείται ότι ο Constraint propagation έχει τη μεγαλύτερη καμπύλη και αυτό δικαιολογείται διότι όπως και στη προηγούμενη μετρική βλέποντας τις τιμές του μέσου όρου **4.6** και της τυπικής απόκλισης **4.10** οι οποίες είναι μ = 2.3 και σ = 2.53 αντίστοιχα.Έτσι καταλήγουμε ότι ο Constraint propagation δίνει καλύτερα αποτελέσματα στο ζήτημα μας σε αυτή τη μετρική. Όμως δε γίνεται να παραβλέψουμε πως και ο Lcv που έδωσε καλύτερα αποτελέσματα στη προηγούμενη μετρική είναι στους αλγορίθμους που ξεχωρίζουν δίνοντας αρκετά ικανοποιήτικές λύσεις.

Τέλος περνάμε στη τελευταία μετρική (Ποσόστά χρήσης αιθουσών) όπου ακολουθώντας την ίδια ακριβώς διαδικασία θα υπολογιστεί αρχικά η τυπική απόκλιση κάθε αλγορίθμου και θα φτιαχτεί ο συγκεντρωτικός Πίνακας **4.12** με τα αποτελέσματα.

Με το τέλος της εύρεσης τυπικής απόκλισης για κάθε αλγόριθμο συνεχίζουμε στον υπολογισμό των τιμών για τη κατασκευή κανονικής κατανομής. Στο πίνακα **4.13** αναγράφονται όλες οι απαραίτητες τιμές.

Με τη δημιουργία και του πίνακα με τις τίμες για την κανονική κατανομή θα δημιουργηθεί το Σχήμα 4.9, στον άξονα x τα ποσοστά χρήσης των αιθουσών και στον άξονα y οι τιμές της κανονικής κατανομής.

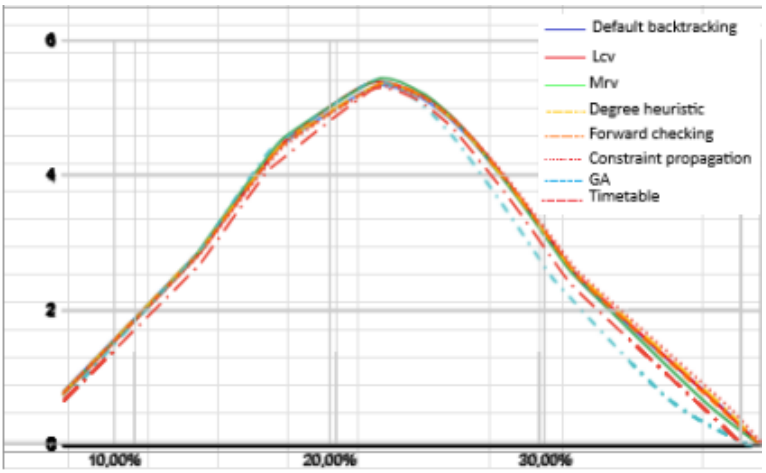
Πίνακας 4.12: Τυπική απόκλιση για κάθε αλγόριθμο-3 μετρική

Τυπική απόκλιση για κάθε αλγόριθμο-3 μετρική	
Algorithms	Standard deviation
Default backtracking	7.49%
Lcv	7.41%
Mrv	7.34%
Degree heuristic	7.46%
Forward checking	7.45%
Constraint Propagation	7.51%
GA	7.32%
Timetable	7.53%

Πίνακας 4.13: Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου

Κανονική κατανομή για κάθε τιμή penalty ενός αλγορίθμου								
Examples	Default backtracking	Lcv	Mrv	Degree heuristic	Forward checking	Constraint Propagation	GA	Timetable
1	4.712	4.801	4.876	4.687	4.924	4.724	4.960	4.802
2	4.800	4.852	4.803	4.817	4.742	4.849	4.949	5.008
3	4.860	4.885	5.002	4.905	4.862	4.869	5.224	5.172
4	4.018	4.016	4.220	3.953	4.016	4.051	4.282	4.042
5	3.478	3.463	3.521	3.478	3.521	3.500	3.859	5.122
6	5.326	5.375	5.431	5.348	5.356	5.313	5.443	5.298
7	2.849	2.862	2.951	2.849	2.789	2.879	2.668	2.818
8	5.153	5.144	5.137	5.108	5.145	5.114	5.284	5.222
9	5.120	5.136	5.197	5.211	5.022	5.040	5.084	5.156
10	0.550	0.538	0.623	0.505	0.583	0.526	0.683	0.559
11	4.525	4.522	4.608	4.490	4.553	4.230	4.566	4.659
12	3.846	3.874	3.831	3.883	3.842	3.816	3.784	3.422
13	4.442	4.493	4.525	4.491	4.407	4.412	4.414	4.066
14	2.979	3.296	3.051	3.022	3.011	2.952	3.062	2.667
15	4.989	5.092	5.002	5.056	5.022	4.995	4.916	5.045
16	2.380	2.550	2.571	2.624	2.465	2.573	2.508	2.365
17	3.707	3.737	3.692	3.713	3.797	3.803	3.634	3.666
18	3.315	3.331	3.351	3.378	3.303	3.321	3.273	3.128
19	2.855	2.858	2.776	2.910	2.869	2.799	2.963	2.663
20	0.777	0.745	0.749	0.767	0.758	0.769	0.691	0.624

Σχήμα 4.9: Κανονική κατανομή-3 μετρική



Σε αυτό το γράφημα δεν γίνεται αντιλητό αν κάποιος αλγόριθμος έχει καλύτερο ποσοστό χρήσης άπο άλλον, διότι όπως φαίνεται κι από τους πίνακες μέσου όρου **4.7** και τυπικής απόκλισης **4.12** οι διαφορές είναι σχεδόν μηδαμινές μεταξύ τους. Υπάρχει όμως εξήγηση διότι σε όλες τις δοκιμές δημιουργίας χρονοδιαγραμμάτων τα μαθήματα τοποθετούνταν με παρόμοιο τρόπο στους CSP αλγόριθμους γεμίζοντας τις αίθουσες με τη σειρά με αποτέλεσμα κάποιες αίθουσες να μην χρησιμοποιούνται και να υπάρχουν υψηλά ποσοστά χρήσης μεταξύ των υπολοίπων. Αυτή είναι και η κύρια διαφορά με τους γενετικούς αλγόριθμους καθώς εκεί τα ποσοστά χρήσης ήταν κατανεμημένα ομοιόμορφα στη πλειονότητα των αιθουσών. Τέλος το γεγονός ότι δε ξεχωρίζει σε αυτήν την περίπτωση κάποιος αλγόριθμος δεν είναι κακό αποτέλεσμα διότι η διαφορά μεταξύ τυπικής απόκλισης και μέσου όρου είναι μικρή κι έτσι μπορεί να βγει σαν συμπέρασμα ότι όλοι οι αλγόριθμοι του πειράματος κατανέμουν τα μαθήματα σε αίθουσες με σχεδόν ίδιο αλλά και αποτελεσματικό ταυτόχρονα τρόπο.

Κεφάλαιο 5

Συμπεράσματα

5.1 Αποτελέσματα

Ο στόχος αυτής της διπλωματικής εργασίας ήταν να ερευνηθούν και να συγκριθούν αποτελεσματικά αλγόριθμοι προγραμματισμού για τα χρονοδιαγράμματα των τάξεων που βελτιστοποιούν την κατανομή των πόρων, ελαχιστοποιώντας τα σφάλματα και ενισχύοντας τη συνολική απόδοση. Οι αλγόριθμοι δοκιμάστηκαν χρησιμοποιώντας δεδομένα πραγματικού κόσμου, τα οποία περιλάμβαναν πολλαπλά τμήματα, διαφορετικά μαθήματα, πολλαπλούς καθηγητές και διάφορους περιορισμούς. Τα αποτελέσματα της υλοποίησης συνοψίζονται ως εξής:

Βελτιστοποίηση κατανομής πόρων: Οι αλγόριθμοι διέθεσαν επιτυχώς αίθουσες διδασκαλίας, εκπαιδευτές και χρονοθυρίδες σε όλες τις τάξεις με υψηλό βαθμό αποτελεσματικότητας. Κατά μέσο όρο, ο αλγόριθμος GA είχε καλύτερο ποσοστό χρήσης συγκριτικά με τους υπόλοιπους με 21.08% γεγονός που τον καθιστά τον πιο αξιόπιστο.

