ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

Πανεπιστήμιο Ιωαννίνων
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Πτυχιακή Εργασία

Χρονοπρογραμματισμός εξετάσεων χωρίς περιορισμούς χωρητικότητας

Βασίλειος Νάστος

Επιβλέπων: Χρήστος Γκόγκος
Αναπληρωτής καθηγητής Πανεπιστημίου Ιωαννίνων

20 Οκτωβρίου 2021

Uncapacitated Examination Timetabling

Εγκρίθηκε από τριμελή εξεταστική επιτροπή
Άρτα, 12/10/21

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων Καθηγητής
   Χρήστος Γκόγκος
   Αναπληρωτής Καθηγητής

2. Μέλος Επιτροπής
   Ευριππίδης Γλαβάς
   Καθηγητής

3. Μέλος Επιτροπής
   Πέτρος Καρβέλης
   Επίκουρος Καθηγητής

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Βασίλειος Νάστος

Υπογραφή

Περίληψη

Συχνό πρόβλημα που αντιμετωπίζουν τα εκπαιδευτικά ιδρύματα είναι η δημιουργία προγραμμάτων εξετάσεων, ικανοποιώντας παράλληλα έναν μεγάλο αριθμό περιορισμών οι οποίοι διαφέρουν ανάλογα με τις ανάγκες του εκάστοτε ιδρύματος, σε εξάρτηση πάντα και με τους πόρους του ιδρύματος. Ο χρονοπρογραμματισμός εξετάσεων αποτελεί ένα ενεργό πρόβλημα συνδυαστικής βελτιστοποίησης που απασχολεί αρκετούς ερευνητές σε διάφορα ερευνητικά πεδία, ενώ ένας μεγάλος αριθμός από ερευνητικά άρθρα έχει σχηματιστεί βασισμένα στο συγκεκριμένο θέμα.

Η παρούσα εργασία εξετάζει το πρόβλημα στην απλή του μορφή λαμβάνοντας υπόψη τον περιορισμό τον κοινών σπουδαστών ανά εξέταση, εξαιρώντας τους υπόλοιπους περιορισμούς. Χρησιμοποιούνται δύο σύνολα δεδομένων και συνολικά επιλύονται 25 προβλήματα. Προτείνονται διαδικασίες μείωσης μεγέθους του προβλήματος, εξετάζονται οι τεχνικές δημιουργίας αρχικών λύσεων, και προτείνονται τεχνικές βελτιστοποίησης του προβλήματος με χρήση τεχνικών τοπικής αναζήτησης, όπως η αναρρίχηση λόφων και η προσομοιωμένη ανόπτηση , εστιάζοντας και στην παραμετροποίηση των δύο αλγορίθμων.

Τέλος παρουσιάζονται τα αποτελέσματα για τα προβλήματα των συνόλων δεδομένων, τα οποία προέκυψαν έπειτα από εφαρμογή των δύο αλγορίθμων.

Λέξεις κλειδιά :Προσομοιωμένη ανόπτηση, αναρρίχηση λόφων, συνδυαστική βελτιστοποίηση, προβλήματα χρονοπρογραμματισμού

Abstract

A common problem which is faced by academical institutes is the creation of a an examination schedule. The problem consists of a various of constraints, which can be different due to the needs of an institution and the available resources of an institution. Examination timetabling is an active research field, where several articles researchers work with and many articles have been written about it.

In this thesis, we examine the uncapacitated examination timetabling problem, whereas the only imposed hard constraint is that no student should allow 6to participate in more than one examination per period, in praise of any other constraints. We use two datasets and we solve twenty-five problems. We propose techniques which reduce the problem, we examine techniques which create an initial solution and we proposed tactics in order to optimize the initial solution. We achieve that using local search algorithms such as simulated annealing and hill climbing and we present some solutions for the dataset problems we examine.

Keywords:simulated annealing, hill climbing, combinatorial optimization, timetabling problems

Ευχαριστίες

Στον καθηγητή μου, Χρήστο Γκόγκο για όλη την βοήθεια κατά την εκπόνηση της πτυχιακής εργασίας και για την μύηση στον χρονοπρογραμματισμό, και στον Άγγελο Δήμητσα για την πολύτιμη βοήθεια καθόλη την διάρκεια της πτυχιακής.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η αποδοτική δημιουργία προγραμμάτων εξετάσεων είναι ένα σημαντικό και επαναλαμβανόμενο πρόβλημα το οποίο καλούνται να αντιμετωπίσουν τα εκπαιδευτικά ιδρύματα ανά τον κόσμο. Το πρόβλημα αφορα την τοποθέτηση εξετάσεων σε χρονικές περιόδους, ώστε να μην υπάρχει σύγκρουση μεταξύ δύο εξετάσεων και εξετάσεις οι οποίες έχουν κοινούς φοιτητές να μην προγραμματίζονται την ίδια χρονική περίοδο. Η πρώτη απλοποιημένη προσέγγιση του προβλήματος προτάθηκε από τους Carter, Laporte και Lee [1], οι οποίοι διέθεσαν 13 στιγμυότυπα προβλημάτων τα οποία έχουν χρησιμοποιηθεί σε πληθώρα επιστημονικών εργασιών χρονοπρογραμματισμού. Το πρόβλημα του χρονοπρογραμματισμού εξετάσεων στην απλούστερη του μορφή αφορά εξετάσεις οι οποίες πρέπει να τοποθετηθούν σε ένα συγκεκριμένο αριθμό χρονικών περιόδων και αποτελείται από δύο βασικούς περιορισμούς:**a)κάθε εξέταση μπορεί να προγραμματιστεί σε μία χρονική περίοδο** και **b)κανένας φοιτητής δεν μπορεί να συμμετέχει σε παραπάνω από μία εξετάση ανά χρονική περίοδο**. Περιορισμοί οι οποίοι σε πραγματικά προβλήματα χρονοπρογραμματισμού υφίστανται, δεν λαμβάνονται υπόψη, όπως ο αριθμός των αιθουσών στις οποίες θα μπορούσε να διοργανωθεί μία εξέταση ή οι προτιμήσεις του εισηγητή μίας εξέτασης όσον αφορά την περίοδο διεξαγωγής. Συνεπώς ο χρονοπρογραμματισμός εξετάσεων χωρις περιορισμούς χωρητικοτήτας θα μπορούσε να θεωρηθεί μία περίληψη πραγματικών προβλημάτων χρονοπρογραμματισμού. Το πρόβλημα χρονοπρογραμματισμού εξετάσεων αποτελεί ένα ενεργό ερευνητικό πεδίο, στο οποίο δραστηριοποιούνται αρκετοί ερευνητές στον τομέα της τεχνιτής νοημοσύνης και στον τομέα της επιχειρισιακής έρευνας. Αρκετές τεχνικές έχουν προταθεί για την βέλτιστη επίλυση του προβλήματος, ωστόσο πολλές από αυτές δεν κατορθώνουν να πλησιάσουν τη βέλτιστη λύση, καθώς ο χρόνος υπολογισμού που απαιτείται είναι εκθετικά αυξανόμενος σε συνδυασμό με το μέγεθος των προβλημάτων. Στην παρούσα εργασία εξετάζεται η εφαρμογή της μεταευρετικής τεχνικής της προσομοιωμένης ανόπτησης στο πρόβλημα του χρονοπρογραμματισμού εξετάσεων, καθώς επίσης και της τεχνικής της αναρρίχισης λόφων. Οι λύσεις που εφαρμόζονται στο πρόβλημα δημιουργούνται με τη χρήση αλγορίθμων χρωματισμού γράφων. Προτείνονται επίσης τεχνικές μείωσης μεγέθους του προβλήματος, οι οποίες και εφαρμόζονται ενώ παρουσίαζονται και τα αποτελέσματα, τα οποία προέκυψαν από την εφαρμογή τεχνικών βελτιστοποιήσης στο πρόβλημα.

1.1 Δομή εργασίας

Στο κεφάλαιο 2, αναφερόμαστε στα προβλήματα P και NP, καθώς και σε ορισμένα προβλήματα χρονοπρογραμματισμού.Στο κεφάλαιο 3, πραγματοποιήται μία περιγραφή του προβλήματος του χρονοπρογραμματισμού εξετάσεων και μία ανάλυση τεχνικών μείωσης του μεγέθου προβλήματος. Εξέταζεται επίσης ποια από τα δεδομένα του προβλήματος κατηγοριοποιούνται ως 'άνευ σημασίας' και παρουσιάζεται και η ιδέα της διάσπασης του προβλήματος σε συνεκτικά τμήματα. Στο τέλος του κεφαλαίου περιγράφεται η διαδικασία χρωματισμού γράφων, μέσω της οποίας θα προκύψουν οι αρχικές λύσεις για τα προβλήματα μας. Στο κεφάλαιο 4, περιγράφονται οι διαδικασίες επίλυσης της προσομοιωμένης ανόπτησης και της αναρρίχισης λόφων. Στο κεφαλαίο 5, γίνεται αναφορά στον κώδικα που χρησιμοποιήθηκε για την επίλυση του προβλήματος και στο κεφάλαιο 6, περιγραφή των αποτελέσμάτων τα οποία προέκυψαν, έπειτα από εφαρμογή των δύο μεθόδων βελτιστοποιήσης. Τέλος στο κεφάλαιο 7, παρουσιάζονται τα συμπεράσματα που προέκυψαν κατά την εκπόνηση της πτυχιακής εργασίας, καθως ορισμένες μελλοντικές επεκτάσεις του προβλήματος.

