ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ
ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σάμος ,2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

“Υλοποίηση αλγορίθμου προγραμματισμού ωρολογίου προγράμματος εξετάσεων”

Του

Καραβούλια Ευάγγελου

321/2011058

Επιβλέπων: Δρ.Αλέξιος Καπόρης

Μόνιμος Επίκουρος Καθηγητής , τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, Πανεπιστήμιο Αιγαίου

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του προπτυχιακού προγράμματος σπουδών του τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων της Πολυτεχνικής Σχολής του Πανεπιστημίου Αιγαίου.

Ευχαριστίες

Η διπλωματική αυτή διεξήχθη υπό την επίβλεψη του Δρ.Αλέξιου Καπόρη τον οποίο σέβομαι για το ότι μου έδωσε την ευκαιρία και στη συνέχεια παρείχε υποστήριξη και καθοδήγηση. Εκτιμώ σε μεγάλο βαθμό όλη την υπομονή του, κατανόηση συνεχή ενθάρρυνση και έμπνευση.

Επιπρόσθετα, ένα μεγάλο ευχαριστώ στην οικογένεια μου για την ηθική, ψυχολογική και οικονομική στήριξη που μου παρείχε καθ' όλη την διάρκεια των σπουδών μου.

Περίληψη

Το πρόβλημα του χρονοδιαγράμματος των γραπτών εξετάσεων των πανεπιστημίων είναι ένα πολύ γνωστό πολυδιάστατο πρόβλημα αντιστοίχησης περιορισμών που επικεντρώνεται στην ανάθεση μαθημάτων σε μέλη ΔΕΠ ,σε τάξεις εντός περιορισμένων χρονικών διαστημάτων. Ως εκ τούτου, είναι ένα δύσκολο χρονοβόρο πρόβλημα που αντιμετωπίζουν τα πανεπιστήμια. Η πολυπλοκότητα αυτή δικαιολογείται από το πλήθος καθηγητών και μαθημάτων και από τον μεγάλο αριθμό περιορισμών και κριτηρίων κατανομής. Μία χειρωνακτική λύση σε αυτό το πρόβλημα, λαμβάνοντας υπόψη όλους τους περιορισμούς, απαιτεί συνήθως μακρόχρονη και επίπονη δουλειά για την προσφορά εφικτής βέλτιστης λύσεως. Καταλαβαίνουμε οτι είναι επιτακτική, λοιπόν η ανάγκη για την ανάπτυξη ενός αυτοματοποιημένου πληροφοριακού συστήματος το οποίο θα είναι ικανό να επιλύσει το παραπάνω πρόβλημα. Η διπλωματική εργασία αυτή επικεντρώνεται στην ανάπτυξη ενός αλγορίθμου με στόχο την αυτόματη ικανοποίηση όλων των περιορισμών για το συγκεκριμένο πρόβλημα του πανεπιστημιακού χρονοδιαγράμματος των γραπτών εξετάσεων του Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων του Πανεπιστήμιου Αιγαίου. Πρέπει να δοθεί ιδιαίτερη έμφαση οτι ο αλγόριθμος καλύπτει τους περισσότερους περιορισμούς για τις ανάγκες του συγκεκριμένου πανεπιστημίου ενώ μπορεί να θεωρηθεί ανεπαρκής για κάποιο άλλο τμήμα, κάτι απόλυτα λογικό από τις ουσιώδης διαφορές που έχουν τα τμήματα μεταξύ τους και από το είδος των περιορισμών που έχει κάθε πανεπιστημιακό ίδρυμα.

Λέξεις Κλειδιά: << χρονοδιαγράμματος ,περιορισμοί, αυτοματοποιημένο, constraint programming, βελτιστοποιημένη λύση>>

Abstract

The problem of university written exam timetables is a well-known multidimensional constraint matching problem that focuses on assigning courses to faculty members in classes over a limited period of time. Therefore, it is a difficult time-consuming problem that universities face. This complexity is justified by the size of exam planning and the large number of restrictions and allocation criteria. Usually a long and hard job of providing an appropriate and optimized solution. We understand that it is imperative that the need for an automated information system be developed. This paper focuses on the development of an algorithm to automatically satisfy all constraints on the particular problem of the university timetable of the written examinations of the Department of Engineering Information and Communication Systems of the University. Emphasize that the algorithm covers most constraints for the needs of the particular university and can be considered insufficient for some other part, which is perfectly reasonable from the essential differences between the parts.

Περιεχόμενα

1.Εισαγωγή

1.1. Το πρόβλημα

Ένα λογισμικό προγραμματισμού των εξετάσεων έχει μεγάλη χρησιμότητα και η δημιουργία του προγράμματος αποτελεί μια αναγκαία, τακτική και χρονοβόρα διαδικασία σε όλα τα εκπαιδευτικά ιδρύματα. Αυτό βοηθά στη διεξαγωγή εξετάσεων των φοιτητών που καθορίζει πότε, πού και πώς θα διεξαχθεί η εξέταση του κάθε μαθήματος. Η δημιουργία ενός καλού λογισμικού χρονοδιαγράμματος εξέτασης που θα ικανοποιεί τους σπουδαστές, τους διδάσκοντες και τη διοίκηση είναι ένα πολύ δύσκολο έργο λόγο περιορισμένων πόρων. Αυτός ο περιορισμός καθιστά εξαιρετικά δύσκολο το χειρισμό πολλών μεταβλητών και την δημιουργία του προγράμματος.

Μερικές φορές οι όροι **πρόγραμμα**, **ακολουθία** και το **χρονοδιάγραμμα** χρησιμοποιούνται χαλαρά χωρίς να έχουμε περισσότερες γνώσεις. Παρόλα αυτά, μπορεί να υπάρξουν ορισμένες σημαντικές διαφορές μεταξύ αυτών των όρων.

Ένα **χρονοδιάγραμμα** δείχνει πότε πρέπει να γίνουν συγκεκριμένα γεγονότα. Δεν προϋποθέτει αναγκαστικά την κατανομή των πόρων. Έτσι σε ένα λεωφορείο ή τρένο το χρονοδιάγραμμα δείχνει πότε πρέπει να πραγματοποιηθούν δρομολόγια σε συγκεκριμένη διαδρομή ή δρομολόγια. Δεν μας λέει ποια οχήματα ή οδηγοί πρόκειται να ανατεθούν σε συγκεκριμένα ταξίδια. Η κατανομή των οχημάτων και των οδηγών αποτελεί μέρος της διαδικασίας προγραμματισμού. Αν και το χρονοδιάγραμμα αποτελεί αυστηρά τον σχεδιασμό του τρόπου ταξιδιών, αυτό το μοτίβο μπορεί να είναι σχεδιασμένο ως μέρος μιας διαδικασίας που έχει κατά νου αν είναι πιθανό ότι ένα αποτελεσματικό χρονοδιάγραμμα μπορεί να προσαρμοστεί στο προκύπτον σχέδιο διαδρομής.

Στον τομέα των σιδηροδρόμων, ο όρος χρονοδιάγραμμα χρησιμοποιείται συχνά για να αναφέρει την κατασκευή μιας διαδρομής (με χρόνους) για μια αμαξοστοιχία μέσω ενός συστήματος. Ένα χρονοδιάγραμμα κλάσης δείχνει πότε πρέπει να γίνουν συγκεκριμένα γεγονότα. Σε ένα σχολείο νηπίων, όπου Α ο μοναδικός δάσκαλος, ο οποίος είναι υπεύθυνος για όλες τις δραστηριότητες μιας συγκεκριμένης τάξης και όπου όλες αυτές οι δραστηριότητες πραγματοποιούνται στο ίδιο δωμάτιο, ένα χρονοδιάγραμμα δεν είναι τίποτα περισσότερο από μία δήλωση σχετικά με τις ώρες κατά τις οποίες θα πραγματοποιηθούν συγκεκριμένες δραστηριότητες. Σε αντίθεση, ένα χρονοδιάγραμμα εξετάσεων στο πανεπιστήμιο θα περιλαμβάνει κανονικά αναθέσεις χώρων με γνώση των μεγεθών των ομάδων και των αναγκαίων ειδικών εγκαταστάσεων. Σε ένα πανεπιστήμιο το ωρολόγιο πρόγραμμα πρέπει επίσης να λαμβάνει υπόψη τη διαθεσιμότητα των μεμονωμένων καθηγητών. Οι δραστηριότητες σύνταξης ωρολογίων εξετάσεων και πανεπιστημιακών μαθημάτων μπορεί να θεωρούνται δραστηριότητες προγραμματισμού.

Μια **ακολουθία** είναι απλώς μια σειρά στην οποία διεξάγονται οι δραστηριότητες. Παραδείγματος χάρη, η σειρά με την οποία οι εργασίες διεκπεραιώνονται μέσω των μηχανών ενός εργοστασίου, εάν οι εργασίες περνούν μέσα από κάθε μηχανή με την ίδια σειρά, είναι μια ακολουθία. Η ακολουθία μπορεί να λάβει υπόψη το κόστος που σχετίζεται με μια συγκεκριμένη εργασία που ακολουθείται ένα άλλο (π.χ., κόστος μετατροπής μηχανής). Το πρόβλημα της ταξινόμησης των θέσεων εργασίας σε αυτά είναι γνωστά ως προβλήματα ροής.

Ένα **πρόγραμμα** συνήθως περιλαμβάνει όλες τις ειδικές και χρονικές πληροφορίες οι οποίες είναι αναγκαίες για τη διεξαγωγή μιας διαδικασίας. Αυτό θα περιλαμβάνει και τις ώρες για τις δραστηριότητες, δηλώσεις σχετικά με τους πόρους που θα ανατεθούν καθώς και σχέδια εργασίας για μεμονωμένο προσωπικό ή μηχανές. Ο στόχος του προγραμματισμού με την ευρύτερη έννοια είναι η επίλυση πρακτικών προβλημάτων σχετικά με την κατανομή των πόρων σε αντικείμενα που υπόκεινται σε περιορισμούς και τοποθετούνται σε χωροχρόνους, χρησιμοποιώντας ή αναπτύσσοντας οποιαδήποτε εργαλεία μπορεί να είναι κατάλληλα. Τα προβλήματα συχνά σχετίζονται με την ικανοποίηση ορισμένων στόχων.

1.2. Παρουσίαση Προβλήματος

Το μεγαλύτερο πρόβλημα στην επίλυση του προβλήματος είναι ο μεγάλος αριθμός περιορισμών με το μεγάλο πλήθος μαθημάτων σε συνδυασμό με την ανάγκη να γίνεται επιλογή από τον κάθε καθηγητή για την ημέρα εξέτασης του μαθήματός του. Στην έρευνα τεχνητής νοημοσύνης και επιχειρήσεων, η ικανοποίηση των περιορισμών είναι η διαδικασία εξεύρεσης λύσης σε μια σειρά περιορισμών που επιβάλλουν συνθήκες που πρέπει να πληρούν οι μεταβλητές.

Μια λύση είναι επομένως ένα σύνολο τιμών για τις μεταβλητές που ικανοποιούν όλους τους περιορισμούς - δηλαδή ένα σημείο στην εφικτή περιοχή. Οι τεχνικές που χρησιμοποιούνται για την ικανοποίηση των περιορισμών εξαρτώνται από το είδος των περιορισμών που εξετάζονται. Συχνά χρησιμοποιούνται περιορισμοί σε ένα πεπερασμένο τομέα, στο σημείο που τα προβλήματα ικανοποίησης περιορίζονται συνήθως με προβλήματα που βασίζονται σε περιορισμούς σε πεπερασμένο τομέα. Τέτοια προβλήματα συνήθως επιλύονται μέσω της αναζήτησης, ιδίως μιας μορφής ανασχέσεως ή τοπικής αναζήτησης.

Η διάδοση περιορισμών είναι μία από τις μεθόδους που χρησιμοποιούνται σε τέτοια προβλήματα. Τα περισσότερα από αυτά είναι εν γένει ελλιπή, δηλαδή, μπορεί να λύσουν το πρόβλημα ή να αποδειχθούν μη ικανοποιητικά, αλλά όχι πάντα. Οι μέθοδοι διάδοσης των περιορισμών χρησιμοποιούνται επίσης σε συνδυασμό με την αναζήτηση για να καταστεί ένα δεδομένο πρόβλημα απλούστερο για την επίλυση. Άλλοι θεωρούμενοι τύποι περιορισμών είναι σε πραγματικούς ή λογικούς αριθμούς.

1.3. Περιορισμοί

Όπως σε κάθε πρόβλημα χρονοπρογραμματισμού, έτσι για την δημιουργία του ωρολόγιού προγράμματος της εξεταστικής έπρεπε να ληφθούν υπόψη κάποιοι περιορισμοί που στη συνέχεια θα επιλύονταν.