Ελαχιστοποίηση σύγκρουσης: Ο πρωταρχικός στόχος ήταν να διασφαλιστεί ότι κανένας μαθητής ή εκπαιδευτής δεν είχε προγραμματιστεί για δύο μαθήματα ταυτόχρονα, κανένα μάθημα ίδιου έτους δεν προγραμματίστηκε για την ίδια χρονοθυρίδα και οι η προσέλευση φοιτητών στα μαθήματα δε ξεπερνούσε την εκάστοτε χωρητικότητα αιθουσών. Οι αλγόριθμοι πέτυχαν στο να ικανοποιηθούν όλοι οι σκληροί περιορισμοί στο 100% και και παράλληλα να δημιουργηθεί ένα αξιόπιστο χρονοδιάγραμμα.

Ικανότητα προσαρμογής: Οι αλγόριθμοι αποδείχθηκαν προσαρμόσιμοι σε αλλαγές στα δεδομένα εισόδου, όπως προσθήκες μαθημάτων ή αλλαγές στη διαθεσι-

μότητα των εκπαιδευτών, αποδεικνύοντας την ευρωστία και την ευελιξία τους σε δυναμικά περιβάλλοντα.

Με βάση τα αποτελέσματα και τις συγκρίσεις μεταξύ όλων των αλγορίθμων αυτός που ξεχώρισε στις περισσότερες των περιπτώσεων είναι ο constraint propagation καθώς και στους ελαστικούς περιορισμούς (soft constraints) και στους σκληρούς (hard constraints) αλλά και μέσα απο διαγράμματα κανονικής κατανομής, έδινε σε όλες τις περιπτώσεις ελαφρώς καλύτερα αποτελέσματα απο τους υπόλοιπους.

5.2 Μελλοντική χρήση

Τα πολλά υποσχόμενα αποτελέσματα υποδεικνύουν σημαντικές δυνατότητες για την ευρύτερη εφαρμογή των αλγορίθμων προγραμματισμού. Τα μελλοντικά σενάρια χρήσης και οι βελτιώσεις αναφέρονται αναλυτικά παρακάτω:

Επεκτασιμότητα: Οι αλγόριθμοι μπορεί να κλιμακωθούν για να φιλοξενήσουν μεγαλύτερα εκπαιδευτικά ιδρύματα με πιο σύνθετες ανάγκες προγραμματισμού. Η μελλοντική έρευνα μπορεί να επικεντρωθεί στη βελτιστοποίηση των αλγορίθμων για το χειρισμό ακόμη μεγαλύτερων συνόλων δεδομένων και πιο περίπλοκων περιορισμών, διασφαλίζοντας την εφαρμογή τους σε διαφορετικά εκπαιδευτικά περιβάλλοντα. Επίσης η ενσωμάτωση των αλγορίθμων με τα υπάρχοντα συστήματα διαχείρισης πανεπιστημίου μπορεί να εξορθολογίσει τη διαδικασία προγραμματισμού.

Ενισχυμένος χειρισμός περιορισμών: Ενώ οι αλγόριθμοι απέδωσαν καλά με τους δεδομένους περιορισμούς, η μελλοντική εργασία μπορεί να επικεντρωθεί στη βελτίωση της ικανότητάς τους να χειρίζονται ένα ευρύτερο φάσμα περιορισμών, συμπεριλαμβανομένων πιο περίπλοκων και διαφοροποιημένων απαιτήσεων. Αυτό μπορεί να περιλαμβάνει την ενσωμάτωση τεχνικών μηχανικής μάθησης για την πρόβλεψη και την προληπτική αντιμετώπιση πιθανών συγκρούσεων.

Βρόχος σχολίων και συνεχής βελτίωση: Η δημιουργία ενός βρόχου ανάδρασης όπου οι χρήστες μπορούν να αναφέρουν προβλήματα και να προτείνουν βελτιώσεις μπορεί να βοηθήσει στη συνεχή βελτίωση των αλγορίθμων. Η ενσωμάτωση των σχολίων των χρηστών σε περιοδικές ενημερώσεις θα διασφαλίσει ότι το σύστημα εξελίσσεται ώστε να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες και προτιμήσεις.

Εφαρμογή πέρα από την ακαδημαϊκή κοινότητα:

Οι αρχές και οι μεθοδολογίες των αλγόριθμων προγραμματισμού μπορούν να προσαρμοστούν για χρήση σε άλλα πεδία που απαιτούν αποτελεσματική κατανομή πόρων και προγραμματισμό χωρίς σφάλματα. Οι πιθανές εφαρμογές περιλαμβάνουν εταιρικά προγράμματα κατάρτισης, προγραμματισμό συνεδρίων και συστήματα ραντεβού για την υγειονομική περίθαλψη, μεταξύ άλλων.

Τέλος οι αλγόριθμοι που μελετήθηκαν αντιπροσωπεύουν μια σημαντική πρόοδο στον τομέα της δημιουργίας χρονοδιαγράμματος. Τα θετικά αποτελέσματα και οι πιθανές μελλοντικές εφαρμογές υπογραμμίζουν την αξία αυτής της έρευνας. Καθώς τα εκπαιδευτικά ιδρύματα και άλλοι οργανισμοί συνεχίζουν να αναζητούν αποδοτικές και αποτελεσματικές λύσεις προγραμματισμού, οι αλγόριθμοι αυτοί παρέχουν μια ισχυρή βάση για την κάλυψη αυτών των αναγκών, ανοίγοντας το δρόμο για συνεχή καινοτομία και βελτίωση στις μεθοδολογίες προγραμματισμού.

Βιβλιογραφία

[1] Alan M.Turing. Computing machinery and intelligence. *Mind*, LIX:433–460, 1950.

[2] Nils Aall Barricelli. Esempi numerici di processi di evoluzione. *Methodos*, page 45–68, 1954.

[3] Nils Aall Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos*, page 143–182, 1957.

[4] Alex Fraser. Simulation of genetic systems by automatic digital computers. i. introduction. *Aust. J. Biol. Sci.*, 10:484–491, 1957.

[5] Donald Burnell Alex Fraser (scientist). *Computer Models in Genetics*. McGraw-Hill.

[6] Jack L Crosby. *Computer simulation in genetics*. 1973.

[7] David B. Fogel. *Evolutionary Computation: The Fossil Record*. IEEE Press.

[8] Nils Aall Barricelli. Numerical testing of evolution theories. part ii. preliminary tests of performance, symbiogenesis and terrestrial life. *Acta Biotheoretica*, 16:99–126, 1963.

[9] Ingo Rechenberg. *Evolutions strategie*. Holzmann-Froboog.

[10] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen (PhD thesis)*. 1974.

[11] Hans-Paul Schwefel. *Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie : mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*. Birkhäuser, 1977.

[12] Hans-Paul Schwefel. *Numerical optimization of computer models (Translation of 1977 Numerische Optimierung von Computor-Modellen mittels der Evolutions strategie)*. Wiley, 1981.

[13] Sehraneh Ghaemi, M. Vakili, and Ali Aghagolzadeh. Using a genetic algorithm optimizer tool to solve university timetable scheduling problem. *2007 9th International Symposium on Signal Processing and Its Applications*, pages 1–4, 2007.

[14] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[15] Fred Glover. Tabu search – part 1. *ORSA Journal on Computing*, 1:190–206, 1989.