Κεφάλαιο 2

Προβλήματα χρονοπρογραμματισμού

2.1 P και NP Προβλήματα

Η αποδοτικότητα ενός αλγορίθμου εξαρτάται από τον αριθμό των υπολογιστικών βημάτων,τα οποία θα πραγματοποιηθούν εως ότου επιλυθεί ένα πρόβλημα. Χωρίζουμε τα προβλήματα σε δύο κατηγορίες. Η πρώτη είναι τα εύκολα στην επίλυση τους (easy to solve)προβλήματα ενώ η δεύτερη τα προβλήματα τα οποία αναφέρονται ως δύσκολα στην επίλυση τους, κάτι που οδηγεί στην αύξηση των υπολογιστικών βημάτων που εκτελούνται με εκθετική μορφή. Οι αλγόριθμοι που χρησιμοποιούνται για την επίλυση των εύκολων προβλημάτων τρέχουν σε πολυωνυμικό χρόνο (class P), ενώ οι αλγόριθμοι που επιλύουν δύσκολα προβλήματα σε μη πολυωνυμικό χρόνο (Not P). Η δυσκολία ενός προβλήματος εκφράζεται συνήθως εν συναρτήση του όγκου δεδομενων. Παραδειγμα προβλήματων που παρουσιάζουν έναν αυξημένο βαθμό δυσκολίας, απότελούν το πρόβλημα του πλανώδιου πωλητή(traveling salesman problem), το πρόβλημα εύρεσης κλίκας(clique) και άλλα προβλήματα αναζήτησης για τα οποία δεν φαίνεται να υπάρχει αποτελεσματικός τρόπός επύλισης.

2.1.1 Κλάση P

Η κλάση P περιλαμβάνει το σύνολο των υπολογιστικών προβλήματων τα οποία μπορούν να επιβεβαιωθούν και να επιλυθούν σε πολυωνυμικό χρόνο. Τα προβλήματα που ανήκουν στην κλάση Π μπορούν να επιλυθούν σε χρόνο $O(n^k)$ στην χειρότερη περίπτωση. Η κλάση προβλημάτων P αποτελεί υποσύνολο της κλάσης NP . Εφόσον μπορούμε να λύσουμε ένα πρόβλημα σε πολυωνυμικό χρόνο, μπορούμε και να επιβεβαιώσουμε μία προτεινόμενη λύση σε πολυωνυιμικό χρόνο.

2.1.2 Κλάση NP

Η κλάση NP περιλαμβάνει τα προβλήματα που μπορούν επιβεβαιωθούν σε πολυωνιμικό χρόνο $O(n^k)$, δοθέντος μίας λύσης τους. Ενα πρόβλημα που ανήκει σε αυτήν την κατηγορία είναι το Sudoku, καθώς η επιβεβαιωση μίας λύσης σε ένα ημι-συμπληρωμένο ταμπλό μπορεί να πραγματοποιηθεί σε πολυωνυμικό χρονο. Στην ίδια κατηγορία τοποθετείται και το πρόβλημα εύρεσης αθροίσματος των στοιχείων ενος υποσυνόλου. Στην γενική του διατύπωση , το πρόβλημα ασχολείται με την εύρεση ενός στοχευόμενου αθροίσματος από ένα σύνολο ακέραιων αριθμών. Αν και δεν υφίσταται αλγόριθμος επίλυσης σε πολυωνυμικό χρόνο, μπορούμε να επιβεβαιώσουμε την ορθότητα μίας προτεινόμενης λύσης σε πολυωνυμικό χρόνο.

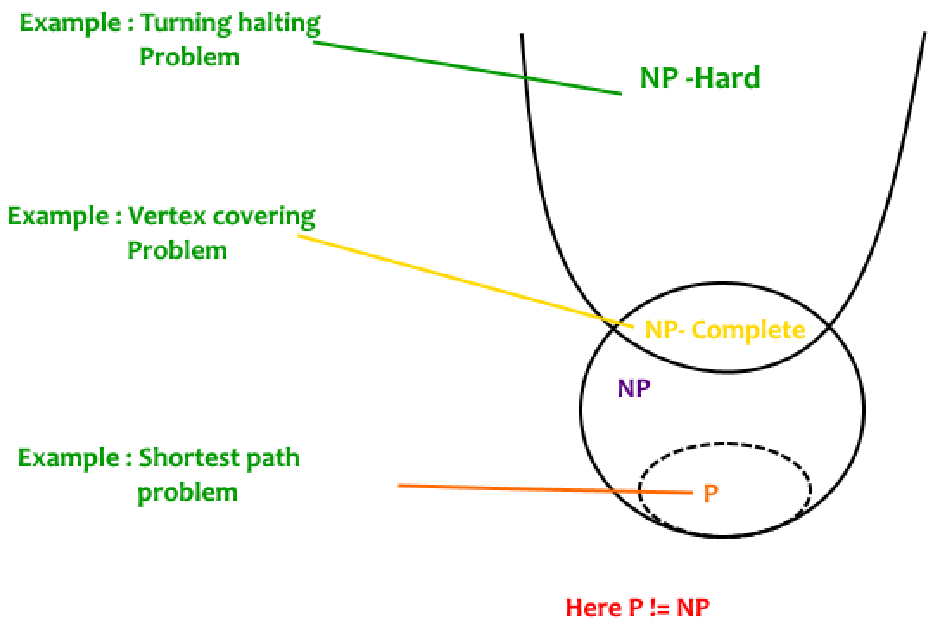
2.1.3 NP-Hardness

Ένα πρόβλημα H ανήκει στη κλάση NP-Hard, όταν κάθε πρόβλημα L της κλάσης NP μπορεί να απλοποιηθεί σε ένα πολυωνυμικό πρόβλημα H. Σε μία απλούστερη διατύπωση ένα πρόβλημα θεωρείται NP-Hard αν παρουσιάζει τουλάχιστον τον ίδιο βαθμό δυσκολίας με τα δυσκολότερα προβλήματα της κλάσης NP. Μία παρανόηση που έχει δημιουργηθεί είναι ότι το NP στο NP-Hard αντιστοιχεί στο μη πολυωνυμικό, ενώ ουσιαστικά αντιστοιχεί στον όρο μη ντετερμινιστικά πολυωνυμικά προβλήματα αποδοχής. Επίσης η υποψία ότι δεν υπάρχουν αλγόριθμοι που να επιλύουν σε πολυωνυμικό χρόνο τα προβλήματα της κλάσης NP-Hard είναι κάτι το οποίο δεν έχει αποδειχτεί [2]. Ένα πρόβλημα το οποίο ανήκει στην κατηγορία NP-Hard είναι το πρόβλημα εύρεσης αθροίσματος υποσυνόλου που περιγράφηκε προηγουμένως.

2.1.4 NP-Completeness

H κλάση NP-Complete περιλαμβάνει το σύνολο των προβλημάτων που η κατάσταση τους παραμένει άγνωστη , καθώς κανένας αλγόριθμος επίλυσης σε πολυωνυμικό χρόνο δεν έχει ανακαλυφθεί για οποιοδήποτε NP-Complete πρόβλημα, ενώ παράλληλα δεν έχει αποδειχθεί ότι δεν είναι εφικτή η επίλυση τους με χρήση αλγορίθμου πολυωνυμικού χρόνου. Στην γενική τους διατύπωση, τα προβλήματα της κλάσης NP-Complete, αφορούν NP-Hard προβλήματα τα οποία μπορούν να επιβεβαιωθούν σε πολυωνυμικό χρόνο, κάτι που εντάσσει το πρόβλημα και στην κατηγορία NP. H κατηγορία NP-Complete, αποτελεί την τομή των κατηγοριών NP και NP-Hard. Εάν ένα πρόβλημα της κλάσης NP-Complete μπορεί να επιλυθεί σε πολυωνιμικό χρόνο τότε είναι εφικτή και η επίλυση των υπόλοιπων προβλημάτων της κλάσης NP-COMPLETE σε πολυωνιμικό χρόνο. Το πρώτο NP-COMPLETE πρόβλημα το οποίο διατυπώθηκε ήταν το SAT [3]. Άλλα προβλήματα της κλάσης NP-Complete είναι το πρόβλημα εύρεσης μονοπατιού η κυκλώματος Hamilton [4]. Επίσης στην κλάση NP-Complete ανήκει και το πρόβλημα του χρονοπρογραμματισμού εξετάσεων, με το οποία ασχολείται και η παρούσα εργασία.

Συνοπτικά περιγράψαμε τις κατηγορίες υπολογιστικής πολυπλοκότητας. Στο σχήμα 2.1 παρουσιάζονται αυτές οι κατηγορίες μαζί με κάποια παραδείγματα προβλήματων που εντάσσωνται σε αυτές.



Σχήμα 2.1: Κατηγορίες υπολογιστικής πολυπλοκότητας

2.1.5 P vs NP

Το πρόβλημα στην απλή του μορφή θέτει το ερώτημα αν ένα πρόβλημα μπορεί να επιλυθεί τόσο γρήγορα από τον υπολογιστή όσο γρήγορα μπορεί να επιβεβαιωθεί η ύπαρξη της λύσης. Ο όρος γρήγορα δηλώνει την ύπαρξη ενός αλγορίθμου ώστε μία διαδικασία να εκτελείται σε πολυωνιμικό χρόνο. Το πρόβλημα P vs NP είναι ένα ανοικτό πρόβλημα στην επιστήμη των υπολογιστών. Επίσης συμπεριλαμβάνεται στα επτά προβλήματα του βραβείου millenium με αμοιβή ενός εκατομμυρίου δολαρίων για την πρώτη σωστή επίλυση του. Η απάντηση στο ερώτημα P VS NP ,μπορεί να οδηγήσει και στο συμπέρασμα αν τα προβλήματα μπορούν εξίσου να επαληθευτούν και να επιλυθούν σε πολυωνυμικό χρόνο.