Συγκεκριμένα:

1. Κάθε καθηγητής μπορεί να διαλέξει ένα εύρος τιμών εντός συγκεκριμένων ημερών (αρχή και τέλος εξεταστικής) όπου θα πρέπει μέσα σε αυτό το διάστημα να κατανεμηθούν τα μαθήματα του που θα είναι προς εξέταση.
2. Μαθήματα ίδιου εξαμήνου να μην εξετάζονται την ίδια μέρα.
3. Δημιουργία βαρδιών σύμφωνα με τις κατάλληλες ώρες και κάθε βάρδια να είναι μοναδική (δηλαδή ένα μάθημα ανά 3ωρο ..9-12,12-3,3-6,6-9).
4. Κάθε μάθημα να αντιστοιχίζεται σε μια βάρδια μόνο μέσα σε μία ημέρα.
5. Αντιστοίχηση μαθημάτων με καθηγητές, μαθήματα με εξάμηνα (μέσα από ειδικές λίστες).
6. Καθηγητές από Αθήνα που θέλουν να αναχωρήσουν αεροπορικώς την τελευταία μέρα εξέτασης να μην εξετάζονται μετά τις 3 την τελευταία μέρα του εύρους που έχουν δηλώσει.
7. Ώρες αιθουσών σύμφωνα με τις ώρες που έχουμε διαθέσιμες από γραμματεία.
8. Διαμοιρασμός των φοιτητών στις κατάλληλες αίθουσες (πχ <80 ή εάν το μάθημα είναι υποχρεωτικό να εξετάζεται κατά κανόνα σε μεγάλη αίθουσα λόγω πλήθους).

1.4. Σχετικές εργασίες

Όσον αφορά τα προβλήματα χρονοδιαγράμματος, πολλά έργα έχουν γίνει για να επιτευχθεί μια καλή λύση για τη χρήση τους σε διαφορετικές προσεγγίσεις επιχειρησιακής έρευνας (Operations Research-OR) [1] και τεχνητής νοημοσύνης (Artificial Intelligence-AI) [2]. Ένας μεγάλος αριθμός από τα προβλήματα στην AI και σε άλλους τομείς της επιστήμης των υπολογιστών μπορούν να θεωρηθούν ως ειδικές περιπτώσεις των προβλημάτων ικανοποίησης περιορισμών. Η ανάπτυξη αποτελεσματικών τεχνικών λύσεων για τα προβλήματα ικανοποίησης περιορισμού είναι ένα σημαντικό ερευνητικό πρόβλημα. Τα προβλήματα προγραμματισμού θεωρούνται ως συνδυαστικά προβλήματα στο OR, τα οποία μπορούν να διαμορφωθούν ως προβλήματα ικανοποίησης περιορισμών (Constraint Satisfaction Problems- CSPs) .

- Αρχικά, οι Abdennadher και Marte [3] έδειξαν πώς να μοντελοποιήσουν το πρόβλημα του χρονοδιαγράμματος ως μερικό πρόβλημα ικανοποίησης περιορισμού και έδωσαν έναν συνοπτικό solver πεπερασμένων τομέων .

- Με παρόμοιο τρόπο, οι Shue, Lin και Tsai [4] ανέπτυξαν ένα σύστημα υποστήριξης αποφάσεων για το χρονοδιάγραμμα των πανεπιστημίων όπου θεωρεί τόσο αυστηρούς όσο και εύκαμπτους περιορισμούς του προβλήματος. Οι αυστηροί περιορισμοί είναι εκείνοι που σίγουρα θέλουμε να είναι αληθής σε διαφορετική περίπτωση το πρόγραμμα μπορεί να μην καταλήξει σε κάποια λύση. Οι εύκαμπτοι περιορισμοί είναι εκείνοι που θα θέλαμε να είναι αληθής- όχι όμως εις βάρος των άλλων. Δήλωσαν ότι το σύστημά τους έχει ικανοποιήσει όλους τους αυστηρούς περιορισμούς. Προτυποποίησαν το πρόβλημα ως ένα ικανοποιητικό πρόβλημα περιορισμού για την εφαρμογή της διαδικασίας λύσης όπου κάθε εύκαμπτος περιορισμός αντιμετωπίζεται ως στόχος με προτεραιότητα.

- Επιπλέον, ο Gavanelli [5] σχεδίασε ένα πανεπιστημιακό χρονοδιάγραμμα πρόβλημα ως Πρόβλημα Βελτιστοποίησης Περιορισμού (Constraint Optimization Problems -COP) και απευθύνθηκε στο μοντέλο του με τη γλώσσα CLP [6], ECLiPSe [7]. Το έγγραφό του έδειξε πώς εκπροσωπούνταν τα δεδομένα στην ECLiPSe και ποιες ήταν οι φάσεις για την κατάρτιση ενός χρονοδιαγράμματος

2. Προβλήματα ικανοποίησης περιορισμών(CSP)

2.1. Γενική Μελέτη

Τα προβλήματα ικανοποίησης περιορισμών (CSPs) είναι μαθηματικά ερωτήματα που ορίζονται ως σύνολο αντικειμένων των οποίων η κατάσταση πρέπει να ικανοποιεί ορισμένους περιορισμούς. Τα CSPs αντιπροσωπεύουν τις οντότητες σε ένα πρόβλημα ως μια ομοιογενή συλλογή πεπερασμένων περιορισμών έναντι των μεταβλητών, η οποία επιλύεται με μεθόδους ικανοποίησης περιορισμών.

Τα CSPs αποτελούν το αντικείμενο έντονης έρευνας τόσο στην τεχνητή νοημοσύνη όσο και στην έρευνα των λειτουργιών, καθώς η κανονικότητα στη διατύπωσή τους παρέχει μια κοινή βάση για την ανάλυση και επίλυση προβλημάτων πολλών φαινομενικά άσχετων οικογενειών. Τα CSP συχνά παρουσιάζουν μεγάλη πολυπλοκότητα, απαιτώντας έναν συνδυασμό ευρετικών και συνδυαστικών μεθόδων αναζήτησης που πρέπει να λυθούν σε εύλογο χρονικό διάστημα.

Ο Προγραμματισμός Περιορισμών (CP) είναι το πεδίο έρευνας που επικεντρώνεται ειδικά στην αντιμετώπιση αυτού του είδους των προβλημάτων. Επιπρόσθετα, όλα τα πεδία έρευνας που επικεντρώνονται στην επίλυση συγκεκριμένων μορφών του προβλήματος ικανοποίησης περιορισμού είναι το πρόβλημα ικανοποίησης ικανοτήτων boolean (SAT), οι θεωρίες modulo (SMT), ο μικτός προγραμματισμός (MIP) και ο προγραμματισμός των απαντήσεων (ASP).

2.2. Ορισμός

Ένας πιο επίσημος **ορισμός** θα ήταν πώς:

ΟΡΙΣΜΌΣ 2.1: Ένα πρόβλημα ικανοποίησης περιορισμού (ή CSP) ορίζεται από ένα σύνολο μεταβλητών, $X_1, X_2, \ldots, X_n$, και ένα σύνολο περιορισμών, $C_1, C_2, \ldots, C_m$. Κάθε μεταβλητή X_i έχει μη αποδεκτή περιοχή D_i των πιθανών τιμών. Κάθε περιορισμός C_i περιλαμβάνει ορισμένο υποσύνολο των μεταβλητών και καθορίζει τους επιτρεπόμενους συνδυασμούς τιμών για αυτό το υποσύνολο. Μια κατάσταση

του προβλήματος καθορίζεται από την εκχώρηση της αξίας ορισμένων ή όλων των μεταβλητών, $\{X_i = U_i , X_j = U_j,\ldots\}$. Μια εργασία που δεν παραβιάζει οποιονδήποτε περιορισμό ονομάζεται συνεπής ή νόμιμη ανάθεση.

- Μια πλήρης ανάθεση είναι αυτή στην οποία αναφέρεται κάθε μεταβλητή και μια λύση σε ένα CSP είναι μια πλήρης ανάθεση που ικανοποιεί όλους τους περιορισμούς. Ορισμένα CSPs επίσης απαιτούν μια λύση που μεγιστοποιεί μια αντικειμενική λειτουργία.

- Η ύπαρξη λύσης σε ένα CSP μπορεί να θεωρηθεί ως πρόβλημα απόφασης. Αυτό μπορεί να αποφασιστεί με την εξεύρεση λύσης ή με την εξεύρεση λύσης μετά από εξαντλητική αναζήτηση (οι στοχαστικοί αλγόριθμοι συνήθως δεν καταλήγουν σε εξαντλητικό συμπέρασμα, ενώ συχνά αναζητούν κατευθυνόμενες αναζητήσεις σε αρκετά μικρά προβλήματα). Σε ορισμένες περιπτώσεις μπορεί να είναι γνωστό ότι ο CSP έχει λύσεις εκ των προτέρων, μέσω κάποιας άλλης διαδικασίας μαθηματικών συμπερασμάτων.

2.3. Παραδείγματα

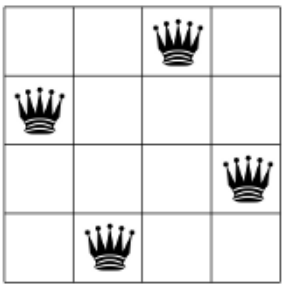
Παραδείγματα απλών προβλημάτων που μπορούν να διαμορφωθούν ως πρόβλημα ικανοποίησης περιορισμού περιλαμβάνουν:

- *N- βασίλισσες παζλ* [8]

  Στο σκάκι, μια βασίλισσα μπορεί να επιτεθεί οριζόντια, κατακόρυφα και διαγώνια. Το πρόβλημα N -βασίλισσες ζητάει:

  - ***Πώς μπορούν να τοποθετηθούν οι N-βασίλισσες σε μια σκακιέρα NxN έτσι ώστε να μην επιτεθούν η μία στην άλλη;***

  Στην επόμενη ενότητα, θα δοθεί μια πιθανή λύση στο πρόβλημα N - βασίλισσες για N = 4.



Δεν υπάρχουν δύο βασίλισσες στην ίδια σειρά, στήλη ή διαγώνιο.

- *Σημείωση ότι αυτό δεν είναι ένα πρόβλημα βελτιστοποίησης: πρέπει να βρεθούν όλες οι πιθανές λύσεις, παρά μια βέλτιστη λύση, που το καθιστά φυσικό υποψήφιο για προγραμματισμό περιορισμού.*

- *Πρόβλημα χρωματισμού χάρτη* [9]*:*

Ο χρωματισμός του γραφήματος είναι ένα από τα πιο συνδυαστικά προβλήματα που μελετήθηκαν στην τεχνητή νοημοσύνη αφού πολλές πραγματικές εφαρμογές όπως καταχώρηση χρόνου και κατανομή συχνότητας μπορούν εύκολα να διατυπωθούν ως πρόβλημα χρώματος γραφήματος. Ο στόχος σε αυτό το πρόβλημα είναι να χρωματιστούν όλοι οι κόμβοι ενός γραφήματος έτσι ώστε κάθε δύο γειτονικές κορυφές πρέπει να έχουν πάρει διαφορετικά χρώματα ,όπου κάθε κόμβος έχει πεπερασμένο αριθμό πιθανών χρωμάτων. Το πρόβλημα χρωματισμού γραφήματος είναι τυποποιημένο ως CSP. Ως εκ τούτου, οι κόμβοι του γραφήματος είναι οι μεταβλητές για το χρώμα και τα πιθανά χρώματα κάθε κόμβου / μεταβλητής δημιουργούν τον τομέα. Υπάρχει ένας περιορισμός μεταξύ κάθε ζεύγους παρακείμενων μεταβλητών / κόμβων που απαγορεύει αυτές τις μεταβλητές να έχουν το ίδιο χρώμα.

Για παράδειγμα έστω στον χάρτη της Αυστραλίας που δείχνει το παραπάνω Σχήμα, ότι μας δίνεται το καθήκον να χρωματίζουμε κάθε περιοχή είτε κόκκινο, πράσινο ή μπλε έτσι ώστε οι γειτονικές περιοχές να μην έχουν το ίδιο χρώμα. Για να το διατυπώσουμε ως CSP, ορίζουμε τις μεταβλητές ώστε να αντικατοπτρίζουν τις περιοχές: WA, NT, Q, NSW, V, SA και T. Ο τομέας κάθε μεταβλητής είναι το σύνολο {κόκκινο, πράσινο, μπλε}. Οι περιορισμοί απαιτούν οι γειτονικές περιοχές να έχουν ξεχωριστά χρώματα.

Για παράδειγμα, οι επιτρεπόμενοι συνδυασμοί για WA και NT είναι τα ζεύγη

{(κόκκινο, πράσινο), (κόκκινο, μπλε), (πράσινο, κόκκινο), (πράσινο, μπλε), (μπλε, κόκκινο), (μπλε, πράσινο)}.

Υπάρχουν πολλές πιθανές λύσεις, όπως:

{WA = κόκκινο, NT = πράσινο, Q = κόκκινο, NSW = πράσινο, V = κόκκινο, SA = μπλε, T = κόκκινο}. .

- *Sudoku* [10]*:*

Το Sudoku είναι ένα πλέγμα 9x9, ομαδοποιημένο σε 3x3 πλέγματα από 3x3 μπλοκ, όπου κάθε τετράγωνο στο πλέγμα πρέπει να συμπληρώνεται με ένα ψηφίο από 1 έως 9 έτσι ώστε κάθε σειρά, στήλη και μπλοκ να περιέχει το κάθε ψηφίο ακριβώς μία φορά. Το πλέγμα είναι γεμάτο έτσι ώστε να υπάρχει εγγύηση ότι υπάρχει ένα μοναδικό.