[16] Fred Glover. Tabu search – part 2. *ORSA Journal on Computing*, 2:4–32, 1990.

[17] Premlata A. Sonawane and Leena Ragha. Solving the class timetable problem by using genetic algorithm and tabu search algorithm. 2014.

[18] Jean-Louis Laurière. A language and a program for stating and solving combinatorial problems. *Artificial Intelligence*, 10:29–127, 1978.

[19] Juan José Blanco and Lina Khatib. Course scheduling as a constraint satisfaction problem. 1998.

[20] James Kennedy and Russell Eberhart. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4:1942–1948 vol.4, 2002.

[21] Yuhui Shi and Russell C. Eberhart. A modified particle swarm optimizer. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, 1998.

[22] Kennedy James. *Swarm Intelligence*, pages 187–219. Springer US, Boston, MA, 2006.

[23] Riccardo Poli. An analysis of publications on particle swarm optimisation applications. 2007. https://api.semanticscholar.org/CorpusID:9112239.

[24] Mohammad Reza Bonyadi and Zbigniew Michalewicz. Particle swarm optimization for single objective continuous space problems: A review. *Evolutionary Computation*, 25:1–54, 2017.

[25] Ruey-Maw Chen and Hsiao-Fang Shih. Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, 6:227–244, 2013.

[26] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.

[27] Arnaldo Vieira Moura and Rafael Augusto Scaraficci. A grasp strategy for a more constrained school timetabling problem. *International Journal of Operational Research*, 7:152–170, 2010.

[28] Ελένη Λεβέντη. Μελέτη και κατασκευή πληροφοριακού συστήματος για την εξαγωγή προγράμματος διδασκαλίας του τμήματος μηχανικών πληροφορικής και τηλεπικοινωνιών του Πανεπιστήμιου Δυτικής Μακεδονίας. Master's thesis, Πανεπιστήμιο Δυτικής Μακεδονίας, Κοζάνη, Ελλάδα, Ιούνιος 2012.

[29] https://cgi.di.uoa.gr/~ys02/dialekseis2020/csp1spp.pdf.

[30] https://www.cs.cornell.edu/courses/cs4700/2011fa/lectures/05_CSP.pdf.

Παραρτήματα

Παράρτημα Α΄

Αποτελέσματα αλγορίθμων

Εδώ θα γίνει παρουσίαση ενός παραδείγματος απεικόνησης χρονοδιαγράμματος για κάθε αλγόριθμο.

Α΄.1 Μαθήματα:25 Διαλέξεις:30 Εργαστήρια:30 Καθηγητές:8 Αίθουσες:12

Σχήμα Α΄.1: 12ο Παράδειγμα Default Backtracking

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for default backtracking: 95
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	4th year,Database 2 (Epilogis) Lily Johns,Practice 2,C3		3rd year,Web Development 3 Andrew Lively,Practice 2,D4 4th year,HTML 2 (Epilogis) James Web,Lecture,G7 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 2,C3	4th year,Mathematics 3 Lily Johns,Lecture,C3 5th year,SQL 2 John Drake,Practice 2,I9	2nd year,Algorithms 2 James Web,Lecture,C3 4th year,Matlab 2 (Epilogis) Katherine Balvin,Practice 2,B2
11:00 - 13:00	4th year,Database 2 (Epilogis) Lily Johns,Lecture,D4 5th year,Javascript 2 (Epilogis) Stamatia Papa,Lecture,C3	3rd year,SQL Lily Johns,Practice 1,I9 4th year,Matlab 2 (Epilogis) Katherine Balvin,Lecture,B2 5th year,JAVA James Web,Practice 2,A1	2nd year,Cloud Computing Stamatia Papa,Lecture,D4 3rd year,Mathematics 2 Katherine Balvin,Lecture,C3	5th year,SQL 2 John Drake,Practice 1,I9	2nd year,Python James Web,Practice 2,B2
13:00 - 15:00	2nd year,Cloud Computing Stamatia Papa,Lecture,C3 3rd year,Algorithms 3 John Drake,Lecture,D4	1st year,Application Engineering Stamatia Papa,Lecture,A1 5th year,Cloud Computing 3 (Epilogis) James Web,Lecture,K11	1st year,Application Engineering Stamatia Papa,Lecture,A1 2nd year,Python James Web,Practice 1,B2 3rd year,Web Development 3 Andrew Lively,Practice 1,C3	1st year,Matlab Steve Day,Lecture,B2 2nd year,HTML Lily Johns,Practice 2,C3	1st year,Algorithms Stamatia Papa,Lecture,C3 5th year,C++ Lily Johns,Practice 1,D4
15:00 - 17:00	4th year,Database 2 (Epilogis) Lily Johns,Practice 1,C3	1st year,Business Intelligence John Drake,Lecture,D4 3rd year,Web Development 3 Andrew Lively,Lecture,G7 4th year,HTML 2 (Epilogis) James Web,Practice 2,C3	4th year,Database 3 (Epilogis) James Web,Lecture,C3 5th year,C++ Lily Johns,Practice 2,D4	1st year,Business Intelligence John Drake,Practice 2,D4 2nd year,Python James Web,Lecture,B2 3rd year,Mathematics 2 Katherine Balvin,Lecture,C3 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 1,G7	1st year,Matlab Steve Day,Practice 2,B2 4th year,Javascript Stamatia Papa,Lecture,K11 5th year,JAVA James Web,Practice 1,A1
17:00 - 19:00	2nd year,Algorithms 2 James Web,Practice 1,C3	3rd year,SQL Lily Johns,Practice 2,K11 4th year,Matlab 2 (Epilogis) Katherine Balvin,Practice 1,B2 5th year,SQL 2 John Drake,Lecture,I9	1st year,Business Intelligence John Drake,Practice 1,C3 4th year,Database 3 (Epilogis) James Web,Practice 1,D4	3rd year,SQL Lily Johns,Lecture,I9 4th year,Javascript Stamatia Papa,Practice 2,K11	4th year,Javascript Stamatia Papa,Practice 1,K11 5th year,C++ Lily Johns,Lecture,C3
19:00 - 21:00	1st year,Algorithms Stamatia Papa,Lecture,C3	3rd year,Algorithms 3 John Drake,Lecture,C3 5th year,JAVA James Web,Lecture,A1	4th year,Application Engineering 2 (Epilogis) Lily Johns,Lecture,C3	1st year,Matlab Steve Day,Practice 1,B2 2nd year,HTML Lily Johns,Lecture,C3	2nd year,HTML Lily Johns,Practice 1,D4 4th year,HTML 2 (Epilogis) James Web,Practice 1,G7 5th year,Application Engineering 3 (Epilogis) Andrew Lively,Lecture,C3