2.2 Προβληματα Χρονοπρογραμματισμού

Τα προβλήματα αναθέσεων αφορούν την κατανομή πόρων σε δραστηριότητες. Τα προβλήματα χρονοπρογραμματισμού δημιουργούνται συχνά από την ανάγκη του ανθρώπου για οργάνωση γεγονότων. Οι τεχνικές οργάνωσης ανά δραστηριότητα διαφέρουν. Χαρακτηριστικά προβλήματα χρονοπρογραμματισμού είναι τα εξής:

- Χρονοπρογραμματισμός βαρδιών υπαλλήλων (Employee scheduling): Το πρόβλημα αφορά την ανάθεση βαρδιών σε ένα σύνολο υπαλλήλων μίας εταιρίας, με την κάθε βάρδια να αντιστοιχεί σε συγκεκριμένο χρονικό όριο, λαμβάνοντας υπόψη τις προτιμήσεις που μπορούν να προκύψουν ανά υπάλληλο, καθώς και τους περιορισμούς στις βάρδιες που μπορούν να προκύψουν από τους κανονισμούς που έχει ορίσει μία εταιρία(π.χ ωράριο λειτουργίας). Χαρακτηριστικό παράδειγμα αποτελεί η ανάθεση βαρδιών σε νοσοκόμες , παράδειγμα που υπήρξε και επίκεντρο του ενδιαφέροντος στον διεθνή διαγωνισμό χρονοπρογραμματισμού το 2010 και το 2014. [5].

- Χρονοπρογραπρογραμματισμός αθλητικών γεγονότων (Sports scheduling): Το πρόβλημα αφορά την δημιουργία προγραμμάτων αγώνων για αθλητικά γεγονότα, με βάση τις ανάγκες και τους περιορισμούς που μπορούν να προκύψουν για το κάθε αθλητικό γεγονός, οι οποίες διαφέρουν ανάλογα και με την φύση του αθλήματος. Ένα κλασικό παράδειγμα στο χώρο της επιχειριασιακής έρευνας αποτελεί ο χρονοπρογραμματισμός πρωταθλημάτων ποδοσφαίρου.
- Χρονοπρογραμματισμός προγραμμάτων εκπαιδευτικών ιδρυμάτων (Educational timetabling): Η δημιουργία προγράμματος εξετάσεων σε ένα εκπαιδευτικό ίδρυμα καθώς και η δημιουργία ενός προγράμματος μαθημάτων είναι διαδικασίες οι οποίες αποσχολούν έντονα εκπαιδευτικά ιδρύματα ανά τον κόσμο. Οι περιορισμοί ανά πανεπιστήμιο διαφέρουν κάτι που αυξάνει την δυσκολία του προβλήματος. [6]

Κεφάλαιο 3

Χρονοπρογραμματισμός εξετάσεων

3.1 Περιγραφή προβλήματος

Το πρόβλημα του χρονοπρογραμματισμού εξετάσεων χωρίς περιορισμούς χωρητικότητας αποτελεί ένα κλασικό πρόβλημα χρονοπρογραμματισμού. Το πρόβλημα, όπως αναφέρθηκε και στην εισαγωγή, αφορά την ομαδοποίηση εξετάσεων με βάση τους σπουδαστές που είναι εγγεγραμένοι σε διάφορες εξέτασεις, με σκοπό κανένας σπουδαστής να μην αντιμετωπίσει σύγκρουση εξετάσεων, να προγραμματιστούν δηλαδή ταυτόχρονα δύο εξετάσεις στις οποίες συμμετέχει ο σπουδαστής. Ο περιορισμός που προαναφέρθηκε αποτελεί και τον ισχυρό περιορισμό του προβλήματος του χρονοπραγραμματισμού. Αντίστοιχα διατυπώνενται και περιορισμοί οι οποίοι δεν επηρεάζουν την εφικτότητα της λύσης ακόμα και να μην τηρηθούν, επηρεάζοντας ωστόσο την ποιότητα της λύσης, στους οποίους αναφερόμαστε και ως ελαστικούς περιορισμούς. Ένας περιορισμός ο οποίος θα μπορούσε να θεωρηθεί ελαστικός είναι η ομοιόμορφη κατανομή των εξετάσεων στις διαθέσιμες περιόδους, η αντίστοιχα ο περιορισμός της σειράς προγραμματισμού των εξετάσεων, δίνοντας βαρύτητα στις εξετάσεις που παρουσιάζουν μεγάλο αριθμό κοινών σπουδαστών με τις υπόλοιπες εξετάσεις. Βασικός σκοπός είναι η επίτευξη του βέλτιστου προγράμματος εξετάσεων. Για την μελέτη του προβλήματος του χρονοπρογραμματισμού χρησιμοποιήθηκαν δύο σύνολα δεδομένων, το σύνολο δεδομένων carter και το σύνολο δεδομένων ITC, αποτελούμενα απο 13 και 12 προβλήματα αντίστοιχα. Η αναπαράσταση των προβλημάτων γίνεται με την χρήση γραφων, όπου οι κορύφες του γραφήματος αναπαριστούν τις εξετάσεις και οι ακμές του γραφήματος αναπαριστούν τους κοινούς φοιτητές μεταξύ των εξετάσεων. Το βάρος κάθε ακμής είναι αριθμός των κοινών φοιτητών μεταξύ δύο εξετάσεων. Η διαδικασία επίλυσης του προβλήματος χωρίζεται σε δύο στάδια. Το πρώτο μέρος αφορά την παραγωγή εφικτού προγράμματος εξετάσεων, την δημιουργία δηλαδή μίας αρχικής λύσης, χωρίς να προκύπτει σύγκρουση εξετάσεων για κανέναν φοιτητή. Για την παραγωγή ενός έγκυρου προγράμματος εξετάσεων με την τήρηση όλων των περιορισμών, χρησιμοποιούνται ευρετικοί αλγόριθμοι χρωματισμού γράφων. Το δεύτερο μέρος της διαδικασίας επίλυσης απότελεί την αξιολόγηση της αρχικής λύσης και την βελτιστοποιήση της, με βάση κάποιες τεχνικές βελτιστοποιήσης. Συγκεκριμένα για την βελτιστοποιήση των προβλημάτων θα χρησιμοποιηθούν αλγόριθμοι τοπικής αναζήτησης όπως ο αλγόριθμός της προσομοιωμένης ανόπτησης καθώς και ο αλγόριθμος βελτιστοποιήσης της αναρρίχησης λόφων. Κάθε λύση αξιολογήται με βάση την αντικειμενική συνάρτηση, η οποία με βάση τους ελαστικούς περιορισμούς παράγει την αντικειμενική τιμή της λύσης.

3.2 Μοντελοποιήση προβλήματος

Σε αυτή την παράγραφο, ορίζουμε τα δεδομένα αναπαράστασης που πρόκειται να χρησιμοποιηθούν αργότερα στην τρέχουσα εργασία. Ως $\mathbb{G}$, ορίζουμε τον γράφο ο οποίος θα μοντελοποιεί το πρόβλημα μας.Ως σύνολο S, ορίζουμε το σύνολο των σπουδαστών, ως σύνολο X το σύνολο των εξετάσεων και ως σύνολο P το σύνολο των περιόδων. Ως X_s, $s \in S$, ορίζουμε το σύνολο των εξετάσεων στις οποίες έχει πραγματοποιήσει εγγραφή ο φοιτητής s. Όπως προαναφέρθηκε σαν $\mathbb{G}$ ορίζουμε τον γράφο, με τον οποίο θα αναπαραστήστουμε το πρόβλημα. Για τον γράφο ορίζουμε ως $\mathbb{V}$ το σύνολο των κορυφών και ως $\mathbb{E}$ το σύνολο των ακμών. Η τιμή του βάρους ακμής για δύο εξετάσεις $x_i, x_j \in Q$ ορίζεται ως $\mathbb{W}_{x_i,x_j}$. Για μία εξέταση $x_i \in \mathbb{V}$, ορίζουμε ως $\mathbb{N}_{x_i}$, το σύνολο των εξετάσεων με τις οποίες η εξέταση x_i έχει κοινούς φοιτητές. Οι εξετάσεις με κοινούς φοιτητές ονομάζονται γειτονικές. Τέλος η περίοδος που προγραματίζεται μία εξέταση x_i ορίζεται ως F_{x_i}.

3.3 Σύνολα δεδομένων

3.3.1 Toronto Datasets

Το σύνολο δεδομένων περιέχει 13 προβλήματα πραγματικών δεδομένων πανεπιστημίων, με δεδομένα εξετάσεων δύο περιόδων (χειμερινής και εαρινής). Τα δεδομένα κάθε γραμμής του συνόλου δεδομένων χωρίζονται μεταξύ τους με κενό. Κάθε γραμμή ενός αρχείου δεδομένων περιέχει δύο αλφαριθμητικά δεδομένα. Η πρώτη γραμμή του συνόλου δεδομένων περιέχει τρία αριθμητικά δεδομένα, όπου το πρώτο αντιστοιχεί στον αριθμό των εξετάσεων που περιέχει το σύνολο δεδομένων, το δεύτερο στον αριθμό των σπουδαστών που θα συμμετέχουν στις συγκεκριμένες εξετάσεις, και το τρίτο στον μέγιστο αριθμό χρονικών περιόδων που διατίθενται ώστε να προγραμματιστούν οι εξετάσεις. Κάθε γραμμή που ξεκινάει με το γραμμα s αντιστοιχεί σε εγγραφή ενός σπουδαστή σε μία εξέταση ενώ σε αντίθετη περίπτωση η γραμμή αντιστοιχεί σε μία εξέταση και την χωρητικότητα της, η οποία δεν λαμβάνεται υπόψη. Τα σύνολα δεδομένων παρουσιάστηκαν από τους Carter,Laporte και Lee [1]. Τα προβλήματα αντιστοιχούν σε εξετάσεις πραγματικού χρόνου από τρία καναδέζικα λύκεια,πέντε καναδέζικα,1 αμερικάνικο και 1 αγγλικό πανεπιστήμιο καθώς και 1 πανεπιστήμιο στη μέση ανατολή. Στον πίνακα 2.1 παρουσιάζονται κάποια από τα δεδομένα για τα προβλήματα που υπάρχουν στο σύνολο δεδομένων carter. Αναλυτικά παρουσιάζονται, ο αριθμός των εξετάσεων ανά πρόβλημα, ο αριθμός των σπουδαστών ανά πρόβλημα, ο αριθμός των περίοδων που μπορούν κατα μέγιστο να διατεθούν για προγραμματιστούν οι εξετάσεις, καθώς και ο συντελεστής πυκνότητας οποίος ορίζει την πιθανότητα να υπάρχει σύγκρουση(κοινοί φοιτητές) μεταξύ δύο εξετάσεων. Η πιθανότητα αυτή ορίζεται από τον μαθηματικό τύπο $|E|/|V|^2$. Ο συντελεστής πυκνότητας θα αντιστοιχεί σε μία δεκαδική τιμή $d \leq 1$.