Αρχικά έχει την παρακάτω μορφή:

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

Ενώ ένα ολοκληρωμένο Sudoku θα είναι της μορφής :

7	2	6	4	9	3	8	1	5
3	1	5	7	2	8	9	4	6
4	8	9	6	5	1	2	3	7
8	5	2	1	4	7	6	9	3
6	7	3	9	8	5	1	2	4
9	4	1	3	6	2	7	5	8
1	9	4	8	3	6	5	7	2
5	6	7	2	1	4	3	8	9
2	3	8	5	7	9	4	6	1

Μεταβλητές:

Ψάχνουμε για 81 μεταβλητές που είναι διατεταγμένες σε ένα πλέγμα 9 × 9, C_{ij} αντιπροσωπεύει την τιμή του στοιχείου στην i-σειρά και τη στήλη j, όπου i = 1, ..., 9 και j = 1, ..., 9.

Πεδίο Ορισμού αυτών των μεταβλητών:

Το C_{ij} μπορεί να πάρει οποιαδήποτε ακέραια τιμή μεταξύ 1 και 9.

Περιορισμοί:

- Για κάθε σειρά i, όλες οι τιμές σε αυτή τη σειρά πρέπει να είναι διαφορετικές.
- Για κάθε στήλη j, όλες οι τιμές στη στήλη αυτή πρέπει να είναι διαφορετικές.
- Για κάθε μπλοκ Β 3 × 3, όλες οι τιμές αυτού του μπλοκ πρέπει να είναι διαφορετικές.

Στη γενική περίπτωση, τα προβλήματα περιορισμού μπορούν να είναι πολύ δυσκολότερα και μπορεί να μην είναι εμφανή σε κάποια από αυτά τα απλούστερα συστήματα. Τα παραδείγματα "πραγματικής ζωής" περιλαμβάνουν αυτοματοποιημένο σχεδιασμό, λεξικολογική αποσαφήνιση, μουσικολογία και κατανομή πόρων.

3. Ανάλυση όρου Προγραμματισμός με Περιορισμούς (CP)

3.1. Μελέτη του όρου

Η βελτιστοποίηση περιορισμού ή ο προγραμματισμός με περιορισμούς (CP) είναι το όνομα που δίνεται για τον εντοπισμό εφικτών λύσεων από ένα πολύ μεγάλο σύνολο υποψηφίων, όπου το πρόβλημα μπορεί να διαμορφωθεί με όρους αυθαίρετων περιορισμών. Τα προβλήματα CP προκύπτουν σε πολλά επιστημονικά και μηχανικά θέματα. (Η λέξη "προγραμματισμός" είναι ένα κομμάτι μίας εσφαλμένης ονομασίας, παρόμοια με την έννοια που ο "υπολογιστής" κάποτε εννοούσε "ένα άτομο που υπολογίζει." Εδώ, ο "προγραμματισμός" αναφέρεται στη ρύθμιση ενός σχεδίου και όχι στον προγραμματισμό σε μια γλώσσα υπολογιστή).

Ο προγραμματισμός με περιορισμούς (Constraint Programming -CP) αποτελεί δηλαδή παράδειγμα για την επίλυση συνδυαστικών προβλημάτων που βασίζονται σε ένα ευρύ φάσμα τεχνικών από την τεχνητή νοημοσύνη, την επιστήμη των υπολογιστών και την έρευνα λειτουργιών. Στον προγραμματισμό με περιορισμούς, οι χρήστες αναφέρουν δηλωτικά τους περιορισμούς στις εφικτές λύσεις για ένα σύνολο μεταβλητών απόφασης.

Οι περιορισμοί διαφέρουν από τα κοινά πρωτεύοντα των γλωσσών προγραμματισμού στο ότι δεν καθορίζουν ένα βήμα ή ακολουθία βημάτων προς εκτέλεση, αλλά μάλλον τις ιδιότητες μιας λύσης που θα βρεθεί. Θα πρέπει να ληφθεί υπόψη ότι ο CP βασίζεται στη σκοπιμότητα (εύρεση μιας εφικτής λύσης) και όχι στη βελτιστοποίηση (βρίσκοντας μια βέλτιστη λύση) και επικεντρώνεται στους περιορισμούς και τις μεταβλητές και όχι στην αντικειμενική λειτουργία. Στην πραγματικότητα, ένα πρόβλημα CP μπορεί να μην έχει καν αντικειμενική λειτουργία - ο στόχος μπορεί απλώς να περιορίσει μια μεγάλη ποικιλία πιθανών λύσεων σε ένα πιο διαχειρίσιμο υποσύνολο προσθέτοντας περιορισμούς στο πρόβλημα. Για παράδειγμα όταν η κάθε λύση εξάγεται σε αντίστοιχο αρχείο θα θέλαμε να περιορίσουμε τον αριθμό των αρχείων που θα δημιουργηθούν όταν αυτός είναι πολύ μεγάλος, δημιουργώντας κάποιους πρόσθετους περιορισμούς.

Σε προσθήκες σε περιορισμούς, οι χρήστες πρέπει επίσης να καθορίσουν μια μέθοδο για την επίλυση αυτών των περιορισμών. Αυτό τυπικά βασίζεται σε τυποποιημένες μεθόδους όπως :

1. Διάδοση(Propagation)
2. οπισθοδρόμηση(backtracking)

**Κυρίως, οι περιορισμοί εφαρμόζονται σε γλώσσες μέσω εργαλείων επίλυσης περιορισμών, οι οποίες είναι ξεχωριστές βιβλιοθήκες για μια υπάρχουσα γλώσσα.*

3.2. Βιβλιοθήκες προγραμματισμού περιορισμών για γλώσσες προγραμματισμού

Ο προγραμματισμός με περιορισμούς συχνά πραγματοποιείται στον προγραμματισμό μέσω μιας ξεχωριστής βιβλιοθήκης. Ορισμένες δημοφιλείς βιβλιοθήκες για τον προγραμματισμό με περιορισμούς είναι:

- Artelys Kalis, C++, Java, Python library, FICO Xpress module (proprietary)
- Cassowary, Smalltalk, C++, Java, Python, JavaScript, Ruby library (LGPL)
- CHIP V5, C++ and C libraries (proprietary)
- Choco, Java library (BSD license)
- Comet, C style language for constraint programming, constraint-based local search and mathematical programming (free binaries available for academic use)
- Cream, Java library (LGPL)
- Disolver, C++ library (proprietary)
- Facile, OCaml library (CC0 1.0)
- finite-domain, Haskell library (MIT)
- Gecode, C++ library, Python bindings (X11-style free software)
- Google OR-Tools, C++, Python, Java, .NET library (Apache License 2.0)
- IBM ILOG CP Optimizer: C++, Python, Java, .NET libraries (proprietary, free for academic use).[6] Successor of ILOG Solver/Scheduler, which was considered the market leader in commercial constraint programming software as of 2006[7]
- JaCoP, Java library (open source)

- JOpt, Java library (free software)
- JSR-331, Java constraint programming API, JCP standard
- Koalog Constraint Solver, Java library (proprietary)
- LINDO (proprietary)
- MonadicCP, Haskell library (BSD-3-Clause)
- Numberjack, Python platform (LGPL)
- Minion, C++ program (GPL)
- python-constraint, Python library (GPL)
- OscaR, Scala library (LGPL)
- Scarab, Scala library (BSD license)
- SMOCS, Scala Monadic library (BSD license)
- OptaPlanner, Java library (that also works in Kotlin, Scala, JRuby and Groovy) and toolkit (benchmarker, server and workbench) (Apache license)
- WSimply
- Z3, C++ solver with C, Java, C#, and Python bindings (MIT license)

3.3. Το πρόβλημα των N -βασίλισσών

Ο CP Solver λειτουργεί με τη συστηματική δοκιμή όλων των πιθανών εκχωρήσεων τιμών στις μεταβλητές ενός προβλήματος, για να βρει τις εφικτές λύσεις. Στο πρόβλημα N -βασίλισσες [8]το οποίο αναφέραμε στην προηγούμενη ενότητα, ο solver ξεκινά από την αριστερή στήλη και τοποθετεί διαδοχικά μία βασίλισσα σε κάθε στήλη, σε μια θέση που δεν δέχεται επίθεση από προγενέστερα τοποθετημένες βασίλισσες.

- *Υπάρχουν δύο βασικά στοιχεία σε μια αναζήτηση προγραμματισμού περιορισμού:*

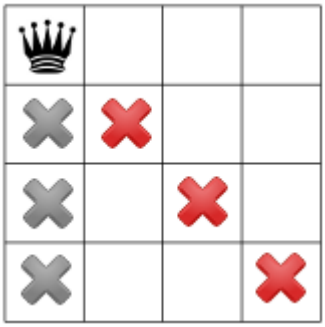
Διάδοση (Propagation) : Κάθε φορά που ο CP Solver αποδίδει μια τιμή σε μια μεταβλητή, οι περιορισμοί προσθέτουν περιορισμούς στις πιθανές τιμές των μη εκχωρημένων μεταβλητών. Αυτοί οι περιορισμοί μεταδίδονται στις μελλοντικές μεταβλητές αναθέσεις. Για παράδειγμα, στο 4 -βασίλισσες πρόβλημα, κάθε φορά

που ο solver τοποθετεί μια βασίλισσα, δεν μπορεί να τοποθετήσει άλλες βασίλισσες στη σειρά και στις διαγώνιες της τρέχουσας θέσης της βασίλισσας. Ο πολλαπλασιασμός μπορεί να επιταχύνει σημαντικά την αναζήτηση μειώνοντας το σύνολο μεταβλητών τιμών που πρέπει να διερευνήσει ο διαλυτής.

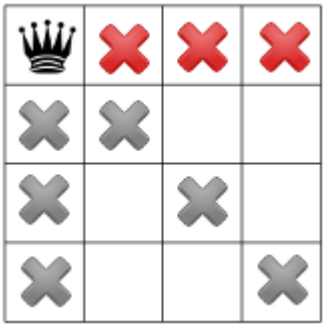
Οπισθοδρόμηση (Backtracking) : Προκύπτει όταν είτε ο CP Solver δεν μπορεί να εκχωρήσει μια τιμή στην επόμενη μεταβλητή, λόγω των περιορισμών, είτε βρίσκει λύση. Σε κάθε περίπτωση, ο CP Solver επιστρέφει σε ένα προηγούμενο στάδιο και αλλάζει την τιμή της μεταβλητής σε μια τιμή που δεν έχει δοκιμαστεί. Στο παράδειγμα N -βασίλισσες, αυτό σημαίνει ότι μετακινείτε μια βασίλισσα σε ένα νέο τετράγωνο στην τρέχουσα στήλη.

Αρχικά,

A. Ο CP Solver ξεκινά τοποθετώντας την πρώτη βασίλισσα στην επάνω αριστερή γωνία. Λόγω του μοντέλου που δώσαμε στον CP Solver , γνωρίζει ότι δεν υπάρχει άλλη βασίλισσα στην ίδια στήλη, εξ' ου και οι γκρι σταυροί στο παρακάτω σχήμα. Ένας περιορισμός λέει στον CP Solver ότι δεν μπορεί να υπάρξει άλλη βασίλισσα στην ίδια διαγώνιο με αρνητική κλίση (οι διαγώνιοι που κατεβαίνουν τότε δεξιά). Οι κόκκινοι σταυροί δείχνουν αυτή την αδυναμία.

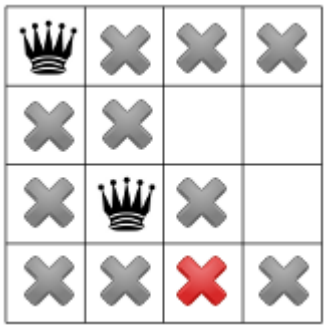


B. Ένας περιορισμός λέει στον CP Solver ότι δεν υπάρχουν δύο βασίλισσες στην ίδια σειρά, εξ ου και οι επόμενοι κόκκινοι σταυροί.

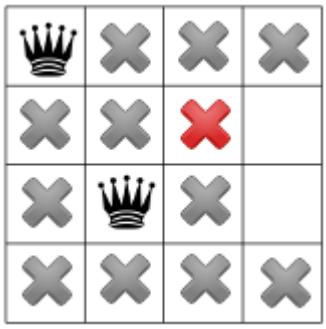


Μετά από αυτό το πρώτο βήμα, μόνο τα λευκά τετράγωνα είναι ακόμα διαθέσιμα για να τοποθετήσετε τις τρεις παραμένοντες βασίλισσες. Η διαδικασία εξαίρεσης ορισμένων τετραγώνων είναι αυτή που ονομάζεται διάδοση (Propagation**)**.

C. Στη συνέχεια προσπαθεί να τοποθετήσει μια δεύτερη βασίλισσα, το οποίο και κάνει τοποθετώντας στο αμέσως επόμενο διαθέσιμο τετράγωνο. Όπως και στο πρώτο βήμα, ο CP Solver ξέρει ότι καμία άλλη βασίλισσα δεν μπορεί να τοποθετηθεί σε μια στήλη όπου τοποθετήθηκε μια βασίλισσα, εξ ου και οι νέοι γκρι σταυροί.



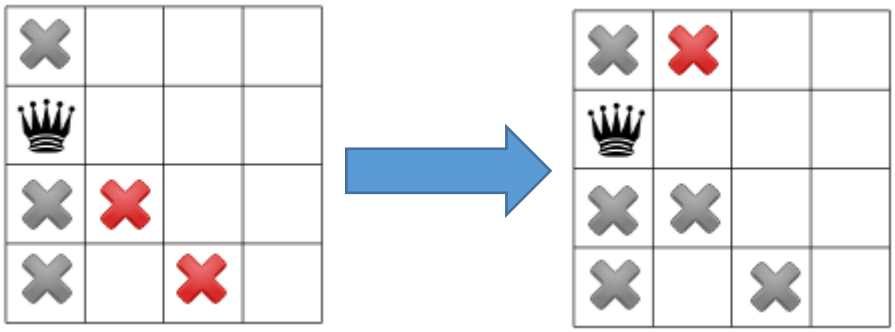
D. Ένας άλλος περιορισμός για τις διαγώνιες με θετικές κλίσεις λέει στον CP Solver ότι καμία βασίλισσα δεν μπορεί να τοποθετηθεί στη θετική διαγώνιο της δεύτερης βασίλισσας, εξ ου και του κόκκινου σταυρού.