Σχήμα Α΄.2: 12ο Παράδειγμα LCV

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for backtracking with LCV_improved: 80
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	4th year,Matlab 2 (Epilogis) Stamatia Papa,Practice 2,B2	4th year,HTML 2 (Epilogis) Lily Johns,Practice 1,D4 5th year,C++ Eirini Best,Lecture,C3	4th year,Database 3 (Epilogis) Andrew Lively,Practice 1,C3 5th year,JAVA James Web,Lecture,A1	4th year,Javascript Lily Johns,Practice 2,K11 5th year,JAVA James Web,Practice 2,A1	2nd year,HTML John Drake,Practice 1,C3
11:00 - 13:00	1st year,Business Intelligence Steve Day,Practice 1,C3 2nd year,Cloud Computing Lily Johns,Lecture,D4	5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 1,C3	1st year,Application Engineering Stamatia Papa,Lecture,A1 3rd year,Web Development 3 Katherine Balvin,Practice 1,C3	4th year,Database 2 (Epilogis) Lily Johns,Practice 2,D4 5th year,C++ Eirini Best,Practice 2,C3	1st year,Business Intelligence Steve Day,Practice 2,C3 4th year,Javascript Lily Johns,Lecture,K11
13:00 - 15:00	4th year,Database 2 (Epilogis) Lily Johns,Practice 1,D4 5th year,Javascript 2 (Epilogis) Stamatia Papa,Lecture,C3	2nd year,HTML John Drake,Lecture,C3 5th year,Cloud Computing 3 (Epilogis) Lily Johns,Lecture,K11	1st year,Matlab Steve Day,Lecture,B2 3rd year,Mathematics 2 Lily Johns,Lecture,C3 5th year,SQL 2 John Drake,Lecture,I9	3rd year,Algorithms 3 James Web,Lecture,C3	4th year,HTML 2 (Epilogis) Lily Johns,Lecture,C3
15:00 - 17:00	1st year,Algorithms Steve Day,Lecture,C3 3rd year,Mathematics 2 Lily Johns,Lecture,D4 4th year,Matlab 2 (Epilogis) Stamatia Papa,Lecture,B2 5th year,SQL 2 John Drake,Practice 2,I9	1st year,Application Engineering Stamatia Papa,Lecture,A1 5th year,SQL 2 John Drake,Practice 1,I9	2nd year,Python Stamatia Papa,Practice 1,B2 3rd year,Web Development 3 Katherine Balvin,Practice 2,C3 4th year,HTML 2 (Epilogis) Lily Johns,Practice 2,D4	2nd year,HTML John Drake,Practice 2,D4 4th year,Javascript Lily Johns,Practice 1,K11 5th year,Application Engineering 3 (Epilogis) James Web,Lecture,C3	1st year,Matlab Steve Day,Practice 1,B2
17:00 - 19:00	2nd year,Algorithms 2 Lily Johns,Lecture,D4 3rd year,Algorithms 3 James Web,Lecture,C3	1st year,Business Intelligence Steve Day,Lecture,C3 3rd year,SQL Katherine Balvin,Practice 1,I9 4th year,Matlab 2 (Epilogis) Stamatia Papa,Practice 1,B2	2nd year,Algorithms 2 Lily Johns,Practice 1,G7 4th year,Mathematics 3 John Drake,Lecture,C3 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 2,D4	2nd year,Cloud Computing Lily Johns,Lecture,C3	3rd year,SQL Katherine Balvin,Practice 2,I9 4th year,Database 2 (Epilogis) Lily Johns,Lecture,C3 5th year,JAVA James Web,Practice 1,A1
19:00 - 21:00	2nd year,Python Stamatia Papa,Practice 2,B2		3rd year,Web Development 3 Katherine Balvin,Lecture,G7 4th year,Database 3 (Epilogis) Andrew Lively,Lecture,C3 5th year,C++ Eirini Best,Practice 1,D4	1st year,Matlab Steve Day,Practice 2,B2 3rd year,SQL Katherine Balvin,Lecture,I9 4th year,Application Engineering 2 (Epilogis) Lily Johns,Lecture,C3	1st year,Algorithms Steve Day,Lecture,C3 2nd year,Python Stamatia Papa,Lecture,B2

Σχήμα Α΄.3: 12ο Παράδειγμα MRV

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for backtracking with MRV: 75
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	2nd year,Python James Web,Lecture,B2	1st year,Algorithms James Web,Lecture,C3 4th year,Database 2 (Epilogis) Andrew Lively,Practice 2,D4	3rd year,Algorithms 3 Lily Johns,Lecture,C3 5th year,SQL 2 John Drake,Practice 1,I9	2nd year,HTML Lily Johns,Practice 2,C3 5th year,C++ Eirini Best,Practice 1,D4	1st year,Matlab Katherine Balvin,Practice 1,F6 2nd year,Python James Web,Practice 2,B2 5th year,Cloud Computing 3 (Epilogis) Lily Johns,Lecture,A1
11:00 - 13:00	3rd year,Web Development 3 John Drake,Practice 1,C3	3rd year,Mathematics 2 Katherine Balvin,Lecture,C3 4th year,Matlab 2 (Epilogis) John Drake,Practice 2,B2	1st year,Algorithms James Web,Lecture,C3 4th year,Matlab 2 (Epilogis) John Drake,Lecture,B2		5th year,JAVA Eirini Best,Practice 1,A1
13:00 - 15:00	2nd year,Algorithms 2 John Drake,Lecture,D4 3rd year,Algorithms 3 Lily Johns,Lecture,G7 4th year,Database 3 (Epilogis) James Web,Practice 1,C3	3rd year,Web Development 3 John Drake,Practice 2,C3 4th year,Javascript Lily Johns,Lecture,K11	2nd year,HTML Lily Johns,Practice 1,C3 4th year,Database 2 (Epilogis) Andrew Lively,Lecture,D4	3rd year,SQL Lily Johns,Lecture,I9 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 1,C3	1st year,Business Intelligence John Drake,Practice 2,G7 4th year,Application Engineering 2 (Epilogis) James Web,Lecture,C3 5th year,JAVA Eirini Best,Practice 2,A1
15:00 - 17:00	2nd year,Cloud Computing Lily Johns,Lecture,D4 4th year,HTML 2 (Epilogis) James Web,Lecture,C3 5th year,C++ Eirini Best,Lecture,G7	4th year,HTML 2 (Epilogis) James Web,Practice 1,C3 5th year,SQL 2 John Drake,Lecture,I9	2nd year,HTML Lily Johns,Lecture,C3	1st year,Application Engineering Stamatia Papa,Lecture,A1 4th year,HTML 2 (Epilogis) James Web,Practice 2,C3	1st year,Application Engineering Stamatia Papa,Lecture,A1 4th year,Javascript Lily Johns,Practice 1,K11
17:00 - 19:00	1st year,Matlab Katherine Balvin,Lecture,F6 2nd year,Python James Web,Practice 1,B2 4th year,Javascript Lily Johns,Practice 2,K11 5th year,JAVA Eirini Best,Lecture,A1	3rd year,Mathematics 2 Katherine Balvin,Lecture,C3 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 2,D4	5th year,SQL 2 John Drake,Practice 2,I9	2nd year,Cloud Computing Lily Johns,Lecture,C3 4th year,Database 2 (Epilogis) Andrew Lively,Practice 1,D4 5th year,C++ Eirini Best,Practice 2,G7	2nd year,Algorithms 2 John Drake,Practice 1,C3 3rd year,SQL Lily Johns,Practice 1,K11 5th year,Javascript 2 (Epilogis) Stamatia Papa,Lecture,D4
19:00 - 21:00	3rd year,SQL Lily Johns,Practice 2,K11 4th year,Matlab 2 (Epilogis) John Drake,Practice 1,B2	1st year,Business Intelligence John Drake,Lecture,D4 4th year,Database 3 (Epilogis) James Web,Lecture,C3 5th year,Application Engineering 3 (Epilogis) Stamatia Papa,Lecture,G7	1st year,Matlab Katherine Balvin,Practice 2,B2	1st year,Business Intelligence John Drake,Practice 1,C3	3rd year,Web Development 3 John Drake,Lecture,C3 4th year,Mathematics 3 Lily Johns,Lecture,D4