3.3.2 ITC Datasets

Το σύνολο δεδομένων ITC περιέχει δώδεκα προβληματα. Τα δωδεκα αυτά προβλήματα προβλήματα χρησιμοποιήθηκαν στον διεθνή διαγωνισμό χρονοπρογραμματισμού [7] για τα οποία πηγή προέλευσης αποτελούν διάφορα πανεπιστήμια, τα οποία αντιστοιχούν σε πραγματικά δεδομένα. Παρόμοια και με το σύνολο δεδομένων Carter, κάθε γραμμή που ξεκινάει με το γραμμα s αντιστοιχεί σε εγγραφή ενός φοιτητή σε μία ε-

Αρχείο δεδομενων	Εξετάσεις	Φοιτητές	Εγγραφές	Περίοδοι	Συντελεστής Πυκνότητας
car92	543	18149	55522	32	0.137521
car91	681	16925	56877	35	0.128125
ear83	190	1125	8109	24	0.26349
hec92	81	2823	10632	18	0.410608
kfu93	461	5349	25113	20	0.0547616
lse91	381	2726	10918	18	0.0623997
pur93	2419	30029	120681	42	0.0294718
rye93	486	11483	45051	23	0.0749124
sta83	139	611	5751	13	0.137156
tre92	261	4360	14901	23	0.17986
uta92	622	21266	58979	35	0.125195
ute92	184	2749	11793	10	0.0841801
yor83	181	941	6034	21	0.28363

Πίνακας 3.1: Χαρακτηριστικά συνόλων δεδομένων Carter

ξέταση ενώ σε αντίθετη περίπτωση η γραμμή αντιστοιχεί σε μία εξέταση, για την οποία δίνεται και πληροφορία για τον συνολικό αριθμό εγγραφών της, η οποία δεν λαμβάνεται υπόψη στην διαδικασία επίλυσης. Στον πίνακα 3.2, παρουσιάζονται τα βασικά δεδομένα των προβλημάτων του συνόλου δεδομένων ITC. Συγκεκριμένα υπολογίζονται ο αριθμός των εξετάσεων, ο αριθμός των φοιτητών, ο αριθμός των εγγραφών, ο αριθμός των κατα μέγιστο διαθέσιμων περιόδων και ο συντελεστής πυκνότητας και για τα προβλήματα του συνόλου δεδομένων ITC.

Αρχείο δεδομενων	Εξετάσεις	Φοιτητές	Εγγραφές	Περίοδοι	Συντελεστής Πυκνότητας
ITC2007_1	607	7883	32380	54	0.0503353
ITC2007_2	870	12484	37379	40	0.0116528
ITC2007_3	934	16365	61150	36	0.026159
ITC2007_4	273	4421	21740	21	0.00867682
ITC2007_5	1018	8719	34196	42	0.00867682
ITC2007_6	242	7909	18466	16	0.0613005
ITC2007_7	1096	13795	45493	80	0.0192938
ITC2007_8	598	7718	31374	80	0.0452624
ITC2007_9	169	624	2532	25	0.0768881
ITC2007_10	214	1415	7853	32	0.0489562
ITC2007_11	934	16365	61150	26	0.026159
ITC2007_12	78	1653	3685	12	0.175871

Πίνακας 3.2: Χαρακτηριστικά συνόλου δεδομένων ITC

3.3.3 Συνεκτικά τμήματα

Τα προβλήματα των δύο συνόλων δεδομένων θα αναπαρασταθούν με την χρήση γράφου. Η μοντελοποιήση αυτή των προβλημάτων μας επιτρέπει να αναζητήσουμε στον γράφο μας σύνολα κορυφών $SG_i \subseteq V$, οι κορυφές των οποίων συνδέονται μεταξύ τους, χωρίς καποία από τις κορυφές $x_i \in SG_i$ του συνόλου να δημιουργεί σύνδεση με κάποια από τις υπόλοιπες κορυφές του γραφήματος, διαδικασία η οποία οδηγεί στην δημιουργία ανεξάρτητων τμημάτων σε ένα γράφημα, τα οποία ονομάζουμε συνεκτικά τμήματα. Ο γράφος $\mathbb{G}$ θα αποτελείται από n τμήματα, όπου για κάθε συνεκτικό τμήμα θα ισχύουν οι εξής διατυπώσεις (3.1) όπως αναφέρεται και στο άρθρο [8]:

$$\begin{aligned}
&SG_i(SV_i, SE_i)\\
&SV_i \subseteq \mathbb{V}, SE_i \subseteq \mathbb{E}\\
&SG_1 \cup SG_2 \cup \ldots \cup SG_n = \mathbb{G}\\
&SG_1 \cap SG_2 \cap \ldots \cap SG_n = \emptyset
\end{aligned} \tag{3.1}$$

Η ύπαρξη συνεκτικών τμημάτων, οδηγεί στην αποσύνθεση του προβλήματος, την δημιουργία επί μέρους προβλήματων και την επίλυση του κάθε τμήματος ξεχωριστά, ως πρόβλημα. Για την εύρεση των συνεκτικών τμημάτων χρησιμοποιήθηκε η μέθοδος connected _components($\mathbb{G}$) της networkx [9]. Ο συνολικός αριθμός των συνεκτικών τμημάτων ανά πρόβλημα και για τα δύο σύνολα δεδομένων παρουσιάζεται στον πίνακα 3.3. Στον πίνακα παρατηρούμε ότι, για το σύνολο δεδομένων carter τα περισσότερα προβλήματα αποτελούνται από ένα συνεκτικό τμήμα το οποίο συγκεντρώνει τις περισσότερες εξετάσεις του προβλήματος και από πολλά μικρού μεγέθους συνεκτικά τμήματα. Χαρακτηριστικό παράδειγμα αποτελεί το αρχείο δεδομένων kfu93 το οποίο αποτελείται από έναν τμήμα το οποίο περιέχει 435 κορυφες και από άλλα 20 τμήματα που που διαθέτουν μία η δύο εξετάσεις. Αντίθετα το αρχείο δεδομένων sta83 αποτελεί εξαίρεση, καθώς χωρίζεται σε 3 τμήματα μεγέθους 30,47,62 κορυφών αντίστοιχα. Μια οπτικοποίηση του προβλήματος sta83 παρουσιάζεται στο σχήμα 3.1, όπου διακρίνονται ξεκάθαρα τα τρία συνεκτικά τμήματα. Επίσης παρατηρούμε ότι στα περισσότερα προβλήματα του συνόλου δεδομένων ITC σχηματίζεται μεγάλος αριθμός συνεκτικών τμημάτων. Ακόμα μία ιδέα που εξετάστηκε ήταν η αφαίρεση ακμών οι οποίες αποτελούν γέφυρες σε ένα γράφο, με σκοπό την δημιουργία ακόμη περισσότερων υπογράφων. Ως γέφυρα σε ένα γράφημα ορίζεται μία ακμή η οποία αν αφαιρεθεί οδηγεί στην δημιουργία επιπλέον συνεκτικών τμημάτων. Για την εύρεση των γεφυρών στα προβλήματα των συνολων δεδομένων χρησιμοποιήθηκε η συνάρτηση της networkx [9], bridges($\mathbb{G}$). Επείτα αφαιρέθηκαν οι ακμές αυτές, διαδικασία η οποία οδήγησε στην δημιουργία περισσότερων συνεκτικών τμημάτων όπως φαίνεται και στον πίνακα 3.3. Ωστόσο η αφαίρεση ακμών που ορίζονταν ως γέφυρες οδήγησε στην δημιουργία εκ' νέου συνεκτικών τμήμάτων μικρού μεγέθους. Αξίζει να σημειωθεί τα περισσότερα από αυτά τα τμήματα δεν επηρεάζουν την αντικειμενική συνάρτηση, άρα και την ποιότητα της λύσης του προβλήματος κάτι που οδήγησε και στην απόρριψη της ιδέας. Ωστόσο δεν αποκλείται το γεγονός ότι εξαιτίας της φύσης του προβλήματος πιθανή αφαίρεση τέτοιων ακμών, από σύνολα δεδομένων με διαφορετική κατανομή εξετάσεων να έδινε την δυνατότητα δημιουργίας συνεκτικών τμημάτων, που θα αποδομούσαν το πρόβλημα σε μικρότερα υποπροβλήματα, με καλύτερη κατανομή εξετάσεων ανά τμήμα.

Αρχείο δεδομένων	Υπογράφοι με μέγεθος >1	Γέφυρες	Υπογράφοι μετά από αφαίρεση γεφυρών
car92	3	5	8
car91	6	5	11
ear83	1	0	1
hec92	1	0	1
kfu93	21	10	31
lse91	3	1	4
pur93	9	12	21
rye93	3	0	3
sta83	3	0	3
tre92	2	1	3
uta92	1	1	2
ute92	2	0	2
yor83	1	0	1
ITC2007_1	3	0	3
ITC2007_2	51	51	108
ITC2007_3	31	11	42
ITC2007_4	1	0	1
ITC2007_5	53	23	76
ITC2007_6	11	9	20
ITC2007_7	85	28	113
ITC2007_8	17	3	20
ITC2007_9	8	4	12
ITC2007_10	23	4	27
ITC2007_11	31	11	42
ITC2007_12	6	0	6

Πίνακας 3.3: Αριθμός υπογράφων,γεφυρών και υπογράφων μετά την αφαίρεση την γεφυρών για τα σύνολα δεδομένων.