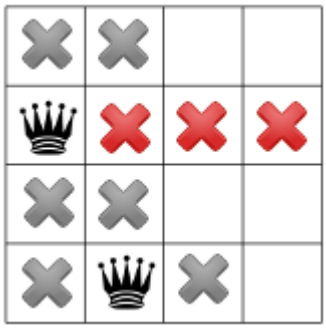
Τελικά έχουμε αποτυχία, καθώς δεν υπάρχει καμία δυνατότητα να τοποθετηθεί μια τρίτη βασίλισσα στην τρίτη στήλη και απλά δεν μπορεί να υπάρξει λύση με αυτή τη

διαμόρφωση. Εφόσον ο CP Solver δεν μπορεί να εκχωρήσει μια τιμή στην επόμενη μεταβλητή, λόγω των περιορισμών πρέπει να οπισθοδρομήσει (backtracking) !

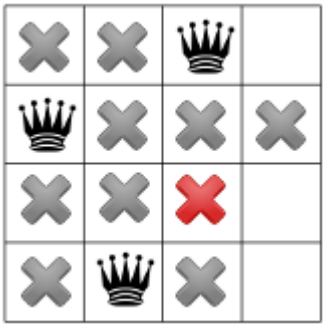
E. Σε αυτό το στάδιο προσπαθεί να αμφισβητήσει την τελευταία επιλογή της για τη δεύτερη βασίλισσα, αλλά ανιχνεύει ότι δεν υπάρχουν άλλες επιλογές. Ο CP Solver πρέπει να αμφισβητήσει την πρώτη του επιλογή κάνει backtracking και τοποθετεί την πρώτη βασίλισσα στη δεύτερη στήλη της πρώτης στήλης εφόσον δεν βρέθηκε κάποια λύση με την πρώτη τοποθέτηση.



F. Υπάρχει ο περιορισμός <<όχι 2 βασίλισσές στην ίδια σειρά>> και είναι υπεύθυνος για την αφαίρεση των επόμενων τριών τετραγώνων που σημειώνονται με κόκκινους σταυρούς:

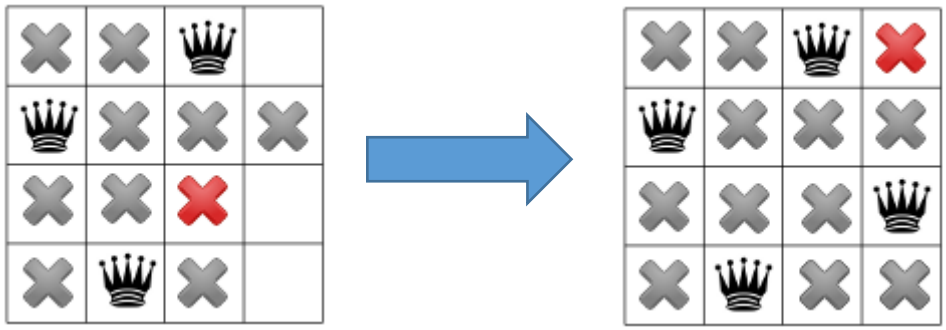


G. Ο θετικός διαγώνιος περιορισμός δικαιολογεί το κόκκινο τετράγωνο και με αυτό τον τρόπο δεν έχει καμία άλλη επιλογή από το να τοποθετήσει μια τρίτη βασίλισσα στην πρώτη στήλη της τρίτης στήλης.



H. Τέλος, ο περιορισμός "όχι δύο βασίλισσες στην ίδια σειρά" απαγορεύει οποιαδήποτε άλλη βασίλισσα να τοποθετηθεί στην τέταρτη σειρά, μη αφήνοντας άλλη επιλογή παρά να τοποθετήσει την τέταρτη βασίλισσα στην τέταρτη στήλη τρίτη σειρά.

I.



Ο CP Solver βρίσκει μία λύση που είναι εφικτή, οπότε έχουμε την πρώτη μας λύση! Αν ο CP Solver συνεχίσει την αναζήτηση, θα πρέπει να επιστρέψει και να προσπαθήσει να τοποθετήσει την πρώτη βασίλισσα στην τρίτη σειρά της πρώτης στήλης ή σε διαφορετική περίπτωση να του δώσουμε εντολή να σταματήσει μετά την εξεύρεση της πρώτης εφικτής λύσης.

4. Επίλυση προβλήματος περιορισμών/ συγκρούσεων

Σε πολλές εφαρμογές πραγματικής ζωής, δεν θέλουμε να βρούμε απλά μια εφικτή λύση αλλά θέλουμε την βέλτιστη εφικτή λύση. Η ποιότητα μιας λύσης συνήθως μετριέται με κάποια εφαρμογή εξαρτώμενης λειτουργίας που ονομάζεται Αντικειμενική Συνάρτηση(objective function). Ο στόχος είναι να βρεθεί μια τέτοια λύση που ικανοποιεί όλους τους περιορισμούς και ελαχιστοποιεί ή μεγιστοποιεί την αντικειμενική συνάρτηση αντίστοιχα. Τέτοια προβλήματα ονομάζονται Constraint Satisfaction Optimisation Problems (CSOP).

Ορισμός 4.1:

Το πρόβλημα (CSOP) είναι ένας Γράφος περιορισμών 4 μεταβλητών Φ = (V, D, C, f) όπου (V, D, C) το τυπικό CSP και το f είναι μια αντικειμενική συνάρτηση που χαρτογραφεί κάθε συνεπής εκχώρηση σε αριθμητική τιμή.

Ο στόχος είναι να βρεθεί μια τέτοια λύση που να είναι βέλτιστη σε σχέση με την Αντικειμενική Συνάρτηση f, δηλ. ελαχιστοποιεί ή μεγιστοποιεί την αντικειμενική λειτουργία.

Ορισμός 4.2:

Μια λύση στον περιορισμό (CSOP) Φ = (V, D, C, f) είναι μία (τοπικά) μέγιστη συνεπής εκχώρηση σ για (V, D, C) έτσι ώστε η αντικειμενική συνάρτηση f (σ) είναι ελάχιστη (ή μέγιστη).

Σύμφωνα με τα παραπάνω, ένα πρόβλημα CSP μμπορεί να αναπαρασταθεί σαν ένας Γράφος: [11]

Φ=(V, D, C) όπου :

- V={v_1,…, v_n} είναι το σύνολο των μεταβλητών, όπου κάθε μεταβλητή αντιστοιχεί σε έναν κόμβο του γράφου.

- D={d_1,…, d_n} είναι το σύνολο των πεδίων, όπου κάθε τιμή του πεδίου αντιπροσωπεύει και ένα χρώμα στο γράφο.

- C={c_1,…, c_n} είναι το σύνολο των περιορισμών, όπου η σχέση C=C{i,j}⊆D_i*D_j περιγράφει τον περιορισμό μμεταξύ των μμεταβλητών (v_i,v_j) όταν έχουν τις ίδιες τιμές και (i,j) είναι δύο κοντινοί κόμβοι (μεταβλητές) του γράφου. Τότε ισχύει ότι v_i≠v_j αν (i,j) είναι κοντινοί κόμβοι.

Ο προγραμματισμός με περιορισμούς λειτουργεί πρώτα με στόχο να μειώσει το σύνολο των πιθανών τιμών των μεταβλητών απόφασης που ικανοποιούν όλους τους περιορισμούς χρησιμοποιώντας λογικά, γραφικά-θεωρητικά, αριθμητικά και άλλα επιχειρήματα. Όταν ορισμένες τιμές από το πεδίο ορισμού της μεταβλητής απόφασης δεν είναι δυνατές, αυτές οι πληροφορίες διαδίδονται μέσω των περιορισμών. Διαφορετικές στρατηγικές αναζήτησης χρησιμοποιούνται επίσης, μέχρις ότου εκχωρηθεί μια τιμή σε κάθε μεταβλητή απόφασης, δηλαδή έως ότου βρεθεί μια λύση. Αφού βρεθεί μια πρώτη λύση, η αναζήτηση προχωρά στην εξεύρεση περαιτέρω εφικτών λύσεων που βελτιώνουν την τιμή της αντικειμενικής συνάρτησης.

Boolean satisfiability problem

Στη λογική και την επιστήμη των υπολογιστών, το πρόβλημα Ικανοποίησης Λογικών Εκφράσεων [12](μερικές φορές ονομάζεται με συντομογραφία SATISFIABILITY ή SAT) είναι το πρόβλημα του προσδιορισμού εάν υπάρχει μια ερμηνεία που ικανοποιεί μια δεδομένη λογική πρόταση μέσω ανάθεση τιμών {0, 1} στις λογικές μεταβλητές . Με άλλα λόγια, ερωτά αν οι μεταβλητές μιας λογικής πρότασης Boolean μπορούν να αντικατασταθούν με τις τιμές TRUE ή FALSE με

τέτοιο τρόπο ώστε η λογική πρόταση να αξιολογηθεί ως TRUE. Εάν συμβαίνει αυτό, η λογική πρόταση καλείται ικανοποιήσιμη. Από την άλλη πλευρά, αν δεν υπάρχει τέτοια ανάθεση τιμών , η λογική πρόταση που εκφράζεται από τον τύπο είναι FALSE για όλες τις δυνατές αναθέσεις τιμών στις μεταβλητές και η λογική πρόταση είναι μη ικανοποιήσιμη.

Για παράδειγμα,

Η λογική πρόταση "a AND NOT b" είναι ικανοποιήσιμη επειδή μπορεί κανείς να βρει τις τιμές a = TRUE και b = FALSE, που κάνουν (a AND NOT b) = TRUE. Αντίθετα, το "a AND NOT a" δεν είναι ικανοποιήσιμο.

Ορισμένα συνδυαστικά προβλήματα επιλύονται από έναν αλγόριθμο ο οποίος ο χρόνος λειτουργίας περιορίζεται από ένα πολυώνυμο στο μέγεθος της αναπαράστασης του προβλήματος. Αυτά τα προβλήματα θεωρούνται πολυωνυμικού χρόνου και λέγεται ότι ανήκουν στην τάξη P [13]. Για άλλα προβλήματα ,τέτοια μέθοδος δεν είναι γνωστή να υπάρχει και ταξινομούνται ως εξής. Δύο πιο γνωστές κλάσεις είναι η κλάση αιτιοκρατικού πολυωνυμικού χρόνου P(polynomial) και η κλάση μη αιτιοκρατικού πολυωνυμικού χρόνου NP(Non-deterministic Polynomial). Εάν ένα πρόβλημα είναι σε NP και κάθε άλλο πρόβλημα στο NP μπορεί να αναχθεί σε αυτό το πρόβλημα σε πολυωνυμικό χρόνο, το πρόβλημα λέγεται ότι είναι NP-πλήρης. Τα NP-πλήρη προβλήματα είναι τα πιο δύσκολα προβλήματα στο NP.

Το κυριότερο εργαλείο για να αποδείξουμε ότι ένα πρόβλημα ανήκει σε μία κλάση πολυπλοκότητας είναι η αναγωγή. Η ιδέα της αναγωγής αναφέρεται στην μετατροπή του στιγμιότυπου π_1 ενός προβλήματος Π_1 σε ένα στιγμιότυπο π_2 του προβλήματος Π_2 με τέτοιο τρόπο, ώστε η λύση στο π_2 να δίνει την λύση και στο π_1.

Το SAT είναι πρόβλημα που αποδείχθηκε ότι είναι NP-πλήρες [14]. Αυτό σημαίνει ότι όλα τα προβλήματα στην τάξη πολυπλοκότητας NP, που περιλαμβάνει ένα ευρύ φάσμα φυσικών προβλημάτων απόφασης και βελτιστοποίησης, είναι εξίσου δύσκολα να λυθούν όπως το SAT. Δεν υπάρχει γνωστός αλγόριθμος που να επιλύει σε πολυωνυμικό χρόνο κάθε πρόβλημα SAT και γενικά πιστεύεται ότι δεν υπάρχει τέτοιος αλγόριθμος. Αλλά αυτή η εικασία δεν έχει αποδειχθεί μαθηματικά και η επίλυση του ζητήματος αν το SAT έχει έναν αλγόριθμο πολυωνυμικού χρόνου είναι ισοδύναμο με το πρόβλημα P ίσο με NP, το οποίο είναι ένα διάσημο ανοικτό πρόβλημα στη θεωρία της πληροφορικής. Εντούτοις, από το 2007, οι ευρετικοί αλγόριθμοι SAT μπορούν να επιλύσουν προβλήματα που αφορούν δεκάδες χιλιάδες

μεταβλητές και τύπους που αποτελούνται από εκατομμύρια σύμβολα , που επαρκούν για πολλά πρακτικά προβλήματα SAT από π.χ. την τεχνητή νοημοσύνη, τον σχεδιασμό κυκλώματος , και αυτόματη απόδειξη θεώρημα [15].