Σχήμα Α΄.4: 12ο Παράδειγμα Degree Heuristic

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for backtracking with Degree Heuristic: 96
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	2nd year,HTML Lily Johns,Practice 2,C3 3rd year,Algorithms 3 John Drake,Lecture,D4	1st year,Algorithms James Web,Lecture,C3 3rd year,Web Development 3 John Drake,Practice 1,D4	2nd year,HTML Lily Johns,Lecture,D4 3rd year,Web Development 3 John Drake,Practice 2,G7 4th year,Mathematics 3 Stamatia Papa,Lecture,C3 5th year,JAVA James Web,Practice 1,A1	4th year,Matlab 2 (Epilogis) John Drake,Lecture,B2 5th year,Javascript 2 (Epilogis) Stamatia Papa,Lecture,C3	1st year,Application Engineering Stamatia Papa,Lecture,A1 3rd year,Mathematics 2 Lily Johns,Lecture,C3
11:00 - 13:00	2nd year,Cloud Computing Stamatia Papa,Lecture,C3 3rd year,SQL Lily Johns,Lecture,I9 4th year,Matlab 2 (Epilogis) John Drake,Practice 1,B2	1st year,Business Intelligence John Drake,Practice 1,G7 2nd year,HTML Lily Johns,Practice 1,D4 5th year,C++ James Web,Practice 2,C3	2nd year,Python John Drake,Practice 1,B2 4th year,Database 2 (Epilogis) Stamatia Papa,Lecture,C3	1st year,Matlab Steve Day,Practice 2,B2 3rd year,Mathematics 2 Lily Johns,Lecture,C3 5th year,JAVA James Web,Lecture,A1	1st year,Business Intelligence John Drake,Practice 2,C3 3rd year,SQL Lily Johns,Practice 1,I9 5th year,JAVA James Web,Practice 2,A1
13:00 - 15:00	1st year,Matlab Steve Day,Lecture,B2 3rd year,Web Development 3 John Drake,Lecture,D4 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 2,C3	5th year,Application Engineering 3 (Epilogis) Andrew Lively,Lecture,C3	2nd year,Algorithms 2 Lily Johns,Practice 1,C3 5th year,SQL 2 Katherine Balvin,Practice 2,I9	1st year,Application Engineering Stamatia Papa,Lecture,A1 3rd year,SQL Lily Johns,Practice 2,I9 4th year,Database 3 (Epilogis) John Drake,Practice 1,D4 5th year,C++ James Web,Lecture,C3	2nd year,Python John Drake,Lecture,B2 4th year,Database 2 (Epilogis) Stamatia Papa,Practice 1,C3
15:00 - 17:00	4th year,Database 2 (Epilogis) Stamatia Papa,Practice 2,C3	3rd year,Algorithms 3 John Drake,Lecture,D4 4th year,HTML 2 (Epilogis) Lily Johns,Lecture,C3	5th year,C++ James Web,Practice 1,C3	1st year,Matlab Steve Day,Practice 1,B2 4th year,HTML 2 (Epilogis) Lily Johns,Practice 1,C3	4th year,HTML 2 (Epilogis) Lily Johns,Practice 2,C3
17:00 - 19:00	1st year,Business Intelligence John Drake,Lecture,C3	4th year,Javascript John Drake,Practice 1,K11 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 1,C3	4th year,Javascript John Drake,Practice 2,K11		2nd year,Python John Drake,Practice 2,B2
19:00 - 21:00	4th year,Database 3 (Epilogis) John Drake,Lecture,C3 5th year,SQL 2 Katherine Balvin,Lecture,I9	1st year,Algorithms James Web,Lecture,C3 4th year,Matlab 2 (Epilogis) John Drake,Practice 2,B2 5th year,Cloud Computing 3 (Epilogis) Lily Johns,Lecture,K11	2nd year,Algorithms 2 Lily Johns,Lecture,C3 4th year,Javascript John Drake,Lecture,K11	2nd year,Cloud Computing Stamatia Papa,Lecture,D4 4th year,Application Engineering 2 (Epilogis) Andrew Lively,Lecture,C3	5th year,SQL 2 Katherine Balvin,Practice 1,I9

Σχήμα Α΄.5: 12ο Παράδειγμα Forward Checking

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for backtracking with Forward Checking: 60
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	1st year,Matlab John Drake,Lecture,B2 5th year,C++ Lily Johns,Practice 2,C3	4th year,Database 2 (Epilogis) Lily Johns,Practice 2,C3	5th year,JAVA James Web,Lecture,A1	4th year,Database 2 (Epilogis) Lily Johns,Practice 1,C3	3rd year,Algorithms 3 John Drake,Lecture,C3
11:00 - 13:00	3rd year,Algorithms 3 John Drake,Lecture,C3 5th year,Javascript 2 (Epilogis) James Web,Practice 1,D4	2nd year,Python Stamatia Papa,Practice 1,B2 3rd year,Mathematics 2 Katherine Balvin,Lecture,C3 5th year,Javascript 2 (Epilogis) James Web,Lecture,D4	2nd year,HTML Lily Johns,Practice 1,C3 5th year,SQL 2 Katherine Balvin,Practice 2,I9	1st year,Application Engineering Katherine Balvin,Lecture,A1 5th year,JAVA James Web,Practice 2,E5	2nd year,Algorithms 2 Lily Johns,Practice 1,C3 4th year,Mathematics 3 John Drake,Lecture,D4 5th year,JAVA James Web,Practice 1,A1
13:00 - 15:00	2nd year,Python Stamatia Papa,Lecture,B2 4th year,Matlab 2 (Epilogis) Katherine Balvin,Lecture,F6 5th year,C++ Lily Johns,Lecture,C3	1st year,Matlab John Drake,Practice 1,B2	5th year,SQL 2 Katherine Balvin,Lecture,I9	1st year,Algorithms Steve Day,Lecture,C3 3rd year,Web Development 3 Katherine Balvin,Practice 1,D4 4th year,Javascript Stamatia Papa,Practice 1,K11 5th year,Javascript 2 (Epilogis) James Web,Practice 2,G7	2nd year,Cloud Computing Lily Johns,Lecture,C3 3rd year,Web Development 3 Katherine Balvin,Practice 2,D4 5th year,Cloud Computing 3 (Epilogis) Andrew Lively,Lecture,K11
15:00 - 17:00	4th year,Database 2 (Epilogis) Lily Johns,Lecture,C3	2nd year,HTML Lily Johns,Lecture,C3 4th year,Database 3 (Epilogis) Andrew Lively,Lecture,D4	4th year,Database 3 (Epilogis) Andrew Lively,Practice 1,C3 5th year,Application Engineering 3 (Epilogis) James Web,Lecture,D4	1st year,Business Intelligence Katherine Balvin,Practice 2,C3 2nd year,Python Stamatia Papa,Practice 2,B2	3rd year,SQL Lily Johns,Lecture,I9
17:00 - 19:00	1st year,Business Intelligence Katherine Balvin,Lecture,C3 4th year,HTML 2 (Epilogis) James Web,Practice 1,D4 5th year,C++ Lily Johns,Practice 1,G7	1st year,Matlab John Drake,Practice 2,B2 2nd year,Cloud Computing Lily Johns,Lecture,C3 3rd year,Mathematics 2 Katherine Balvin,Lecture,D4 4th year,HTML 2 (Epilogis) James Web,Lecture,G7	1st year,Application Engineering Katherine Balvin,Lecture,A1	4th year,Application Engineering 2 (Epilogis) Lily Johns,Lecture,C3	1st year,Algorithms Steve Day,Lecture,C3 5th year,SQL 2 Katherine Balvin,Practice 1,I9
19:00 - 21:00	1st year,Business Intelligence Katherine Balvin,Practice 1,C3 2nd year,Algorithms 2 Lily Johns,Lecture,D4 4th year,Javascript Stamatia Papa,Practice 2,K11	3rd year,SQL Lily Johns,Practice 1,I9 4th year,Matlab 2 (Epilogis) Katherine Balvin,Practice 2,B2	2nd year,HTML Lily Johns,Practice 2,C3 4th year,Matlab 2 (Epilogis) Katherine Balvin,Practice 1,B2	3rd year,SQL Lily Johns,Practice 2,I9 4th year,HTML 2 (Epilogis) James Web,Practice 2,C3	3rd year,Web Development 3 Katherine Balvin,Lecture,C3 4th year,Javascript Stamatia Papa,Lecture,K11

Σχήμα Α΄.6: 12ο Παράδειγμα Constraint Propagation

File - main

```
C:\Users\johngnwst\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\johngnwst\Desktop\διπλωματικη κωδικες\timetableCSP-master\main.py"
Counter for backtracking with Constraint Propagation: 60
```