3.3.4 Άνευ σημασίας σπουδαστές

Οι άνευ σημασίας σπουδαστές αποτελούν σπουδαστές των προβλημάτων s$\in \mathbb{X}$ οι οποίοι δεν θα έχουν συμμετοχή σε κανένα στάδιο της επίλυσης του προβλήματος, συμμετέχοντας σε εξετάσεις οι οποίες σε οποιαδήποτε περίοδο και να διεξαχθούν θα έχουν την ίδια επιρροή στην αντικειμενική συνάρτηση. Ως 'άνευ σημασίας' σπουδαστές,όπως παρουσιάζεται και στο άρθρο [10], θεωρούνται εκείνοι οι οποίοι επιθυμούν να συμμετάσχουν σε μία και μόνο εξέταση. Οι σπουδαστές αυτοί δεν μπορούν να δημιουργήσουν σύγκρουση ανάμεσα σε δύο εξετάσεις επομένως δεν έχουν καμία επιρροή στην αντικειμενική συνάρτηση. Στον πίνακα 3.4 παρουσιάζονται οι 'άνευ σημασίας' σπουδαστές και για όλα τα προβλήματα των συνόλων δεδομένων.

Αρχείο δεδομένων	**Συνολικός αριθμός σπουδαστών**	**'Ανευ σημασίας' σπουδαστές**
car92	18419	3969
car91	16925	3409
ear83	1125	1
hec92	2823	321
kfu93	5349	276
lse91	2726	99
pur93	30029	2627
rye93	11483	2025
sta83	611	0
tre92	4360	667
uta92	21266	6180
ute92	2749	78
yor83	941	1
ITC2007_1	7883	227
ITC2007_2	12484	2430
ITC2007_3	16365	1306
ITC2007_4	4421	4
ITC2007_5	8719	407
ITC2007_6	7909	2622
ITC2007_7	13795	2620
ITC2007_8	7718	229
ITC2007_9	624	9
ITC2007_10	1415	91
ITC2007_11	16365	1306
ITC2007_12	1653	684

Πίνακας 3.4: 'Ανευ σημασίας' σπουδαστές ανά σύνολο δεδομένων

3.4 ‘Άνευ σημασίας’ εξετάσεις

3.4.1 ‘Ανευ σημασίας’ εξετάσεις με βάση ‘ασήμαντους’ σπουδαστές

Με βάση την ιδέα εύρεσης άνευ σημασίας φοιτητών θα προκύψουν εξετάσεις οι οποίες θα χαρακτηριστούν ως ‘άνευ σημασίας’. Όπως περιγράφεται και στο [10] μία εξέταση θεωρείται ως ‘άνευ σημασίας’ όταν σε αυτήν συμμετέχουν αποκλειστικά ‘άνευ σημασίας σπουδαστές’ . Αν μία εξέταση χαρακτηριστεί ως ‘άνευ σημασίας’, δεν αναμένεται να υπάρξει κάποια συμμετοχή της στην αντικειμενική συνάρτηση και η διεξαγωγή της μπορεί να πραγματοποιηθεί σε οποιαδήποτε περίοδο, χωρίς να επηρεαστεί η ποιότητα της λύσης. Μία εξέταση που χαρακτηρίζεται ‘άνευ σημασίας’, δεν πρόκειται να συμμετέχει σε κάποια από τις ακμές του γραφήματος. Τέτοιες κορυφές στο γράφημα οδηγούν στην δημιουργία συνεκτικών τμημάτων που αποτελούνται από μία κορυφή. Επομένως μπορούμε να εξαιρέσουμε αυτές τις εξετάσεις από τον γράφο $\mathbb{G}$, χωρίς να επηρεάσουμε την ποιότητας της λύσης μας. Οι εξετάσεις αυτές δεν θα έχουν καμία επιρροή στην τελική μας λύσης και συγκεκριμένα στο κόστος της, ανεξάρτητα από την περίοδο διεξαγωγής. Στον πίνακα 3.5 παρουσιάζονται οι ‘άνευ σημασίας’ εξετάσεις ανά πρόβλημα και για τα δύο σύνολα δεδομένων.

Αρχείο δεδομένων	**Αριθμός Εξετάσεων**	**'Ανευ σημασίας εξετάσεις'**
car92	543	519
car91	682	33,349,440,657
ear83	190	∅
hec92	81	∅
kfu93	461	6, 16, 22, 50, 95, 122, 178, 204, 205, 216, 285, 329, 330, 355, 369, 381, 443
lse91	381	168, 256
pur93	2419	153, 552, 976, 983, 1454, 1520
rye93	486	304
sta83	139	∅
tre92	261	186
uta92	622	∅
ute92	184	∅
yor83	181	∅
ITC2007_1	607	∅
ITC2007_2	870	38, 107, 180, 210, 228, 234, 305, 308, 316, 333, 338, 344, 502, 595, 611, 613, 618, 619, 668, 676, 748, 750, 751, 752, 789, 864
ITC2007_3	934	870, 79, 125, 126, 304, 399, 525, 531, 593, 596, 607, 610, 643, 704, 736, 737, 738, 798, 887, 934
ITC2007_4	273	∅
ITC2007_5	1018	146, 252, 496, 516, 556, 587, 657, 798, 816, 817, 819, 821, 831, 869, 873, 939
ITC2007_6	242	89, 190, 230, 239, 242
ITC2007_7	1096	21, 28, 29, 43, 62, 80, 81, 99, 132, 140, 146, 149, 150, 192, 211, 213, 214, 216, 217, 219, 222, 234, 236, 237, 249, 250, 252, 263, 266, 267, 292, 294, 295, 296, 297, 298, 299, 300, 348, 351, 353, 366, 414, 437, 586, 587, 588, 589, 593, 594, 607, 620, 654, 837, 948, 951, 979, 1024, 1075
ITC2007_8	598	0, 25, 33, 44, 50, 199, 245, 285, 322, 345, 401, 476, 477
ITC2007_9	169	1, 3, 89, 160
ITC2007_10	214	44
ITC2007_11	934	79, 125, 126, 304, 399, 525, 531, 593, 596, 607, 610, 643, 704, 736, 737, 738, 798, 887, 934
ITC2007_12	78	4, 7, 14, 45

Πίνακας 3.5: 'Ανευ σημασίας' εξετάσεις ανά πρόβλημα

3.4.2 ‘Άνευ σημασίας’ εξετάσεις με βάση το μέγεθος των συνεκτικών τμημάτων

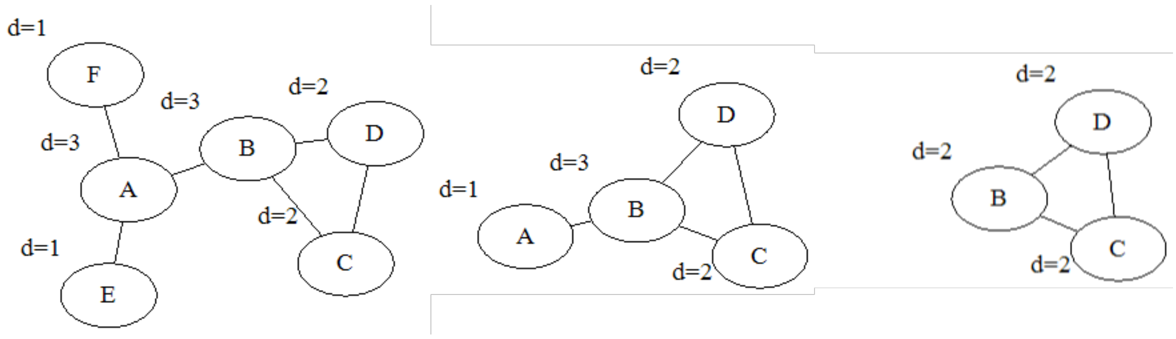
Η δεύτερη κατηγορία ‘άνευ σημασίας’ εξετάσεων, η οποία παρουσιάζεται στα προβλήματα, βασίζεται στην ιδέα των συνεκτικών τμημάτων, εισάγωντας έναν περιορισμό ορίου μεγέθος για τα τμήματα που προκύπτουν σε ένα πρόβλημα. Κάθε τμήμα του προβλήματος $G_i \in \mathbb{G}$, στο οποίο αντιστοιχεί αριθμός κορυφών μικρότερος από $\lfloor \frac{P-1}{6} \rfloor + 1$, θεωρείται ως ‘άνευ σημασίας’. Οι εξετάσεις τέτοιων τμημάτων μπορούν να διεξαχθούν σε χρονικές περιοδούς, με τον κατάλληλο τρόπο χωρίς να εχουν συμμετοχή στην αντικειμενική συνάρτηση. Επομένως οι εξετάσεις που ανήκουν στους αντίστοιχα τμήματα θεωρούνται ‘άνευ σημασίας’, κάτι που θα οδήγησει και στην αφαίρεση τους από το γράφο $\mathbb{G}$. Αυτή η κατηγορία ‘άνευ σημασίας’ εξετάσεων οδηγεί και στην εύρεση περαιτέρω σπουδαστών ‘άνευ σημασίας’, καθώς οι σπουδαστές που θα συμμετέχουν σε αυτές τις εξετάσεις δεν θα μεταβάλλουν την τελική λύση. Στον πίνακα 3.6 φαίνεται οι ‘άνευ σημασίας’ εξετάσεις ανά αρχείο δεδομένων με βάση τον περιορισμό μεγέθους των συνεκτικών τμημάτων.