5. Εισαγωγή στα OR-Tools

Το **Google Developers** [16]είναι ο ιστότοπος της Google για εργαλεία ανάπτυξης λογισμικού, διεπαφές προγραμματισμού εφαρμογών (API) και τεχνικούς πόρους. Ο ιστότοπος περιέχει τεκμηρίωση σχετικά με τη χρήση εργαλείων προγραμματισμού Google και API, συμπεριλαμβανομένων ομάδων συζήτησης και ιστολογίων για προγραμματιστές που χρησιμοποιούν τα προϊόντα προγραμματιστών της Google. Υπάρχουν API που προσφέρονται για σχεδόν όλα τα δημοφιλή καταναλωτικά προϊόντα της Google, όπως οι Χάρτες Google [17], το YouTube [18],τα Google Apps [19] και άλλα. Ο ιστότοπος διαθέτει επίσης μια ποικιλία προϊόντων για προγραμματιστές και εργαλεία ειδικά σχεδιασμένα για προγραμματιστές. Το Google App Engine [20] είναι μια υπηρεσία φιλοξενίας (hosting service) για εφαρμογές ιστού. Το hosting service παρέχει στους χρήστες τη δυνατότητα ελέγχου της έκδοσης κώδικα ανοιχτού κώδικα. Το Google Web Toolkit (GWT) [21] επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές Ajax [22]στη γλώσσα προγραμματισμού Java [23].

Τα Google OR Tools παρέχουν συναρτήσεις περιτύλιξης (wrappers) [24]για εργαλεία έρευνας, όπως βελτιστοποίηση και επίλυση περιορισμών. Μια συνάρτηση περιτύλιξης είναι μια υπορουτίνα σε μια βιβλιοθήκη λογισμικού ή σε ένα πρόγραμμα υπολογιστή, ο κύριος σκοπός του οποίου είναι να καλέσει μια δεύτερη υπορουτίνα ή μια κλήση συστήματος με λίγο ή καθόλου επιπλέον υπολογισμό. Οι λειτουργίες περιτύλιξης χρησιμοποιούνται για να διευκολύνουν τα προγράμματα ηλεκτρονικού υπολογιστή γραφής. Αφορούν δηλαδή, ένα τύπο λογισμικού ανοιχτού κώδικα για βελτιστοποίηση, συντονισμό και αντιμετώπιση των πιο δύσκολων προβλημάτων στον τομέα της δρομολόγησης οχημάτων, ροών, ακέραιων και γραμμικών προγραμματισμών και προγραμματισμού με περιορισμούς [24].

Το OR-Tools παρέχει δύο λογισμικά επίλυσης του προβλήματος του προγραμματισμού με περιορισμούς τα οποία είναι : [25]

1. Ο CP-SAT solver
2. Ο original CP solver

Ο CP-SAT [26] είναι μια βιβλιοθήκη η οποία προσφέρει υλοποιήσεις σε διαφορετικές γλώσσες χρησιμοποιώντας την βασική υλοποίηση η οποία είναι γραμμένη σε C++. Η αρχική βιβλιοθήκη αποτελείται από 2 βασικά αρχεία linear_solver.h και linear_solver.cpp. Συγκεκριμένα,

- linear_solver.h: Το σημείο εισόδου για την συνάρτηση περιτύλιξης (wrapper) που παρέχει μια απλή και ενιαία διεπαφή σε διάφορα λογισμικά επίλυσης γραμμικού προγραμματισμού ή μικτού ακεραίου προγραμματισμού.

- linear_solver.cpp: Ο κώδικας C ++ της γραμμικής συνάρτησης περιτύλιξης του solver που είναι κοινό σε όλους τους solver που είναι προσβάσιμοι από την συνάρτηση περιτύλιξης (wrapper).

Είναι τεχνολογικά ανώτερος και πιο κατάλληλος για προβλήματα τα οποία είναι εγγενώς Boolean από τον αρχικό CP και θα πρέπει να προτιμάται σε όλες σχεδόν τις περιπτώσεις. Για αυτό τον λόγο και στην περίπτωση της δημιουργίας του ωρολόγιού προγράμματος χρησιμοποιήθηκε ο πρώτος .

6. Λειτουργία συστήματος

6.1. Θεωρητική αναπαράσταση του προβλήματος

Ένα παράδειγμα που με βοήθησε να ξεκινήσω για να μελετηθεί εκτενώς το πρόβλημα δημιουργίας του ωρολογίου προγράμματος σε αρχική μορφή παρουσιάζεται στην σελίδα τον Or-Tools όπου δίνεται ένα παράδειγμα Employee Scheduling, όπου οι οργανισμοί των οποίων οι εργαζόμενοι εργάζονται πολλαπλές βάρδιες και πρέπει να προγραμματίζουν επαρκείς εργαζόμενους για κάθε καθημερινή βάρδια. Τυπικά, τα προγράμματα θα έχουν περιορισμούς, όπως "κανένας υπάλληλος δεν θα πρέπει να εργάζεται δύο βάρδιες στη σειρά".

Στο συγκεκριμένο παράδειγμα [27], ένας νοσοκομειακός επόπτης πρέπει να δημιουργήσει ένα πρόγραμμα για τέσσερις νοσηλευτές σε μια τριήμερη περίοδο, υπό τις ακόλουθες προϋποθέσεις:

- Κάθε μέρα χωρίζεται σε τρεις βάρδιες 8 ωρών.
- Κάθε μέρα, κάθε βάρδια εκχωρείται σε μία μόνο νοσοκόμα και καμία νοσοκόμα δεν εργάζεται πάνω από μία βάρδια.
- Κάθε νοσοκόμα έχει αναλάβει τουλάχιστον δύο βάρδιες κατά τη διάρκεια της τριήμερης περιόδου.

Το πρόβλημα χρονοδιαγράμματος περιλαμβάνει πληθώρα μεταβλητών, με μια ομάδα p καθηγητών, μια ομάδα c μαθημάτων και μια ομάδα r αιθουσών εξέτασης. Ο κάθε καθηγητής εξετάζει ένα συγκεκριμένο μάθημα της ομάδας των μαθημάτων του, το οποίο έχει αναλάβει. Κάθε μάθημα θα πρέπει να τοποθετηθεί σε μια εκ των d ημερών που διαρκεί η εξεταστική και σε αυτήν την μέρα να επιλεχθεί καταλλήλως το χρονικό διάστημα εξέτασης από της s διαθέσιμες 3ώρες περιόδους που αναλογούν σε

κάθε ημέρα. Σε κάθε τέτοια ημέρα και ώρα εξέτασης θα πρέπει να ανατεθεί επίσης και η αίθουσα εξέτασης η οποία εξαρτάται από διάφορους παράγοντες οι οποίοι είναι μη σταθεροί. Τέτοιοι παράγοντες είναι ο αριθμός των ατόμων που πρόκειται να εξεταστούν καθώς και η μέρα και ώρα εξέτασης. Ο βασικός στόχος του προβλήματος είναι η μεγιστοποίηση ικανοποίησης προτιμήσεων των καθηγητών σχετικά με την ημέρα που θα εξεταστεί κάποιο τους μάθημα. Στον δικό μας αλγόριθμο επίλυσης, ο κάθε καθηγητής επιλέγει ένα εύρος ημερών με βάση την διαθεσιμότητά του, ωστόσο ο αλγόριθμος θα εξετάσει κάθε μάθημα ξεχωριστά για κάθε μία ημέρα του επιθυμητού διαστήματος.

Τα προβλήματα χρονοδιαγράμματος μοντελοποιούνται συνήθως μέσω της διατύπωσης ακέραιου γραμμικού προγραμματισμού [28]. Παρακάτω παρουσιάζουμε την αναπαράσταση του μαθηματικού μοντέλου που προκύπτει από το πρόβλημά μας. Ο συμβολισμός και οι παράμετροι που χρησιμοποιούνται προκύπτουν ως εξής:

- p – Ο αριθμός των καθηγητών
- c – Ο αριθμός των μαθημάτων
- r – Ο αριθμός των αιθουσών εξέτασης
- t – Ο αριθμός των ημερών της εξεταστικής
- h – Ο αριθμός των πιθανών ωρών εξέτασης σε μία ημέρα
- m – Ο αριθμός των εξαμήνων που συμμετέχουν στην εξεταστική
- d – Ο δείκτης των εργάσιμων ημερών {1,2,….,t}
- i – Ο δείκτης των καθηγητών {1, 2, …., p}
- n – Ο δείκτης των μαθημάτων {1, 2, …., c}
- k – Ο δείκτης των αιθουσών {1, 2,…., r}
- s – Ο δείκτης των πιθανών ωρών εξέτασης {1, 2,…, h}
- e – Ο δείκτης των εξαμήνων {1, 2, ….., m}

6.2. Μεταβλητή απόφασης:

Σε μια γενικευμένη μεταβλητή όπου τίποτα δεν είναι προκαθορισμένο, η μεταβλητή απόφασης θα ήταν ως εξής:

Xi,n,d,s,k = 1 εάν ο καθηγητής i μπορεί να εξετάσει το μάθημα n την ημέρα d στην βάρδια s και στην αίθουσα k, 0 διαφορετικά

Ωστόσο, τα μαθήματα και οι αίθουσες εξέτασης είναι παράμετροι που καθορίζονται πριν την επίλυση του προβλήματος. Πιο συγκεκριμένα, κάθε μάθημα έχει ήδη ανατεθεί σε κάποιον καθηγητή που θα το εξετάσει από την αρχή του εξαμήνου, ενώ η αίθουσα εξέτασης ενός μαθήματος προκύπτει από διάφορες παραμέτρους, όπως η

ημέρα και η ώρα εξέτασης, ο αριθμός των φοιτητών που εξετάζονται αλλά και η διαθεσιμότητά της αίθουσας. Συνεπώς, οι μεταβλητές του καθηγητή και της αίθουσας εξέτασης είναι γνωστές εκ των προτέρων και για αυτό στο πρόβλημά μας τις αγνοούμε. Έτσι λόγω των παραπάνω η μεταβλητή απόφασης όπως προκύπτει στο δικό μας πρόβλημα είναι:

Xn,d,s = 1 εάν το μάθημα n θα εξεταστεί την ημέρα d και στην βάρδια s, διαφορετικά 0.

6.3. Περιορισμοί Προβλήματος

Για την σωστή επίλυση του προβλήματος θα πρέπει να συλλέξουμε όλους τους περιορισμούς που προκύπτουν από τις ανάγκες τις σχολής και να τις τροποποιήσουμε ώστε να μπορούν να συμπεριληφθούν στο μοντέλο μας. Παρακάτω αναφέρονται οι διαφορετικοί περιορισμοί και η ενσωμάτωσή τους στο μοντέλο με βάση τις ήδη αναφερθέντες μεταβλητές.

- **Δεν μπορεί να υπάρχει εξέταση Σάββατο ή Κυριακή ή σε αργία**

Συνεπώς το άθροισμα όλων των πιθανών τιμών της μεταβλητής απόφασης για όλες τις βάρδιες σε ημέρα Σαββάτου, Κυριακής ή αργίας θα πρέπει να είναι <= 0 για κάθε μάθημα. Μιας και η περίοδος εξεταστικής είναι πάντα Ιούνιο, Γενάρη έως και Φλεβάρη, ή Σεπτέμβρη, οι σταθερές αργίες που εξετάζουμε είναι η 30η Ιανουαρίου (Τριών Ιεραρχών) και η 8η Ιουνίου (Αγίου Πνεύματος).

<=0 d= «Σάββατο» ή d= «Κυριακή» ή d= «Αργία» , n

- **Για κάθε φιξαρισμένο μάθημα n και για κάθε φιξαρισμένη ημέρα d θα πρέπει το μάθημα να εξεταστεί σε πολύ ένα χρονικό διάστημα s**

Δηλαδή το άθροισμα όλων των δυνατών τιμών της μεταβλητών απόφασης όλων των πιθανών ωρών εξέτασης θα πρέπει να είναι <=1 για κάθε φιξαρισμένο μάθημα n και για κάθε φιξαρισμένη ημέρα d της εξεταστικής.

<=1 d, n

- **Κάθε φιξαρισμένο 3ώρο εξέτασης και φιξαρισμένη ημέρα d της εξεταστικής θα πρέπει να εξετάζεται το περισσότερο ένα μάθημα.**

Επομένως το άθροισμα όλων των πιθανών τιμών της μεταβλητής απόφασης όλων των μαθημάτων θα πρέπει να είναι <=1 για κάθε φιξαρισμένο 3ώρο s εξέτασης και κάθε ημέρα της εξεταστικής.

<=1 d, s

- **Κάθε φιξαρισμένη ημέρα d της εξεταστικής θα πρέπει να εξετάζεται το περισσότερο ένα μάθημα που ανήκει σε ένα φιξαρισμένο εξάμηνο υποχρεωτικών μαθημάτων.**

Συνεπώς το άθροισμα όλων των μαθημάτων ενός φιξαρισμένου εξαμήνου, όπου n΄ ένα τέτοιο μάθημα και N το σύνολό τους, σε όλες τα δυνατά 3ώρα εξέτασης μιας ημέρας πρέπει να είναι <=1 για φιξαρισμένη ημέρα d εξέτασης.

$$\sum_{n'=1}^{N} X_{n',d,s} <=1 \ \forall \ d$$

Το αποτέλεσμα προκύπτει από το μοντέλο :

$$\text{Max Z} = \sum_{s=1}^{h}\sum_{d=1}^{t}\sum_{n=1}^{c} X_{n,d,s}$$

στο οποίο θέλουμε να μεγιστοποιήσουμε τις τιμές της μεταβλητής απόφασης, δηλαδή να ικανοποιηθούν όσο το δυνατόν περισσότεροι περιορισμοί για όσο το δυνατόν περισσότερα μαθήματα.