Lesson Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	2nd year,Cloud Computing Lily Johns,Lecture,C3 4th year,Database 2 (Epilogis) Stamatia Papa,Lecture,D4	1st year,Business Intelligence John Drake,Practice 1,C3 4th year,Matlab 2 (Epilogis) Stamatia Papa,Practice 1,B2	1st year,Matlab Katherine Balvin,Practice 1,B2 4th year,Database 3 (Epilogis) Andrew Lively,Lecture,C3 5th year,C++ Lily Johns,Practice 2,D4	2nd year,Algorithms 2 Lily Johns,Practice 1,C3	2nd year,HTML Lily Johns,Lecture,C3 4th year,HTML 2 (Epilogis) James Web,Practice 2,D4 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 2,G7
11:00 - 13:00	2nd year,HTML Lily Johns,Practice 2,C3 4th year,Mathematics 3 John Drake,Lecture,D4	3rd year,SQL Katherine Balvin,Lecture,I9 4th year,Database 2 (Epilogis) Stamatia Papa,Practice 2,C3	2nd year,Python James Web,Practice 1,B2	3rd year,SQL Katherine Balvin,Practice 1,I9 4th year,HTML 2 (Epilogis) James Web,Lecture,C3	5th year,Javascript 2 (Epilogis) Stamatia Papa,Lecture,C3
13:00 - 15:00	1st year,Application Engineering Katherine Balvin,Lecture,A1 4th year,Javascript Lily Johns,Lecture,K11 5th year,Javascript 2 (Epilogis) Stamatia Papa,Practice 1,C3	1st year,Algorithms James Web,Lecture,C3 5th year,C++ Lily Johns,Practice 1,D4	2nd year,Cloud Computing Lily Johns,Lecture,C3	3rd year,Mathematics 2 Katherine Balvin,Lecture,C3 4th year,HTML 2 (Epilogis) James Web,Practice 1,D4 5th year,JAVA Lily Johns,Practice 2,A1	3rd year,Web Development 3 Andrew Lively,Practice 1,C3 4th year,Matlab 2 (Epilogis) Stamatia Papa,Practice 2,B2 5th year,SQL 2 Katherine Balvin,Practice 1,I9
15:00 - 17:00	3rd year,Algorithms 3 Lily Johns,Lecture,C3 5th year,Application Engineering 3 (Epilogis) James Web,Lecture,D4	1st year,Application Engineering Katherine Balvin,Lecture,A1 3rd year,Web Development 3 Andrew Lively,Lecture,C3	1st year,Business Intelligence John Drake,Practice 2,C3 2nd year,Algorithms 2 Lily Johns,Lecture,D4 5th year,SQL 2 Katherine Balvin,Lecture,I9	4th year,Application Engineering 2 (Epilogis) James Web,Lecture,C3 5th year,JAVA Lily Johns,Practice 1,A1	
17:00 - 19:00	1st year,Algorithms James Web,Lecture,C3 3rd year,Algorithms 3 Lily Johns,Lecture,D4 4th year,Matlab 2 (Epilogis) Stamatia Papa,Lecture,B2	4th year,Javascript Lily Johns,Practice 1,K11 5th year,SQL 2 Katherine Balvin,Practice 2,I9	3rd year,Web Development 3 Andrew Lively,Practice 2,C3 4th year,Javascript Lily Johns,Practice 2,K11	2nd year,Python James Web,Lecture,B2	1st year,Matlab Katherine Balvin,Lecture,B2 4th year,Database 2 (Epilogis) Stamatia Papa,Practice 1,C3
19:00 - 21:00	5th year,JAVA Lily Johns,Lecture,A1	1st year,Business Intelligence John Drake,Lecture,C3 2nd year,HTML Lily Johns,Practice 1,D4 3rd year,Mathematics 2 Katherine Balvin,Lecture,G7 4th year,Database 3 (Epilogis) Andrew Lively,Practice 1,H8 5th year,Cloud Computing 3 (Epilogis) James Web,Lecture,K11	1st year,Matlab Katherine Balvin,Practice 2,B2 2nd year,Python James Web,Practice 2,F6	5th year,C++ Lily Johns,Lecture,C3	3rd year,SQL Katherine Balvin,Practice 2,I9