Αρχείο δεδομένων	'Ανευ σημασίας' εξετάσεις
car92	253,254,519
car91	22,654,655,656,657,440,439
ear83	∅
hec92	∅
kfu93	6,138,139,16,22,285,50,178,313,314,443, 329,330,204,205,216,95,355,369,122,381
lse91	168,256
pur93	552,1454,976,1520,983,2131, 340,341,342,343,2133,153
rye93	304
sta83	∅
tre	186
uta92	∅
ute92	∅
yor83	∅
ITC2007_1	44, 45, 46, 82, 83, 84
ITC2007_2	10, 37, 38, 68, 107, 144, 180, 202, 203, 209, 210, 226, 227, 228, 229, 234, 280, 281, 284, 297, 304, 305, 306, 308, 314, 315, 316, 317, 318, 319, 333, 338, 341, 342, 344, 345, 346, 378, 379, 380, 502, 595, 606, 609, 610, 611, 613, 617, 618, 619, 643, 648, 668, 676, 688, 689, 690, 691, 748, 750, 751, 752, 757, 758, 759, 761, 762, 789, 857, 858, 864
ITC2007_3	21, 22, 23, 24, 79, 125, 126, 142, 143, 144, 239, 241, 248, 304, 308, 309, 310, 311, 312, 344, 345, 371, 399, 525, 531, 593, 596, 607, 610, 643, 669, 670, 671, 704, 736, 737, 738, 798, 887, 932, 933, 934
ITC2007_4	∅
ITC2007_5	146, 147, 148, 149, 198, 200, 202, 252, 346, 347, 348, 350, 352, 452, 453, 454, 480, 482, 491, 496, 503, 506, 508, 516, 552, 554, 556, 558, 564, 578, 582, 584, 587, 632, 634, 640, 641, 657, 798, 816, 817, 819, 821, 831, 836, 860, 866, 869, 873, 903, 939, 6636
ITC2007_6	1, 2, 89, 123, 124, 190, 194, 209, 230, 239, 242
ITC2007_7	1, 21, 28, 29, 43, 44, 62, 79, 80, 81, 85, 98, 99, 103, 132, 140, 146, 149, 150, 192, 211, 213, 214, 216, 217, 219, 222, 234, 236, 237, 242, 244, 249, 250, 251, 252, 259, 260, 261, 262, 263, 266, 267, 292, 294, 295, 296, 297, 298, 299, 300, 348, 351, 353, 366, 414, 437, 576, 578, 581, 586, 587, 588, 589, 591, 593, 594, 598, 600, 607, 620, 621, 622, 623, 624, 654, 738, 837, 948, 951, 979, 1024, 1029, 1030, 1031, 1032, 1045, 1075, 1077
ITC2007_8	10, 25, 31, 32, 33, 40, 41, 42, 44, 50, 199, 245, 285, 322, 345, 401, 476, 477, 498, 499, 500, 501
ITC2007_9	1, 3, 89, 160
ITC2007_10	8, 9, 10, 11, 16, 17, 18, 19, 36, 37, 44, 67, 68, 69, 70, 71, 72
ITC2007_11	21, 22, 23, 24, 79, 125, 126, 142, 143, 144, 239, 241, 248, 304, 308, 309, 310, 311, 312, 344, 345, 371, 399, 525, 531, 593, 596, 607, 610, 643, 669, 670, 671, 704, 736, 737, 738, 798, 887, 932, 933, 934
ITC2007_12	4, 7, 14, 45

Πίνακας 3.6: 'Ανευ σημασίας' εξετάσεις με βάση τα συνεκτικά τμήματα

3.4.3 ‘Άνευ σημασίας’ εξετάσεις με βάση των βαθμό τους

Η τρίτη κατηγορία ‘άνευ σημασίας’ εξετάσεων βασίζεται στον βαθμό μίας εξέτασης, έχοντας μοντελοποιήσει το πρόβλημα υπό την μορφή ενός γραφήματος. Σε ένα γράφημα, ο βαθμός μίας κορυφής είναι ο συνολικός αριθμός των γειτονικών κορυφών της. Στο πρόβλημα του χρονοπρογραμματισμού εξετάσεων, ως βαθμό μίας εξέτασης e ορίζουμε τον συνολικό αριθμό των εξετάσεων με τις οποίες η εξέταση e έχει κοινούς σπουδαστές. Για παράδειγμα, αν η εξέταση $e \in \mathbb{X}$, έχει ακμή με τις εξετάσεις e_1, e_2, e_3, όπου $e_1, e_2, e_3 \in N(e)$, τότε ο βαθμός της εξέτασης e είναι 3. Με βάση το άρθρο [8], όπου αποδεικνύεται ότι στο χειρότερο δυνατό σενάριο κάθε εξέταση θα ασκεί επιρροή το πολύ σε 11 περιόδους, βάσει του ορίου P/11, οπου P ο αριθμός των διαθέσιμων περιόδων οι εξετάσεις κάτω από αυτό το όριο θα είναι πάντοτε δυνατόν να τοποθετηθούν χωρίς επίπτωση στην αντικειμενική συνάρτηση. Αυτή η λογική λειτουργεί και αναδρομικά οπότε μπορούμε να αφαιρέσουμε τις εξετάσεις κάτω του ορίου, να επαναυπολογίσουμε τους βαθμούς, να ελέγξουμε εαν κάποιες εξετάσεις βρίσκονται κάτω του ορίου και να συνεχίσουμε εως ότου όλες οι εξετάσεις να είναι πάνω του ορίου. Στον πίνακα 3.7 παρουσιάζοναι οι εξετάσεις οι οποίες κατηγοριοποιούνται ως ‘άνευ σημασίας’ λόγω του βαθμού τους, για τα δύο σύνολα δεδομένων.



Σχήμα 3.1: Παράδειγμα αναδρομικής απαλοιφής ‘άνευ σημασίας’ εξετάσεων για 13 διαθέσιμες περιόδους

Αρχείο δεδομένων	**'Ανευ σημασίας' εξετάσεις**
car92	253,254,519
car91	33,349,440,654,655,656,657
ear83	∅
hec92	∅
kfu93	6,16,22,50,95,122,138,139,178,204,205,216,285,313, 314,329,330,355,369,381,443
lse91	168,256
pur93	153,340,341,342,343,552,976,983,1454,1520,2131,2133
rye93	304
sta83	∅
tre92	186
uta92	∅
ute92	∅
yor83	∅
ITC2007_1	∅
ITC2007_2	10, 37, 38, 55, 107, 144, 180, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 228, 234, 278, 280, 281, 284, 292, 297, 299, 304, 305, 308, 314, 316, 317, 319, 333, 334, 338, 340, 341, 342, 344, 345, 346, 366, 436, 502, 504, 529, 587, 588, 595, 611, 612, 613, 617, 618, 619, 643, 644, 645, 648, 654, 668, 676, 688, 689, 706, 735, 736, 748, 750, 751, 752, 754, 755, 758, 759, 789, 826, 827, 834, 841, 849, 857, 858, 864
ITC2007_3	63, 64, 69, 79, 125, 126, 131, 133, 134, 138, 139, 142, 143, 144, 156, 180, 239, 241, 242, 245, 246, 247, 248, 249, 254, 304, 311, 312, 328, 344, 345, 387, 399, 407, 442, 443, 523, 524, 525, 527, 531, 547, 578, 593, 596, 603, 607, 610, 643, 655, 669, 670, 671, 676, 678, 686, 704, 718, 720, 721, 736, 737, 738, 798, 887, 897, 901, 927, 932, 933, 934
ITC2007_4	∅
ITC2007_5	145, 146, 252, 350, 352, 494, 495, 496, 516, 552, 554, 556, 558, 563, 564, 578, 582, 584, 587, 632, 634, 636, 640, 641, 657, 663, 798, 814, 816, 817, 819, 821, 822, 831, 836, 860, 866, 869, 873, 897, 903, 939, 940
ITC2007_6	1, 2, 37, 39, 75, 76, 89, 123, 124, 190, 194, 209, 230, 239, 242
ITC2007_7	1, 21, 28, 29, 43, 44, 54, 62, 80, 81, 99, 132, 140, 146, 148, 149, 150, 163, 192, 199, 208, 211, 213, 214, 216, 217, 219, 222, 234, 236, 237, 242, 244, 249, 250, 251, 252, 261, 262, 263, 266, 267, 289, 292, 293, 294, 295, 296, 297, 298, 299, 300, 332, 348, 351, 353, 366, 414, 437, 468, 511, 585, 586, 587, 588, 589, 591, 592, 593, 594, 598, 600, 607, 618, 620, 621, 622, 623, 624, 628, 654, 683, 715, 738, 837, 948, 951, 979, 1024, 1029, 1030, 1031, 1032, 1042, 1045, 1075, 1077
ITC2007_8	10, 23, 25, 31, 32, 33, 44, 50, 199, 245, 285, 322, 345, 401, 476, 477, 564
ITC2007_9	1, 3, 43, 79, 89, 124, 138, 160
ITC2007_10	9, 16, 36, 37, 44, 67, 68
ITC2007_11	63, 64, 69, 79, 125, 126, 131, 133, 134, 138, 139, 142, 143, 144, 156, 180, 239, 241, 242, 245, 246, 247, 248, 249, 254, 304, 311, 312, 328, 344, 345, 387, 399, 407, 442, 443, 523, 524, 525, 527, 531, 547, 578, 593, 596, 603, 607, 610, 643, 655, 669, 670, 671, 676, 678, 686, 704, 718, 720, 721, 736, 737, 738, 798, 887, 897, 901, 927, 932, 933, 934
ITC2007_12	4, 7, 14, 45

Πίνακας 3.7: Άνευ σημασίας εξετάσεις με βάση τον βαθμό των εξετάσεων

Παρουσιάστηκαν τρεις κατηγορίες εξετάσεων ‘άνευ σημασίας’, οι οποίες μειώνουν το μέγεθος του προβλήματος μας. Οι τρεις κατηγορίες είναι άνευ σημασίας εξετάσεις με βάση τους ‘άνευ σημασίας’ φοιτητές, άνευ σημασίας εξετάσεις με βάση το μεθεγος των συνεκτικών τμημάτων των προβλημάτων, και άνευ σημασίας εξετάσεις με βάση τοω βαθμό κάθε εξέτασης. Στον πίνακα 3.8 παρουσιάζετε ο συνολικός αριθμός άνευ σημασίας εξετάσεων, και ο συνολικός αριθμός για κάθε κατηγορία, για τα σύνολα δεδομένων. Σημειώνεται ότι μία εξέταση μπορεί να ανήκει σε δύο κατηγορίες ‘άνευ σημασίας εξετάσεων’. Οι εξετάσεις αυτές έχουν αφαιρεθεί από τον γράφο. Κατά την διαδικασία βελτιστοποιήσης της λύσης ενός προβλήματος, οι εξετάσεις αυτές είναι ασήμαντες για το πρόβλημα, κάτι που θα μας επιτρέψει να τις αφαιρέσουμε και από τον γράφο $\mathbb{G}$, και κατ' επέκταση δεν συμμετέχουν στην αντικειμενική συνάρτηση.