Εντούτοις, ο κάθε καθηγητής επιλέγει για κάθε μάθημά του ένα εύρος από προτεινόμενες ημερομηνίες εξέτασης. Αυτό μεταφράζεται ότι για κάθε μάθημά του ο εκάστοτε καθηγητής επιλέγει να εξεταστεί μια από αυτές τις ημέρες. Για τον λόγο αυτό θα πρέπει στο πρόβλημα να συμπεριληφθεί μια νέα μεταβλητή απόφασης η οποία ορίζει για κάθε μάθημα εάν επιλέχθηκε από τον καθηγητή να εξεταστεί σε μια συγκεκριμένη μέρα. Η ώρα εξέτασης σε αυτό το στάδιο δεν είναι επιλογή του καθηγητή, συνεπώς όλες οι ώρες είναι αποδεκτές και έτσι θα επιλεχθεί τελικά αυτή η οποία θα βελτιστοποιήσει την λύση. Η νέα μεταβλητή απόφασης που προκύπτει είναι :

$P_{n,d,s}$ = 1 εάν το μάθημα n προτιμήθηκε να εξεταστεί την ημέρα d και στην βάρδια s, διαφορετικά 0.

Με βάση τις προτιμήσεις του κάθε καθηγητή το μοντέλο διαφοροποιείται και πλέον προκύπτει ως εξής :

$$\text{Max } Z = \sum_{s=1}^{h} \sum_{d=1}^{t} \sum_{n=1}^{c} P_{n,d,s^*} X_{n,d,s}$$

6.4. Πρακτική εφαρμογή του προβλήματος

Το πρόγραμμα μπορεί να τρέξει σε Linux και σε Windows χωρίς κανένα πρόβλημα, ενώ στο σημείο αυτό θα μελετηθούν οι ενέργειες που πρέπει να γινούν με στόχο να βγάλει το τελικό αποτέλεσμα μέσα από ένα τυχαίο παράδειγμα.

6.4.1. Οδηγίες για την Εγκατάσταση του προγράμματος:

- Για να μπορεί να τρέξει το πρόγραμμα θα πρέπει να εγκατασταθεί το Pycharm.

Το PyCharm είναι ένα εξειδικευμένο Python Integrated Development Environment (IDE) που παρέχει ένα ευρύ φάσμα βασικών εργαλείων για τους προγραμματιστές της Python, οι οποίοι είναι στενά ενσωματωμένοι μαζί για να δημιουργήσουν ένα βολικό περιβάλλον για την παραγωγική ανάπτυξη της Python, του Ιστού και των δεδομένων.

Οδηγίες Εγκατάστασης [29]**:**

- Αντιγράψτε το pycharm-2019.3.2.tar.gz στην επιθυμητή θέση εγκατάστασης

  (βεβαιωθείτε ότι έχετε δικαιώματα rw για αυτόν τον κατάλογο)

- Αποσυμπιέστε το αρχείο pycharm-2019.3.2.tar.gz σε έναν κενό κατάλογο χρησιμοποιώντας την ακόλουθη εντολή:

tar -xzf pycharm-2019.3.2.tar.gz

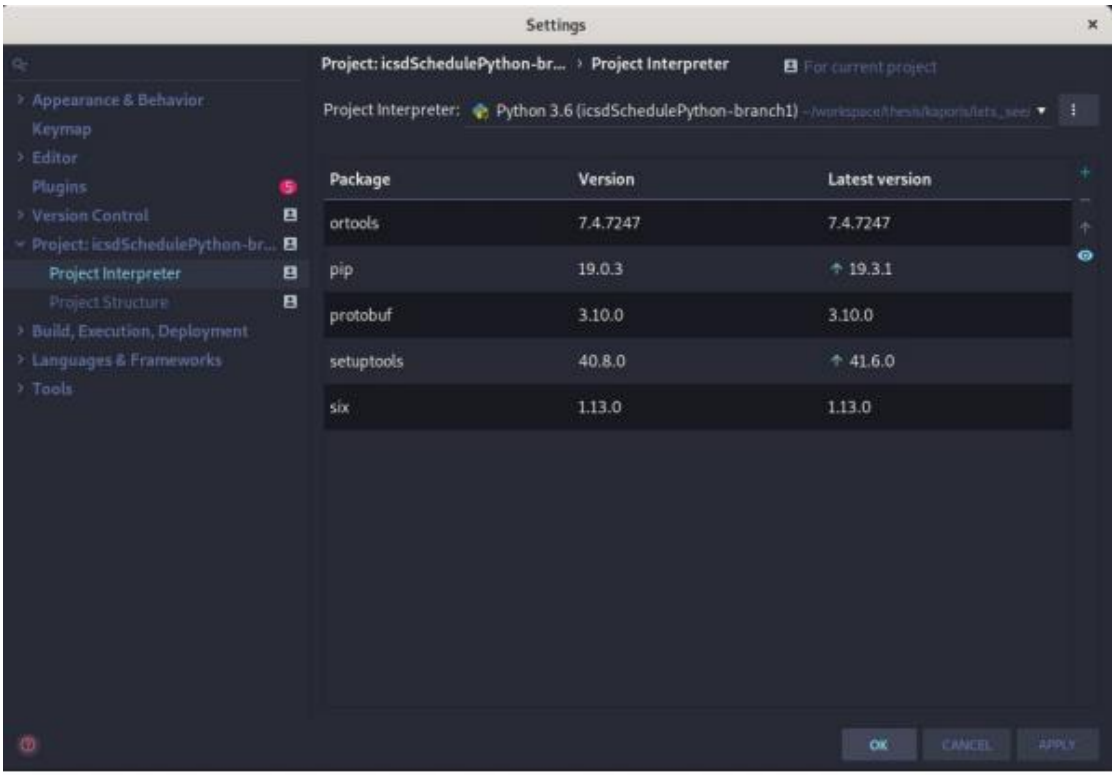
ΔΕΝ ΠΡΕΠΕΙ να εξαχθεί σε ήδη υπάρχων. Ο φάκελος προορισμού πρέπει να είναι κενός

- Εκτελέστε **pycharm.sh** από τον υποκατάλογο bin

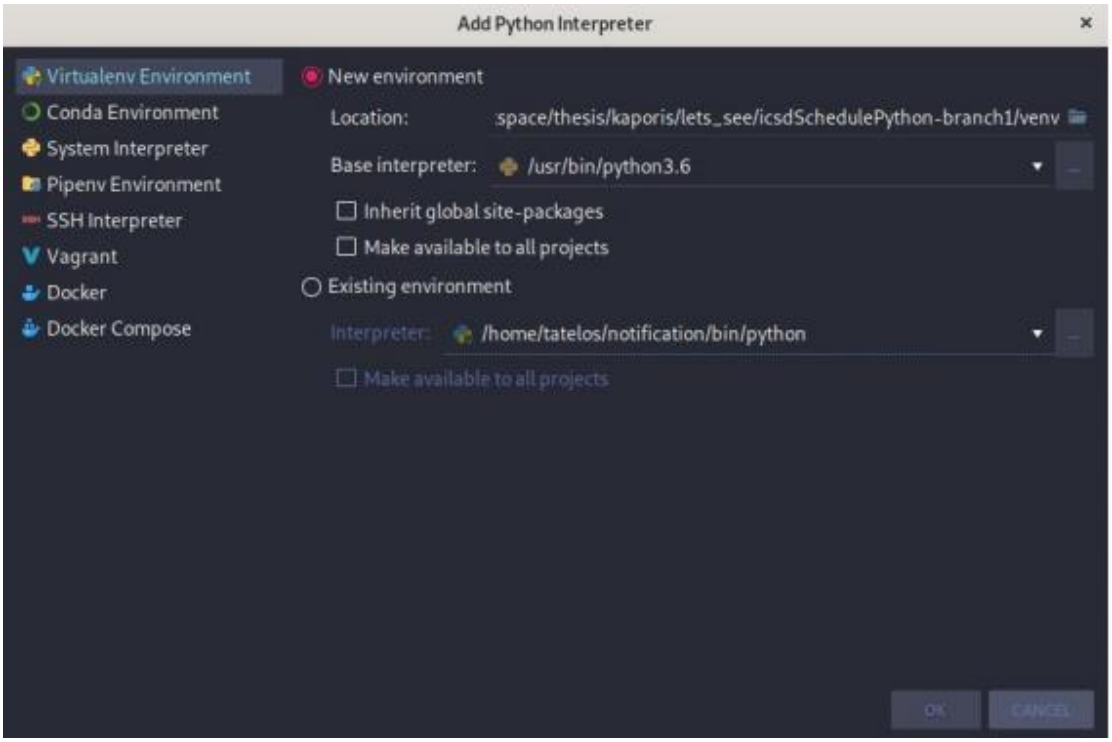
Το PyCharm διατίθεται σε τρεις εκδόσεις: Επαγγελματική, Κοινοτική και Εκπαιδευτική (Edu). Οι εκδόσεις της Κοινότητας και του Edu είναι έργα ανοιχτού κώδικα και είναι δωρεάν, αλλά έχουν λιγότερα χαρακτηριστικά. Το PyCharm λειτουργεί σε Windows, macOS και Linux.

- Στην συνέχεια θα πρέπει να ανοιχθεί το Pycharm και στην πάνω αριστερή γωνία να γίνει η επιλογή **File → Open → επιλογή του αρχείου (**icsdSchedulePython) σύμφωνα με την τοποθεσία που βρίσκεται στο υπολογιστή του κάθε χρήστη.

- Αφού ανοίξει το project (icsdSchedulePython) απαραίτητο είναι να δημιουργηθεί ένα νέο virtual environment. Για αυτό **File → Settings → Project → Project Interpreter.**



- Στην συνέχεια κλικ στις τρεις τελείες δίπλα από το Project Interpreter ώστε να γίνει η δημιουργία του νέου περιβάλλοντος.



- Επιλογή του **New environment και OK**.

- Στην συνέχεια πρέπει να γίνει εγκατάσταση του ortools με την εντολή:

```
pip install ortools
```

- Τέλος θα **γίνει εκτέλεση της main2.py**

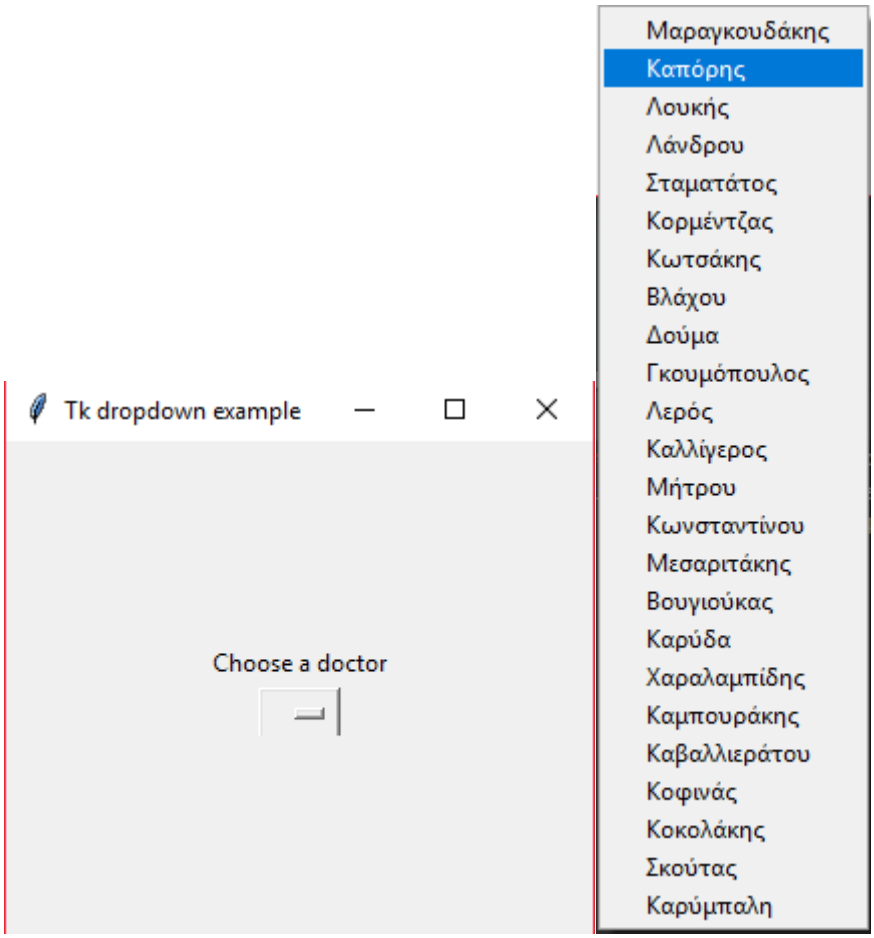
6.4.2. Οδηγίες για την εκτέλεση του προγράμματος:

1) Αρχικά, το πρόγραμμα ζητάει από τον χρήστη να εισάγει την πρώτη ημέρα της εξεταστικής

2) `Dwse tin prwti imera tis exetastikis :` → `19/01/18` (date format : d/m/y)

3) Αφού εισαχθεί η σωστή πρώτη ημέρα της εξεταστικής συνεχίζει με το πλήθος των ημερών που έχουμε διαθέσιμο προς εξέταση,
`Dwse to plithos imerwn tis exetastikis` → `30`

4) Στη συνέχεια πατώντας στο κουμπί στο αναδυόμενο παράθυρο γίνεται η επιλογή του καθηγητή που θέλουμε να εισάγουμε στο σύστημα.