Σχήμα Α΄.7: 12ο Παράδειγμα Timetable

File - main

```
Solution found in 279 generations
Final solution fitness: 1.0
Clashes: 100
---- Monday ----
+--------------------------------+----------+------+-----------------+----------------------+
|             Course             |  Group   | Room |    Professor    |       Timeslot       |
+--------------------------------+----------+------+-----------------+----------------------+
|           Algorithms           | 1st Year |  G7  |  Stamatia Papa  | Monday 11:00 - 13:00 |
|             Matlab             | 1st Year |  B2  |    Steve Day    | Monday 9:00 - 11:00  |
|        Cloud Computing         | 2nd Year |  C3  |  Stamatia Papa  | Monday 15:00 - 17:00 |
|              SQL               | 3rd Year |  H8  | Katherine Balvin | Monday 15:00 - 17:00 |
| Database 2 (Epilogis) Practice | 4th Year |  C3  |  Stamatia Papa  | Monday 9:00 - 11:00  |
|         JAVA Practice          | 5th Year |  E5  |   Eirini Best   | Monday 15:00 - 17:00 |
|         SQL 2 Practice         | 5th Year |  A1  | Katherine Balvin | Monday 11:00 - 13:00 |
+--------------------------------+----------+------+-----------------+----------------------+
**************************************************************
---- Tuesday ----
+--------------------------------+----------+------+-----------------+----------------------+
|             Course             |  Group   | Room |    Professor    |       Timeslot       |
+--------------------------------+----------+------+-----------------+----------------------+
|  Business Intelligence Practice  | 1st Year |  G7  |    Steve Day    | Tuesday 11:00 - 13:00 |
|     Application Engineering      | 1st Year |  E5  |    Steve Day    | Tuesday 15:00 - 17:00 |
|         Cloud Computing          | 2nd Year |  C3  |    Steve Day    | Tuesday 13:00 - 15:00 |
|              Python              | 2nd Year |  F6  |  Stamatia Papa  | Tuesday 11:00 - 13:00 |
|           SQL Practice           | 3rd Year |  I9  | Katherine Balvin | Tuesday 19:00 - 21:00 |
|        Web Development 3         | 3rd Year |  G7  |  Andrew Lively  | Tuesday 15:00 - 17:00 |
|    Web Development 3 Practice    | 3rd Year | J10  |  Andrew Lively  | Tuesday 17:00 - 19:00 |
|          Mathematics 2           | 3rd Year | J10  |    James Web    |  Tuesday 9:00 - 11:00 |
|          Mathematics 3           | 4th Year | J10  |    John Drake   | Tuesday 19:00 - 21:00 |
|            Javascript            | 4th Year |  F6  |    Lily Johns   | Tuesday 17:00 - 19:00 |
|   Matlab 2 (Epilogis) Practice   | 4th Year |  B2  |    John Drake   | Tuesday 13:00 - 15:00 |
|    HTML 2 (Epilogis) Practice    | 4th Year | J10  |    James Web    | Tuesday 15:00 - 17:00 |
|      Database 3 (Epilogis)       | 4th Year | J10  |  Andrew Lively  | Tuesday 11:00 - 13:00 |
| Javascript 2 (Epilogis) Practice | 5th Year |  C3  |    James Web    | Tuesday 19:00 - 21:00 |
|               JAVA               | 5th Year |  B2  |   Eirini Best   | Tuesday 9:00 - 11:00 |
|          JAVA Practice           | 5th Year |  E5  |   Eirini Best   | Tuesday 13:00 - 15:00 |
|           C++ Practice           | 5th Year |  D4  |    Lily Johns   | Tuesday 15:00 - 17:00 |
+--------------------------------+----------+------+-----------------+----------------------+
**************************************************************
---- Wednesday ----
+-----------------------------------+----------+------+--------------+----------------------+
|              Course               |  Group   | Room |   Professor  |       Timeslot       |
+-----------------------------------+----------+------+--------------+----------------------+
|             Algorithms            | 1st Year |  C3  |   Steve Day  | Wednesday 19:00 - 21:00 |
|           Algorithms 2            | 2nd Year |  C3  |   Lily Johns | Wednesday 17:00 - 19:00 |
|          Python Practice          | 2nd Year |  G7  |   John Drake |  Wednesday 9:00 - 11:00 |
|          HTML Practice            | 2nd Year |  D4  | Stamatia Papa | Wednesday 11:00 - 13:00 |
|         HTML 2 (Epilogis)         | 4th Year |  G7  |   James Web  | Wednesday 11:00 - 13:00 |
|       Database 2 (Epilogis)       | 4th Year | J10  |   Lily Johns | Wednesday 13:00 - 15:00 |
|   Database 2 (Epilogis) Practice  | 4th Year |  D4  | Stamatia Papa |  Wednesday 9:00 - 11:00 |
| Application Engineering 2 (Epilogis) | 4th Year | J10 | Andrew Lively | Wednesday 19:00 - 21:00 |
|          SQL 2 Practice           | 5th Year |  C3  |   John Drake | Wednesday 15:00 - 17:00 |
|    Cloud Computing 3 (Epilogis)   | 5th Year |  B2  | Andrew Lively | Wednesday 11:00 - 13:00 |
+-----------------------------------+----------+------+--------------+----------------------+
**************************************************************
---- Thursday ----
+--------------------------------+----------+------+-----------------+----------------------+
|             Course             |  Group   | Room |    Professor    |       Timeslot       |
+--------------------------------+----------+------+-----------------+----------------------+
|     Business Intelligence      | 1st Year |  G7  | Katherine Balvin | Thursday 19:00 - 21:00 |
|  Business Intelligence Practice  | 1st Year |  H8  |    Steve Day    | Thursday 15:00 - 17:00 |
|         Python Practice        | 2nd Year |  F6  |    John Drake   | Thursday 17:00 - 19:00 |
|     Algorithms 2 Practice      | 2nd Year |  C3  |    James Web    | Thursday 11:00 - 13:00 |
|         Mathematics 2          | 3rd Year | J10  | Katherine Balvin | Thursday 17:00 - 19:00 |
|          SQL Practice          | 3rd Year |  E5  |    John Drake   | Thursday 15:00 - 17:00 |
|      Javascript Practice       | 4th Year |  B2  |    Lily Johns   | Thursday 11:00 - 13:00 |
|  Database 3 (Epilogis) Practice  | 4th Year |  D4  |  Andrew Lively  | Thursday 15:00 - 17:00 |
|     Javascript 2 (Epilogis)    | 5th Year |  G7  |    James Web    | Thursday 9:00 - 11:00  |
| Javascript 2 (Epilogis) Practice | 5th Year | J10  |  Stamatia Papa  | Thursday 13:00 - 15:00 |
+--------------------------------+----------+------+-----------------+----------------------+
**************************************************************
---- Friday ----
+-----------------------------------+----------+------+-----------------+----------------------+
|              Course               |  Group   | Room |    Professor    |       Timeslot       |
+-----------------------------------+----------+------+-----------------+----------------------+
|          Matlab Practice          | 1st Year |  F6  | Katherine Balvin | Friday 11:00 - 13:00 |
|      Application Engineering      | 1st Year |  C3  |    Steve Day    | Friday 17:00 - 19:00 |
|          Matlab Practice          | 1st Year |  F6  |    Steve Day    | Friday 13:00 - 15:00 |
|               HTML                | 2nd Year | J10  |  Stamatia Papa  | Friday 17:00 - 19:00 |
|          HTML Practice            | 2nd Year |  G7  |    Lily Johns   | Friday 15:00 - 17:00 |
|          Algorithms 3             | 3rd Year |  D4  |    James Web    | Friday 17:00 - 19:00 |
|     Web Development 3 Practice    | 3rd Year |  D4  |    John Drake   | Friday 15:00 - 17:00 |
|          Algorithms 3             | 3rd Year | J10  |    James Web    | Friday 9:00 - 11:00  |
|    Matlab 2 (Epilogis) Practice   | 4th Year |  B2  |  Stamatia Papa  | Friday 9:00 - 11:00  |
|       Matlab 2 (Epilogis)         | 4th Year |  B2  | Katherine Balvin | Friday 19:00 - 21:00 |
|       Javascript Practice         | 4th Year | K11  |    John Drake   | Friday 13:00 - 15:00 |
|     HTML 2 (Epilogis) Practice    | 4th Year |  H8  |    Lily Johns   | Friday 11:00 - 13:00 |
|          C++ Practice             | 5th Year | J10  |   Eirini Best   | Friday 13:00 - 15:00 |
|              C++                  | 5th Year |  G7  |   Eirini Best   | Friday 9:00 - 11:00  |
|             SQL 2                 | 5th Year |  E5  |    James Web    | Friday 15:00 - 17:00 |
| Application Engineering 3 (Epilogis) | 5th Year |  G7  |  Stamatia Papa  | Friday 19:00 - 21:00 |
+-----------------------------------+----------+------+-----------------+----------------------+
**************************************************************
679.5911554999766
```