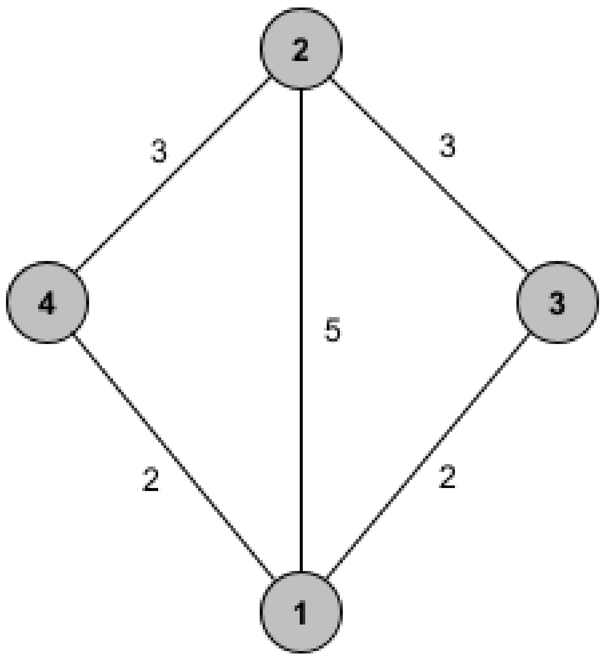
Αρχείο δεδομένων	**NE1**	**NE2**	**NE3**	**NES**
car91	4	7	11	11
car92	1	3	10	10
ear83	0	0	0	0
hec92	0	0	0	0
kfu93	17	21	29	
lse91	2	2	3	3
pur93	6	12	71	71
rye93	1	1	1	1
sta83	0	0	0	0
tre92	1	1	2	2
uta92	0	0	2	2
ute92	0	0	0	0
yor83	0	0	0	0
ITC2007_1	0	6	0	6
ITC2007_2	26	71	86	104
ITC2007_3	19	42	71	79
ITC2007_4	0	0	0	0
ITC2007_5	16	52	43	61
ITC2007_6	5	11	15	15
ITC2007_7	59	89	97	106
ITC2007_8	13	22	17	24
ITC2007_9	4	4	8	8
ITC2007_10	1	17	7	17
ITC2007_11	19	42	71	79
ITC2007_12	4	4	4	4

Πίνακας 3.8: Συνολική κατανομή άνευ σημασίας εξετάσεων στα συνόλα δεδομένων

- NE1:‘Ανευ σημασίας’ εξετάσεις βάση των σπουδαστών.
- NE2:‘Ανευ σημασίας’ εξετάσεις βάση των συνεκτικών τμημάτων.
- NE3:‘Ανευ σημασίας’ εξετάσεις βάση βαθμού εξέτασης.
- NES:Συνολικός αριθμός ‘Ανευ σημασίας’ εξετάσεων.

3.5 Συμμετρικές εξετάσεις

Σε αυτήν την παράγραφο θα ασχοληθούμε με τις συμμετρικές εξετάσεις, εξετάσεις δηλαδή οι οποίες μπορούν να προγραμματιστούν στην ίδια περίοδο και έχουν κοινά χαρακτηριστικά. Το πρώτο σύνολο συμμετρικών εξετάσεων με τις οποίες θα ασχοληθούμε, αφορούν εξετάσεις οι οποίες δεν έχουν κοινούς σπουδαστές μεταξύ τους, ωστόσο έχουν τις ίδες γειτονικές εξετάσεις και για κάθε μία από αυτές που συνδέονται έχουν τον ίδιο αριθμό κοινών σπουδαστών. Αυτό το σύνολο συμμετρικών εξετάσεων το ονομάζουμε $\mathbb{I}$. Αυτές οι εξετάσεις θα προγραμματιστούν την ίδια βέλτιστη χρονική στιγμή σε μία βέλτιστη λύση, μιας και δεν συγκρούονται μεταξύ τους και παρουσιάζουν συμμετρία ως προς τον τρόπο επίλυσης τους, συμπερένωντας ότι, η λύση θα είναι κοινή ανεξαρτήτως της εξέτασης η οποία θα επιλεχθεί, χωρίς να υφίστανται διαφορετικές βέλτιστες λύσεις για κάποια αυτές τις εξετάσεις. Ένα παράδειγμα συμμετρικών εξετάσεων φαίνεται στην εικόνα ανάμεσα στις εξετάσεις 3 και 4. Εξατάσεις που παρουσιάζουν την συγκεκριμένη μορφή συμμετρίας μπορούν να προγραμματιστούν ταυτόχρονα. Άλλη μία μορφή συμμετρικών εξετάσεων που παρουσιάζεται στα προβλήματα είναι οι εξετάσεις που έχουν τους ίδιους εγγεγραμμένους σπουδαστές. Αυτές οι εξετάσεις θα προγραματίζονται πάντα σε διαφορετική περίοδο, στοχεύοντας πάντα στην κατανομή που θα επιτύχει την μικρότερη συμμετοχή στην αντικειμενική συνάρτηση. Αναζητώντας συμμετρικές εξετάσεις, επιτυνχάνουμε την μείωση του χώρου αναζήτησης λύσεων, καθώς αποφεύγουμε την εξετάσεις με παρόμοια μορφή. Ο αριθμός των εξετάσεων οι οποίες ανήκουν στο σύνολο I παρουσιάζετε στον πίνακα και για τα σύνολα δεδομένων.



Σχήμα 3.2: Εξετάσεις 3,4 $\in \mathbb{I}$

Αρχείο δεδομένων	Συμμετρικές εξετάσεις
car92	∅
car91	∅
ear83	∅
hec92	∅
kfu93	232,237
lse91	∅
pur93	∅
rye93	∅
sta83	∅
tre92	∅
uta92	∅
ute92	∅
yor83	∅
ITC2007_1	∅
ITC2007_2	∅
ITC2007_3	∅
ITC2007_4	∅
ITC2007_5	301,302,836,350,632,636,739,741
ITC2007_6	1,209
ITC2007_7	312,41,310,165,78,715,511
ITC2007_8	∅
ITC2007_9	140,159
ITC2007_10	∅
ITC2007_11	∅
ITC2007_12	∅

Πίνακας 3.9: Συμμετρικές εξετάσεις

3.6 Χρωματισμός Γράφων

Το πρόβλημα του χρωματισμού των γράφων αποτελεί ένα από τα πιο ιστορικά και ευρέως μελετημένα προβλήματα της θεωρίας των γράφηματων με αρκετές πρακτικές εφαρμογές σε διάφορα πεδία. Συγκεκριμένα ως χρωματισμό γράφου ορίζουμε την απόδοση χρωμάτων στις κορυφές του ώστε να μην υπάρχουν κορυφές με ακμή που να έχουν χρωματιστεί με το ίδιο χρώμα. Για ένα γράφο $\mathbb{G}$, ορίζουμε ως c-χρωματισμό την ανάθεση c χρωμάτων στους κόμβους του γραφήματος, τηρώντας πάντα τον περιορισμό κοινού χρωματισμού ανέμεσα σε γειτονικές κορυφές. Ως χρωματική κλάση X_i, ορίζουμε το σύνολο των κόρυφών που θα τους αποδοθεί ένα συγκεκριμένο χρώμα. Ως χρωματικό αριθμό ορίζουμε το σύνολο των χρωμάτων που θα χρησιμοποιηθούν στην διαδικασία ενός έγκυρου χρωματισμού. Μερικές από τις εφαρμογές που χρησιμοποιήται ο χρωματισμός γράφων είναι:

- **Δημιουργία προγραμματων και χρονοπρογραμμάτων**
- **Κατανομή ραδιοφωνικών συχνοτήτων(Mobile Radio Frequency Assignment) [11]**.
- **Sudoku**