5) Έπειτα ζητάει την 1η ημέρα όπου ο καθηγητής θα είναι διαθέσιμος και θα μπορεί να παρευρίσκεται για την εξέταση του μαθήματός του.

```
Dwse tin prwti imera pou diatihetai o kathigitis :
```
→ `20/01/18`

Καθώς και την τελευταία μέρα που θα βρίσκεται στο νησί

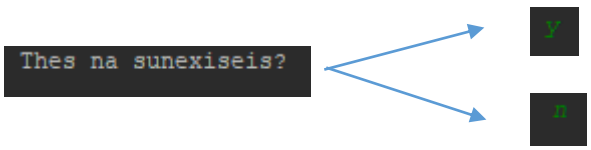
```
Dwse tin teleutaia imera pou diatihetai o kathigitis :
```
→ `27/01/18`

6) Εφόσον έχουν τοποθετηθεί όλες οι ημερομηνίες ορθά θα ζητήσει των αριθμό των ατόμων που περιμένει ο καθηγητής για κάθε μάθημα.

Κάθε καθηγητής έχει αντιστοιχηθεί με τα μαθήματα του το οποίο το πρόγραμμα διαβάζει από το αρχείο courses.txt

```
Dwse atoma gia to mathima Αλγόριθμοι & Πολυπλοκότητα 85
Dwse atoma gia to mathima Θεωρία Υπολογισμού 48
Dwse atoma gia to mathima Θεωρία Παιγνίων 95
Dwse atoma gia to mathima Αλγόριθμοι και Συνδυαστική Βελτιστοποίηση 24
```

7) Σε αυτό το σημείο το πρόγραμμα έχει λάβει υπόψη τον ένα καθηγητή(συγκεκριμένο παράδειγμα: *Καπόρης*) και ρωτάει εάν ο χρήστης επιθυμεί να συνεχίσει.



8) Τέλος εάν διαλέξουμε να συνεχίσουμε θα επαναληφθεί ακριβώς η ίδια διαδικασία και για τον επόμενο καθηγητή ενώ εάν τερματιστεί νωρίτερα θα ταξινομήσει τους εναπομείναντα καθηγητές αυτόματα και θα τυπώσει το αποτέλεσμα τόσο στο πρόγραμμα όσο και σε excel το οποίο αποθηκεύεται αυτόματα με την ονομασία output.xls στο γονικό φάκελο του project.

```
Day Saturday 20/01
Day Sunday 21/01
Day Monday 22/01
Course Αλγόριθμοι & Πολυπλοκότητα  is examed at 12:00-15:00 (requested). Aithousa  No.1, No.3  examino  4
Course Θεωρία Υπολογισμού  is examed at 15:00-18:00 (requested). Aithousa  No.1, No.3  examino  5
Course Θεωρία Πληροφορίας  is examed at 09:00-12:00 (not requested).Aithousa  Κων. Σοφούλη  examino  8
Course Δίκτυα Ευρείας Ζώνης  is examed at 18:00-21:00 (not requested).Aithousa  Κων. Σοφούλη  examino  9
Day Tuesday 23/01
Course Υπολογιστική Λογική & Λογικός Προγραμματισμός  is examed at 12:00-15:00 (not requested).Aithousa  Κων. Σοφούλη  examino  7
Course Θεωρία Παιγνίων  is examed at 15:00-18:00 (requested). Aithousa  No.1, No.3  examino  8
Course Υπολογιστική Όραση  is examed at 09:00-12:00 (not requested).Aithousa  Κων. Σοφούλη  examino  9
Day Wednesday 24/01
Course Ρομποτικός Έλεγχος  is examed at 15:00-18:00 (not requested).Aithousa  Κων. Σοφούλη  examino  7
Course Αποθήκες Δεδομένων και εξόρυξης  is examed at 12:00-15:00 (not requested).Aithousa  Κων. Σοφούλη  examino  8
Course Μηχανική Γνώσης και Συστήματα Γνώσης  is examed at 09:00-12:00 (not requested).Aithousa  Κων. Σοφούλη  examino  9
Day Thursday 25/01
Course Ασφάλεια Πληροφοριακών & Επικοινωνιακών Συστημάτων  is examed at 18:00-21:00 (not requested).Aithousa  No.1, No.3  examino  6
Course Πρωτόκολλα και Αρχιτεκτονικές Διαδικτύου  is examed at 15:00-18:00 (not requested).Aithousa  Κων. Σοφούλη  examino  7
Course Ανάκτηση Πληροφορίας  is examed at 12:00-15:00 (not requested).Aithousa  Κων. Σοφούλη  examino  8
Course Σχεδιασμός και Ανάπτυξη Εφαρμογών Κινητού Υπολογισμού  is examed at 09:00-12:00 (not requested).Aithousa  Κων. Σοφούλη  examino  9
Day Friday 26/01
Course Λειτουργία των Επιχειρήσεων & Πληροφοριακά Συστήματα  is examed at 15:00-18:00 (not requested).Aithousa  No.1, No.3  examino  5
Course Τεχνολογίες Δικτύων και Νέφους  is examed at 12:00-15:00 (not requested).Aithousa  Κων. Σοφούλη  examino  8
Course Αλγόριθμοι και Συνδυαστική Βελτιστοποίηση  is examed at 18:00-21:00 (requested). Aithousa  Κων. Σοφούλη  examino  9
Day Saturday 27/01
Day Sunday 28/01
```

7. Επίλυση προβλήματος με χρήση CP-SAT Solver (Google OR-Tools)

Για την αναπαράσταση του κάθε μαθήματος χρειάζεται μια κλάση στην οποία θα αποθηκεύονται όλα τα στοιχεία του, όπως το όνομά του, το εξάμηνο και το έτος στο οποίο ανήκει το μάθημα, εάν είναι υποχρεωτικό ή όχι καθώς και ο καθηγητής που το διδάσκει.

```
class Course:

    def __init__(self, coursename, semester, req, doctor):
        self.courseName = coursename
        self.semester = int(semester) - 1
        if req == "1":
            self.required = True
            self.room = 2
        else:
            self.required = False
            self.room = 1
        self.year = (int(semester) - 1) // 2
        self.doctor = doctor

    def setParticipants(self, parts):
        self.participants = parts

    def setShift(self, examShift):
        self.examShift = examShift

    def print(self):
        return self.courseName + " " + self.semester + " " +
str(self.required) + " " + str(self.doctor)
```

Για την δημιουργία μοντέλου θα χρειαστούμε μεταβλητή λογικού τύπου (boolean), για την αναπαράσταση της μεταβλητής απόφασής μας η οποία αποτελείται από τρεις διαφορετικές μεταβλητές, το μάθημα (c), την ημέρα εξέτασης (d) καθώς και την ώρα εξέτασης (s). Ως λογική μεταβλητή θα δέχεται τιμές 0 (false) ή 1 (true), ανάλογα με το αν το μάθημα c, εξετάζεται την ημέρα d και ώρα s ή όχι. Ωστόσο καθώς χρειαζόμαστε περισσότερες πληροφορίες για την εξέταση του μαθήματος, δημιουργούμε μια πλειάδα (tuple) με τρεις παραμέτρους, το time, το semester και το room. Η πρώτη μεταβλητή, διατηρεί την τελική τιμή της λογικής μεταβλητής μας, όπως προαναφέρθηκε παραπάνω, με 0 ή 1. Στις επόμενες δύο διατηρείται το σε ποιο εξάμηνο ανήκει το μάθημα και σε ποια αίθουσα θα εξεταστεί. Η μεταβλητή semester θα αρχικοποιηθεί στην συνέχεια με το εξάμηνο του κάθε μαθήματος, έτσι ώστε να μπορούμε να αποκλείσουμε στην συνέχεια να αποκλείσουμε το ενδεχόμενο δύο μαθήματα του ίδιου εξαμήνου να εξετάζονται την ίδια ημέρα. Η μεταβλητή room δεν

χρειάζεται σε κάποιον περιορισμό, είναι όμως μέρος του ζητήματος και δημιουργείται μηχανικά χωρίς την χρήση του μοντέλου.

```
model = cp_model.CpModel()

    course_type = collections.namedtuple('course_name', 'time
semester room')

    for c in all_courses:
        for d in all_days:
            for s in all_shifts:
                #η μεταβλητή απόφασης μεταφέρεται σε μια μεταβλητή
                #time
                time = model.NewBoolVar('shift_n%id%is%i' % (c, d,
s))

                #ο πίνακας shifts που διατηρεί την μεταβλητή απόφασης
                #αλλά και το εξάμηνο του μαθήματος και την αίθουσα
                #εξέτασης
                shifts[c, d, s] = course_type(time,
semester=courses[c].semester, room=courses[c].room)
```

Για να επιλέξουμε την αίθουσα που θα εξεταστεί το μάθημα ζητάμε από τον χρήστη του συστήματος να δώσει τον αριθμό των αναμενόμενων εξεταζόμενων φοιτητών.

```
room = ["Νέο Σχολικό", "Κων. Σοφούλη", "No.1, No.3"]

# an to mathima einai upoxrewtiko i an to xrwstane polla atoma 8a
prepei na exetastei sto mesaio
if courses[mathima].required or atoma > 85:
      courses[mathima].room = 2
elif atoma > 20:
      courses[mathima].room = 1
else:
      courses[mathima].room = 0
```

Κατά την έναρξη του προγράμματος ο χρήστης θα πρέπει να εισάγει το διάστημα στο οποίο επιθυμεί να εξεταστούν τα μαθήματά του. Το διάστημα αυτό μεταφράζεται σε δυο μεταβλητές όπου «firstDay» είναι η πρώτη μέρα και «lastDay» η τελευταία. Με βάση αυτό το διάστημα θα δημιουργήσουμε τον πίνακα «preferedDays» ο οποίος είναι αντίστοιχος με τον πίνακα shifts μόνο που αντί για τις πραγματική ώρα και μέρα εξέτασης του μαθήματος, περιέχει 1 στις ώρες και μέρες που για τα μαθήματα του κάποιος καθηγητής επιθυμεί να εξεταστεί.

Στο συγκεκριμένο παράδειγμα για ένα συγκεκριμένο διάστημα ημερών [firstDay, lastDay], στον πίνακα preferedDays τοποθετείται «1» στα αντίστοιχα μαθήματα του καθηγητή κάθε ημέρα και ώρα του συγκεκριμένου έυρους. Η παρακάτω συνάρτηση ελέγχει εάν κάποια ημέρα πέφτει Σάββατο ή Κυριακή ώστε να την εξαιρέσει από τον έλεγχο. Επίσης αν κάποιος καθηγητής είναι Αθηναίος και δεν κατοικεί μόνιμα στην Σάμο επιλέγουμε στην τελευταία μέρα του διαστήματος να εξαιρεθούν οι 2 τελευταίες ώρες εξέτασης και να μείνουν μόνο οι 2 πρωινές διαθέσιμες, ώστε ο καθηγητής να μπορεί να προλάβει κάποια πιθανή πτήση. Σαν τελευταία ενέργεια η συνάρτηση καλεί την fillDay, η οποία θα εισάγει στον πίνακα «prefferedDays» 1, μόνο τις ώρες της ημέρας τις οποίες είναι εφικτή κάποια εξέταση ανάλογα με το πλήθος των φοιτητών και την αίθουσα που έχει επιλεχτεί.

```
# sunartisi gemismatos tou pinaka preferedDays
# opou gia kathe imera tou mathimatos kai kathe
# vardia gemizoume kaloume antistoixa tin fillDay
def makerange(firstDay, lastDay, mathima, room, doctor, startDay):
    done = False
    no_shifts = num_shifts

    for d in range(firstDay, lastDay + 1):
        check = (startDay +
datetime.timedelta(days=d)).strftime("%A")
        if (check == "Saturday" or check == "Sunday"):
            print("not")
            for ns in range(0, 4):
                preferedDays[mathima][d][ns] = 0
            continue
        # tin teleutaia imera afise na epilextoun mono 2 vardies gia
tous athinaious
        if doctor in athinaioi and d == day2:
            no_shifts = 2
        for sh in range(no_shifts):
            fillDay(mathima, d, sh, room, check)

    return done
```

Για παράδειγμα, ένα μάθημα με πάνω από 100 πιθανούς εξεταζόμενους φοιτητές θα πρέπει να εξεταστεί στην αίθουσα «Σχολικό» στην οποία τις Δευτέρες είναι διαθέσιμη μόνο στην $1^{η}$ και στην $3^{η}$ βάρδια, ενώ τις υπόλοιπες εξετάζεται κάποια άλλη σχολή.

```
def fillDay(mathima, d, sh, room, day):
    if room == 0:
        if sh == 2:
            preferedDays[mathima][d][sh] = 1
            return True
    elif room == 1:
        if day == "Monday":
            if sh == 0 or sh == 3:
                preferedDays[mathima][d][sh] = 1
                return True
        elif day == "Tuesday":
            if sh == 0 or sh == 1 or sh == 3:
                preferedDays[mathima][d][sh] = 1
            return True
```