Σχήμα Α΄.8: 12ο Παράδειγμα GA

File - main

```
> Generation # 1311
+-----------------+-------------+-------------------+
| schedule number | health rate | numb of conflicts |
+-----------------+-------------+-------------------+
|        0        |     1.0     |         0         |
|        1        |     0.5     |         1         |
|        2        |     0.25    |         3         |
|        3        |    0.143    |         6         |
|        4        |    0.143    |         6         |
|        5        |    0.111    |         8         |
|        6        |    0.091    |         10        |
|        7        |    0.091    |         10        |
|        8        |    0.083    |         11        |
|        9        |    0.077    |         12        |
+-----------------+-------------+-------------------+
+---------+----------+-----------------------------------------+-----------------+------------------+-----------------------+
| Class # |   Spec   |    Course (number, max num of students)   | Room (Capacity) |     Teacher      |      Lesson Time)     |
+---------+----------+-----------------------------------------+-----------------+------------------+-----------------------+
|    0    | 1st Year |              Algorithms (30)              |     G7 (30)     |    John Drake    | 13:00 - 15:00THURSDAY |
|    1    | 1st Year |              Algorithms (30)              |     D4 (30)     |    John Drake    |  19:00 - 21:00FRIDAY  |
|    2    | 1st Year |        Application Engineering (25)       |     A1 (25)     |    Lily Johns    |  15:00 - 17:00MONDAY  |
|    3    | 1st Year |        Application Engineering (25)       |     F6 (26)     |    Lily Johns    | 09:00 - 11:00WEDNESDAY |
|    4    | 1st Year |         Business Intelligence (30)        |     J10 (30)    |  Stamatia Papa   | 11:00 - 13:00THURSDAY |
|    5    | 1st Year |               Matlab (26)                 |     C3 (30)     |    John Drake    |  17:00 - 19:00MONDAY  |
|    6    | 1st Year |    Business Intelligence Practice (30)    |     G7 (30)     | Katherine Balvin | 11:00 - 13:00WEDNESDAY |
|    7    | 1st Year |    Business Intelligence Practice (30)    |     J10 (30)    | Katherine Balvin |  17:00 - 19:00FRIDAY  |
|    8    | 1st Year |           Matlab Practice (26)            |     H8 (30)     |  Stamatia Papa   |  09:00 - 11:00TUESDAY |
|    9    | 1st Year |           Matlab Practice (26)            |     G7 (30)     |  Stamatia Papa   |  13:00 - 15:00FRIDAY  |
|    10   | 2nd Year |               Python (26)                 |     F6 (26)     |    James Web     |  17:00 - 19:00TUESDAY |
|    11   | 2nd Year |           Python Practice (26)            |     D4 (30)     |    John Drake    |  11:00 - 13:00FRIDAY  |
|    12   | 2nd Year |           Python Practice (26)            |     J10 (30)    |  Stamatia Papa   | 09:00 - 11:00WEDNESDAY |
|    13   | 2nd Year |            Algorithms 2 (30)              |     C3 (30)     |    Lily Johns    |  09:00 - 11:00FRIDAY  |
|    14   | 2nd Year |         Algorithms 2 Practice (30)        |     H8 (30)     |    Lily Johns    |  15:00 - 17:00TUESDAY |
|    15   | 2nd Year |           Cloud Computing (30)            |     C3 (30)     |    John Drake    | 13:00 - 15:00THURSDAY |
|    16   | 2nd Year |           Cloud Computing (30)            |     G7 (30)     |    James Web     |  17:00 - 19:00FRIDAY  |
|    17   | 2nd Year |            HTML Practice (30)             |     J10 (30)    |    James Web     |  19:00 - 21:00FRIDAY  |
|    18   | 2nd Year |            HTML Practice (30)             |     C3 (30)     |    James Web     | 19:00 - 21:00THURSDAY |
|    19   | 2nd Year |                HTML (30)                  |     H8 (30)     |    John Drake    |  11:00 - 13:00MONDAY  |
|    20   | 3rd Year |            Mathematics 2 (30)             |     G7 (30)     |  Andrew Lively   |  13:00 - 15:00TUESDAY |
|    21   | 3rd Year |            Mathematics 2 (30)             |     G7 (30)     |    Steve Day     |  13:00 - 15:00MONDAY  |
|    22   | 3rd Year |            Algorithms 3 (30)              |     H8 (30)     |    Steve Day     |  11:00 - 13:00FRIDAY  |
|    23   | 3rd Year |            Algorithms 3 (30)              |     C3 (30)     |    Steve Day     |  13:00 - 15:00FRIDAY  |
|    24   | 3rd Year |                 SQL (15)                  |     C3 (30)     |    John Drake    |  09:00 - 11:00TUESDAY |
|    25   | 3rd Year |            SQL Practice (15)              |     B2 (26)     |  Stamatia Papa   |  09:00 - 11:00MONDAY  |
|    26   | 3rd Year |            SQL Practice (15)              |     D4 (30)     |    John Drake    |  11:00 - 13:00MONDAY  |
|    27   | 3rd Year |          Web Development 3 (30)           |     C3 (30)     |  Andrew Lively   |  15:00 - 17:00TUESDAY |
|    28   | 3rd Year |      Web Development 3 Practice (30)      |     G7 (30)     |  Andrew Lively   |  15:00 - 17:00MONDAY  |
|    29   | 3rd Year |      Web Development 3 Practice (30)      |     H8 (30)     |  Andrew Lively   | 11:00 - 13:00THURSDAY |
|    30   | 4th Year |            Mathematics 3 (30)             |     G7 (30)     |  Andrew Lively   |  13:00 - 15:00FRIDAY  |
|    31   | 4th Year |             Javascript (20)               |     G7 (30)     |    Lily Johns    |  09:00 - 11:00MONDAY  |
|    32   | 4th Year |          Javascript Practice (20)         |     B2 (26)     |    John Drake    |  15:00 - 17:00TUESDAY |
|    33   | 4th Year |          Javascript Practice (20)         |     B2 (26)     |    John Drake    |  09:00 - 11:00TUESDAY |
|    34   | 4th Year |          Matlab 2 (Epilogis) (26)         |     F6 (26)     | Katherine Balvin |  11:00 - 13:00MONDAY  |
|    35   | 4th Year |     Matlab 2 (Epilogis) Practice (26)     |     G7 (30)     |    James Web     | 19:00 - 21:00THURSDAY |
|    36   | 4th Year |     Matlab 2 (Epilogis) Practice (26)     |     B2 (26)     |  Stamatia Papa   |  11:00 - 13:00TUESDAY |
|    37   | 4th Year |           HTML 2 (Epilogis) (30)          |     J10 (30)    | Katherine Balvin | 15:00 - 17:00WEDNESDAY |
|    38   | 4th Year |      HTML 2 (Epilogis) Practice (30)      |     D4 (30)     | Katherine Balvin |  15:00 - 17:00FRIDAY  |
|    39   | 4th Year |      HTML 2 (Epilogis) Practice (30)      |     C3 (30)     | Katherine Balvin |  19:00 - 21:00MONDAY  |
|    40   | 4th Year |         Database 2 (Epilogis) (30)        |     G7 (30)     |  Stamatia Papa   |  17:00 - 19:00FRIDAY  |
|    41   | 4th Year |    Database 2 (Epilogis) Practice (30)    |     H8 (30)     |  Stamatia Papa   |  17:00 - 19:00MONDAY  |
|    42   | 4th Year |    Database 2 (Epilogis) Practice (30)    |     H8 (30)     |  Stamatia Papa   | 13:00 - 15:00THURSDAY |
|    43   | 4th Year |         Database 3 (Epilogis) (30)        |     C3 (30)     |  Andrew Lively   |  17:00 - 19:00TUESDAY |
|    44   | 4th Year |    Database 3 (Epilogis) Practice (30)    |     D4 (30)     |    John Drake    | 09:00 - 11:00THURSDAY |
|    45   | 4th Year | Application Engineering 2 (Epilogis) (30) |     D4 (30)     |    Lily Johns    | 13:00 - 15:00WEDNESDAY |
|    46   | 5th Year |        Javascript 2 (Epilogis) (30)       |     C3 (30)     |    Lily Johns    | 11:00 - 13:00THURSDAY |
|    47   | 5th Year |   Javascript 2 (Epilogis) Practice (30)   |     C3 (30)     |    James Web     |  13:00 - 15:00FRIDAY  |
|    48   | 5th Year |   Javascript 2 (Epilogis) Practice (30)   |     C3 (30)     |  Stamatia Papa   |  15:00 - 17:00MONDAY  |
|    49   | 5th Year |                JAVA (22)                  |     F6 (26)     |    Lily Johns    |  19:00 - 21:00MONDAY  |
|    50   | 5th Year |            JAVA Practice (22)             |     E5 (25)     |    Lily Johns    | 11:00 - 13:00WEDNESDAY |
|    51   | 5th Year |            JAVA Practice (22)             |     C3 (30)     |    James Web     |  09:00 - 11:00FRIDAY  |
|    52   | 5th Year |                 C++ (30)                  |     G7 (30)     |    Lily Johns    | 13:00 - 15:00WEDNESDAY |
|    53   | 5th Year |            C++ Practice (30)              |     H8 (30)     |    Lily Johns    |  17:00 - 19:00TUESDAY |
|    54   | 5th Year |            C++ Practice (30)              |     G7 (30)     |  Stamatia Papa   | 09:00 - 11:00THURSDAY |
|    55   | 5th Year |                SQL 2 (15)                 |     I9 (15)     |    James Web     |  15:00 - 17:00TUESDAY |
|    56   | 5th Year |           SQL 2 Practice (15)             |     H8 (30)     | Katherine Balvin |  13:00 - 15:00MONDAY  |
|    57   | 5th Year |           SQL 2 Practice (15)             |     C3 (30)     | Katherine Balvin | 17:00 - 19:00WEDNESDAY |
|    58   | 5th Year |     Cloud Computing 3 (Epilogis) (20)     |     L12 (25)    |    James Web     | 13:00 - 15:00THURSDAY |
|    59   | 5th Year | Application Engineering 3 (Epilogis) (30) |     D4 (30)     |  Andrew Lively   |  15:00 - 17:00FRIDAY  |
+---------+----------+-----------------------------------------+-----------------+------------------+-----------------------+
```