Αφού έχουν ολοκληρωθεί όλες οι απαραίτητες ενέργειες για την προετοιμασία του μοντέλου, το μοντέλο δημιουργείται και επιλύεται με βάση τις προτιμήσεις των καθηγητών αλλά και τους παρακάτω περιορισμούς:

Θέλουμε από τα αποτελέσματα να εξαιρεθούνε όσες ημέρες είναι Σάββατο ή Κυριακή, ώστε να μείνουν οι καθημερινές μόνο ως διαθέσιμες.

```
# Exclude days from the check if the day is Sunday or Saturday
    for d in all_days:
        check = (startDay +
datetime.timedelta(days=d)).strftime("%A")
        if check == "Saturday" or check == "Sunday":
            print(check)
            for c in all_courses:
                model.Add(sum(shifts[c, d, s].time for s in
all_shifts) <= 0)
```

Σε κάθε ώρα εξέτασης θα πρέπει να εξετάζεται το πολύ ένα μάθημα.

```
# Each shift is assigned to exactly one course in the schedule
period.
for d in all_days:
    for s in all_shifts:
         model.Add(sum(shifts[(c, d, s)].time for c in all_courses)
<= 1)
```

Κάθε ημέρα της εξεταστικής θα πρέπει να εξετάζονται μαθήματα με διαφορετικό εξάμηνο. Κάθε εξάμηνο θα πρέπει να εμφανίζεται το πολύ μια φορά σε μια ημέρα.

```
# Each day is scheduled to have unique semesters
for d in all_days:
   for sem in all_semester:
        model.Add(sum(
                shifts[c, d, s].time for c in all_courses for s in
all_shifts if shifts[c, d, s].semester == sem) <= 1)
```

Κάθε μάθημα θα πρέπει να εξετάζεται μια φορά σε κάθε εξεταστική περίοδο.

```
# Each course is scheduled at exactly one shift per exam period
for c in all_courses:
      for d in all_days:
            model.Add(sum(shifts[(c, d, s)].time for s in all_shifts)
<= 1)
```

Το ελάχιστο και το μέγιστο όπου κάθε μάθημα θα πρέπει να εξεταστεί. Σε μια εξεταστική Σεπτεμβρίου όλα τα μαθήματα θα πρέπει να εξετάζονται, ενώ σε κάποια εξεταστική Φεβρουαρίου ή Ιουνίου θα επιλέγεται για κάθε μάθημα ο μέγιστος αριθμός εμφανίσεων (0 ή 1) ανάλογα αν υπάρχει εμβόλιμη ή όχι.

```
# the minumum and maximum shifts that each course should be examined
within an exam period
# (if we have a June exam, the odd semester courses should be
excluded)
elaxistesWresExetasisAnaMathima = 1
megistesWresExetasisAnaMathima = 1
for c in all_courses:
      num_shifts_worked = sum(
            shifts[(c, d, s)].time for d in all_days for s in
all_shifts)
      model.Add(elaxistesWresExetasisAnaMathima <= num shifts worked)
      model.Add(num_shifts_worked <= megistesWresExetasisAnaMathima)
```

Τέλος, το μοντέλο θα πρέπει να μεγιστοποιεί τις τιμές του πίνακα λαμβάνοντας υπόψη τις προτιμήσεις των καθηγητών. Η παρακάτω συνάρτηση όπως προαναφέρθηκε σκοπεύει να επιλύσει το πρόβλημα στον μεγαλύτερο δυνατό βαθμό, χωρίς να σημαίνει πως οι λύσεις θα είναι πάντα απόλυτα αποδεκτές εφόσον οι προτιμήσεις των καθηγητών δεν το επιτρέπουν.

```
# Megistopoiisi twn pithanwn apotelesmatwn etsi wste na
ikanopoiithoun oso perissoteroi
# kathigites ginetai me vasi tis apaitiseis tous
model.Maximize(
      sum(preferedDays[c][d][s] * shifts[(c, d, s)].time for c in
all_courses
            for d in all_days for s in all_shifts))
```

8.Συμπεράσματα

Η τρέχουσα διπλωματική δημιούργησε και παρουσίασε ένα μοντέλο για το πρακτικό πρόβλημα προγραμματισμού των καθηγητών για το τμήμα του μηχανικών πληροφοριακών και επικοινωνιακών συστημάτων. Ο στόχος είναι να μεγιστοποιηθεί ο αριθμός των καθηγητών που έχουν ανατεθεί πλήρως. Μετά από ενδελεχή έλεγχο των αποτελεσμάτων παρατηρήθηκε ότι το πρόγραμμα είναι σε θέση να καλύψει όλες τις προτιμήσεις των καθηγητών χωρίς να καταπατάει τους περιορισμούς που του έχουν δοθεί.

Οι συνεισφορές σε αυτή τη διατριβή καθιστούν σαφή τον δρόμο για πολλές μελλοντικές εργασίες οι οποίες θα αφορούν το πανεπιστημιακό χρονοδιάγραμμα. Το χρονοδιάγραμμα καθηγητών και εξετάσεων είναι στενά συνδεδεμένα προβλήματα τα οποία, όταν εξεταστούν μαζί σε ένα πλήρως αυτοματοποιημένο σύστημα, θα παρέχουν μια ολοκληρωμένη λύση σε ένα ακαδημαϊκό ίδρυμα. Η ευκολία και η ευελιξία εφαρμογής και προσαρμογής που προσφέρονται από τον προγραμματισμό με περιορισμούς μας ενθαρρύνουν για παράδειγμα να εφαρμοστεί και να προσαρμοστεί μέσα από την ίδια προσέγγιση το πρόβλημα προγραμματισμού των σπουδαστών και χρονοδιαγράμματος μαθημάτων.

Μία καλή ιδέα ακόμα θα ήταν η δημιουργία ενός εξυπηρετητή όπου ο κάθε καθηγητής θα μπορεί να συνδέεται με τα στοιχεία του στον προσωπικό του λογαριασμό και εκεί θα βλέπει τα προσφερόμενα μαθήματα τα οποία έχει το τρέχων εξάμηνο και στην συνέχεια θα τοποθετεί τις ημερομηνίες που επιθυμεί να εξετάσει τα συγκεκριμένα μαθήματα. Στην συνέχεια και αφού έχουν εισέλθει όλοι οι καθηγητές στο σύστημα, δίνοντας τις επιθυμητές ημερομηνίες, θα καλεί τον συγκεκριμένο αλγόριθμο που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής και θα δίνει το κατάλληλο αποτέλεσμα σε ένα πιο οικείο γραφικό περιβάλλον. Με αυτό τον τρόπο η διαδικασία αυτοματοποιείται ακόμα περισσότερο.

9. Βιβλιογραφία

[1] W.-J. v. Hoeve, «Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference,» σε *CPAIOR 2018, Delft*, The Netherlands, June 26–29, 2018.

[2] W.-J. v. Hoeve, Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Delft, The Netherlands: Proceedings, CPAIOR 2018.

[3] S. &. M. M. Abdennadher, University timetabling using constraint handling rules, (1998).

[4] L. Y. L. P. C. &. T. C. Y. Shue, Constraint Programming Approach for a University Timetabling Decision Support System with Hard and Soft Constraints. In Opportunities and Challenges for Next-Generation Applied Intelligence (pp. 93-98), Springer, Berlin, Heidelberg., (2009).

[5] M. Gavanelli, University timetabling in ECLiPSe. ALP Newsletter, (2006).

[6] J. &. M. M. J. Joxan Jaffar, Constraint logic programming: A survey. The journal of logic programming, 19, 503-581., (1994). .

[7] Eclipse, «https://eclipseclp.org/,» eclipse. [Ηλεκτρονικό].

[8] G. &. A. N. &. S. R. &. A. R. Sadia, Graphical Simulation of N Queens Problem. International Journal of Computer Technology and Applications., 02, (2011).

[9] S. J. &. N. P. Russell, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited, 2016.

[10] H. Simonis, Sudoku as a constraint problem. In CP Workshop on modeling and reformulating Constraint Satisfaction Problems, (Vol. 12, pp. 13-27). Citeseer, 2005, October.

[11] K. Ghédira, Constraint satisfaction problems: csp formalisms and techniques., John Wiley & Sons., (2013). .

[12] Y. W. G. &. M. S. Vizel, Boolean satisfiability solvers and their applications in model checking., Proceedings of the IEEE, 103(11), 2021-2035., (2015).

[13] S. Cook, «The P versus NP problem. The millennium prize problems,» σε *P versus NP* , (2006), pp. 87-104..

[14] J. C. Ruben Gamboa, A Mechanical Proof of the Cook-Levin Theorem Proving in Higher Order Logics, Volume 3223, 2004.

[15] D. Babic, J. Bingham και A. J. Hu, B-Cubing: New Possibilities for Efficient SAT-Solving" (PDF). IEEE Transactions on Computers., (2006).

[16] google, «https://developers.google.com/,» google. [Ηλεκτρονικό].

[17] mapsGoogle, «https://developers.google.com/maps/documentation,» mapsGoogle. [Ηλεκτρονικό].

[18] developersYoutube, «https://developers.google.com/youtube/v3,» developersYoutube. [Ηλεκτρονικό].

[19] google, «https://developers.google.com/gsuite/aspects/apis,» google2. [Ηλεκτρονικό].

[20] D. Sanderson, Programming Google App Engine, O'Reilly Media, p. 536, ISBN 978-1449398262, (October 26, 2012).

[21] R. Dewsbury, Google Web Toolkit Applications. Prentice Hall., ISBN 978-0-321-50196-7., December 15, 2007.

[22] J. J. Garrett, "Ajax: A New Approach to Web Applications"., AdaptivePath.com. Archived from the original on 10 September 2015. Retrieved 19 June 2008., (18 February 2005).

[23] J. Gosling, B. Joy, G. Steele, G. Bracha και A. Buckley, (2014), (Java SE 8 ed.), The Java® Language Specification (PDF) .

[24] R. &. J. G. H. Kohavi, Wrappers for feature subset selection. Artificial intelligence, 97(1-2), 273-324, (1997).

[25] CpSolver, «https://developers.google.com/optimization/cp/cp_solver,» DevelopersGoogle. [Ηλεκτρονικό].

[26] N. F. T. M. &. N. E. Zhou, « A Comparison of CP, IP, and SAT Solvers through a Common Interface,» σε *In 2012 IEEE 24th International Conference on Tools with Artificial Intelligence (Vol. 1, pp. 41-48). IEEE.*, 2012, November.

[27] E. Scheduling, «https://developers.google.com/optimization/scheduling/employee_scheduling,» https://developers.google.com/optimization/scheduling/employee_scheduling, 2019. [Ηλεκτρονικό].

[28] K. G. Murty, «Linear programming,» *Chichester*, 1983.

[29] Pycharm, «https://www.jetbrains.com/help/pycharm/quick-start-guide.html,» Pycharm. [Ηλεκτρονικό].

[30] S. A. a. M. Marte, «University timetabling using constraint handling rules,» σε

JFPLC, 1998, pp. 39-50.

[31] T. Arbaoui, «Modeling and solving university timetabling,» (2014).

[32] E. Aycan, Solving the course scheduling problem by constraint programming and simulated annealing, Master's thesis, İzmir Institute of Technology.

[33] P. L. P. C. &. N. W. Baptiste, Constraint-based scheduling: applying constraint programming to scheduling problems, Springer Science & Business Media, (2012).

[34] I. P. &. W. T. Gent, CSPLib: a benchmark library for constraints. In International Conference on Principles and Practice of Constraint Programming, (pp. 480-481). Springer, Berlin, Heidelberg., (1999, October.

[35] D. A. F. M. Z. H. H. &. Z. A. H. Muktar, Examination Scheduling System Based on Quadratic Assignment, Assignment. In The Third International Conference on Informatics & Applications (ICIA2014) (pp. 64-71)., (2014). .

[36] A. Schaerf, A survey of automated timetabling.Artificial intelligence review, 13(2), 87-127., (1999).

[37] L. G. A. &. L. M. D. Trilling, (2006). Nurse scheduling using integer linear programming and constraint programming., IFAC Proceedings Volumes, 39(3), 671-676., (2006).

[38] G. Van Rossum, Python Programming Language, In USENIX annual technical conference (Vol. 41, p. 36)., 2007, June.

[39] googleoptimi, «https://developers.google.com/optimization,» googleoptimi. [Ηλεκτρονικό].

[40] GoogleoptimiCp, «https://developers.google.com/optimization/cp,» GoogleoptimiCp. [Ηλεκτρονικό].

[41] BooleanSat, «https://en.wikipedia.org/wiki/Boolean_satisfiability_problem,» BooleanSat. [Ηλεκτρονικό].

[42] ConstraintPrograming, «https://en.wikipedia.org/wiki/Constraint_programming,» ConstraintPrograming. [Ηλεκτρονικό].

[43] ConstraintSatisfaction, «https://en.wikipedia.org/wiki/Constraint_satisfaction,» ConstraintSatisfaction. [Ηλεκτρονικό].

[44] SatisfactionProblem, «https://en.wikipedia.org/wiki/Constraint_satisfaction_problem,» SatisfactionProblem. [Ηλεκτρονικό].

[45] Python, «https://www.python.org/,» Python. [Ηλεκτρονικό].

[46] Intelij, «https://www.jetbrains.com/idea/,» Intelij. [Ηλεκτρονικό].

10.Παράρτημα Α