Στο πρόβλημα του χρονοπρογραματισμού εξετάσεων τα χρώματα αντιστοιχούν στις διαθέσιμες χρονικές περιόδους. Από την μοντελοποιήση του προβλήματος ως ένος μη κατευθυνόμενου γράφου, οι εξετάσεις αντιστοιχούν στις κορυφές. Οι ακμές αντιπροσωπεύουν την ύπαρξη κοινών φοιτητών ανάμεσα σε δύο εξετάσεις, ενώ το βάρος μίας ακμής των αριθμό των κοινών φοιτητών ανάμεσα σε δύο εξετάσεις, όπως αναφέρεται και στο κεφάλαιο 1. Σκοπός είναι είναι να προγραμματίσουμε τις εξετάσεις κάθε προβλήματος αποφεύγοντας την τοποθέτηση εξετάσεων με κοινούς φοιτητές στην ίδια χρονική περίοδο. Επιπλέον περίορισμος υπάρχει στον αριθμό των περιόδων που είναι διαθέσιμες για κάθε πρόβλημα. Στον πίνακα 3.10 παρουσιάζονται τα προβλήματα όλων των συνόλων δεδομένων καθώς και ο διαθέσιμός αριθμός περιόδων ανά πρόβλημα. Με τη μέθοδο του χρωματισμού επιτυνχάνουμε την δημιουργία ένος αρχικού προγράμματος των εξετάσεων. Για να θεωρηθεί έγκυρή μία λύση πρέπει ο αριθμός των περιόδων να είναι μικρότερος από τον όριο περιόδων, που φαίνεται στον πίνακα 3.10, για κάθε πρόβλημα χωρίς να παραβιάζεται ο ισχυρός περιορισμός τον οποίο ορίσαμε. Για την εκτέλεση χρωματισμού για τους γράφους, στα προβλημάτα των συνόλων δεδομένων, χρησιμοποιήθηκε η συνάρτηση της βιβλιοθήκης networkx [9] greedy_color($\mathbb{G}$). Οι στρατηγικές που χρησιμοποιούνται για τον χρωματισμό των γράφων είναι άπληστες κάτι που σημαίνει ο αριθμός των περιόδων που θα χρησιμοποιήσει θα είναι κατα μέγιστο d+1, όπου το d ορίζεται ως ο μέγιστος βαθμός εξέτασης στον γράφο των οποίο εξετάζουμε. Οι στρατηγικές χρωματισμού που χρησιμοποιούνται και υποστηρίζονται και από το πακέτο της networkx είναι οι εξής:

- largest_first
- random_sequential
- smallest_last
- independent_set
- connected_sequential_bfs
- connected_sequential_dfs
- saturation_largest_first|DSATUR

Επίσης για κάποιες από τις στρατηγικές, η βιβλιοθήκη της networkx υποστηρίζει εναλλακτικές μορφές κινήσεων χρωματισμού, οι οποίες ορίζεται ως intechangable_coloring. Δοθέντος μίας συνάρτησης χρωματισμού CF και μίας εξέτασης e, όπου $CF(e) \in [i, j]$, οι τεχνικές εναλλακτικού χρωματισμού επιτρέπουν τον επανορισμό της συνάρτησης χρωματισμού, ώστε αν $CF(e) \in i$, με τον επανορισμό της συνάρτησης χρωματισμού να ισχύει $CF(e) \in j$, δύνοτας την δυνατότητα στην συνάρτηση χρωματισμού να αποφύγει ένα καινούργιο χρώμα, εναλλάσοντας χρώματα ανά τους υπογράφους $G_i \in \mathbb{G}$ όταν είναι εφικτό, τροποποιώντας την υπαρχουσα λύση [12], χωρίς να απαιτήται η δημιουργία νέου χρώματος. Οι στρατηγικές οι οποίες υποστηρίζουν εναλλακτικές μορφές χρωματισμού είναι οι εξής:

- largest_first
- random_sequential
- smallest_last
- connected_sequential_bfs

- connected_sequential_dfs

Στον πίνακα 3.9 παρουσιάζονται ο αριθμός μέγιστος αριθμός περιόδων ο οποίος μπορεί να χρησιμοποιηθεί ανα πρόβλημα. Έγκυρα προγράμματα εξετάσεων, θα θεωρούνται εκείνα τα οποία ο αριθμός περιόδων που χρησιμοποιούν είναι μικρότερος από το αριθμό που δίνεται σαν όριο περιόδων.

Αρχείο δεδομένων	Μέγιστος αριθμός περιόδων
car92	32
car91	35
ear83	24
hec92	18
kfu93	20
lse91	18
pur93	42
rye93	23
sta83	13
tre92	35
uta92	10
ute92	21
yor83	54
ITC2007_1	40
ITC2007_2	36
ITC2007_3	21
ITC2007_4	42
ITC2007_5	16
ITC2007_6	80
ITC2007_7	80
ITC2007_8	25
ITC2007_9	32
ITC2007_10	26
ITC2007_11	12
ITC2007_12	38

Πίνακας 3.10: Μέγιστος αριθμός περιόδων ανά σύνολο δεδομένων

3.6.1 Largest First

Ο αλγόριθμος largest first αποτελεί έναν από τους πιο δημοφιλής άπληστους ευρετικούς αλγορίθμους που χρησιμοποιούνται στο πρόβλημα του χρωματισμού γράφων. Οι χρωματικές κλάσεις κατασκευάζονται με χρήση άπληστων επιλογών. Ο αλγόριθμος χρωματίζει την κορυφή με τον μεγαλύτερο βαθμό τοποθετώντας την σε μία χρωματική κλάση C_i και στην συνέχεια επιλέγει και τοποθετεί στην ίδια χρωματική κλάση κορυφές που έχουν όσο το δυνατόν λιγότερες γειτονικές κορυφές που μπορούν να τοποθετηθούν στην χρωματική κλάση C_i. Ο αλγόριθμος αναλαμβάνει την κατασκευή συνόλων που θα αποτελούν τις κορυφές που ανήκουν σε χρωματικές κλάσεις. Στο σχήμα 3.2 περιγράφεται η διαδικασία κατασκευής χρωματικής κλάσης του αλγορίθμου largest_first. Σαν είσοδος εισάγεται η κορυφή που θέλουμε να χρωματιστεί, και σαν έξοδος επιστρέφεται το σύνολο με τις κορυφές που θα ανήκουν στην χρωματική κλάση που θα τοποθετηθεί η κορυφή u. Το σύνολο U, περιέχει τις κορυφές που δεν έχει χρωματίσει ο αλγόριθμος, ενώ το σύνολο W, τις κορυφές $e \in U$, οι οποίες συνδέονται με την κορυφή u. Ο αλγόριθμος επιλέγει την κορυφή που έχει τους περισσότερους γείτονες στο σύνολο W. Έπειτα πραγματοποιήται ο χρωματισμός της κορυφής και η διαδικασία επαναλαμβάνεται όσο $U \neq \emptyset$ [13]. Για να εκτελέσουμε χρωματισμό γράφων στο παράδειγμα μας με χρήση του αλγορίθμου largest first χρησιμοποιήσαμε την συνάρτηση greedy˝color της βιβλιοθήκης networkx [9] . Η στρατηγική largest˝first υπάρχει ενσωματωμένη σαν στρατηγική χρωματισμού στην βιβλιοθήκη της networkx.

Κατασκευή χρωματικής κλάσης C_v.
Είσοδος Ένα σύνολο U που περιέχει τις μη χρωματισμένες κορυφές και μία κορυφή που θα εξεταστεί $u \in U$.
Έξοδος Ένα σύνολο C_v που θα αποτελείται από τις κορυφές που θα σχηματίζουν την χρωματική κλάση C_v.
Αρχικοποίηση του $\mathbb{W}$ ως σύνολο από τις κορυφές που συνδέονται με την u.
Διαγραφή της κορυφής u και των γειτόνικών κορυφών της από το σύνολο U.
while $U \neq \emptyset$ **do**
    Επιλογή μίας κορυφής $u \in \mathbb{U}$ με τον μεγαλύτερο αριθμό γειτόνων στο σύνολο $\mathbb{W}$.
    Σε περίπτωση ισοπαλιών επιλέγεται η κορυφή με τον μικρότερο αριθμό γειτόνων στο σύνολο $\mathbb{U}$.
    Μετακίνηση της κορυφής u από το σύνολο U στο σύνολο C_v, και μετακίνηση των γειτονικών κορυφών $n_i \in U$ της κορυφής u στο σύνολο W.
end while

Σχήμα 3.3: Αλγόριθμος κατασκευής χρωματικής κλάσης

Αρχείο δεδομένων	Αριθμός περιόδων	Αριθμός περιόδων με χρήση εναλλακτικής συνάρτησης χρωματισμού
car92	32	31
car91	34	31
ear83	26	23
hec92	20	19
kfu93	20	19
lse91	19	18
pur93	38	34
rye93	25	22
sta83	13	13
tre92	23	22
uta92	36	33
ute92	11	10
yor83	23	22
ITC2007_1	23	22
ITC2007_2	15	15
ITC2007_3	23	23
ITC2007_4	20	19
ITC2007_5	14	13
ITC2007_6	15	13
ITC2007_7	20	19
ITC2007_8	22	21
ITC2007_9	13	11
ITC2007_10	18	18
ITC2007_11	23	23

Πίνακας 3.11: Αποτελέσματα αλγορίθμου Largest First

3.6.2 Smallest Last

Άλλη μία μέθοδος που χρησιμοποιήται για να επιτευχθεί χρωματισμός ενός γράφου είναι η στρατηγική smallest last. Για έναν γράφο $\mathbb{G}$, ο αλγόριθμος υπολογίζει των βαθμό των κορυφών, επιλέγει την κορυφή με τον μικρότερο βαθμό v_i την οποία και χρωματίζει με το πρώτο διαθέσιμο χρώμα. Στην συνέχεια η κορυφή v_i που επιλέχθηκε θα αφαιρεθεί από τον γράφο, και θα επιλεχθεί ξανά η κορυφή με τον μικρότερο βαθμό. Η διαδικασία χρωματισμού εκτελείται επαναληπτικά εώς ότου ολοκληρωθεί ο χρωματισμός όλων των κορυφών του γραφήματος. Στον πίνακα 3.11 παρουσιάζονται το σύνολο των περιόδων που παράγει για το κάθε πρόβλημα η στρατηγική smallest last [14].

Αρχείο δεδομένων	Αριθμός περιόδων	Αριθμός περιόδων με χρήση εναλλακτικής συνάρτησης χρωματισμού
car92	33	29
car91	34	30
ear83	23	22
hec92	19	18
kfu93	20	19
lse91	18	17
pur93	39	35
rye93	22	21
sta83	13	13
tre92	24	21
uta92	33	31
ute92	10	10
yor83	22	21
ITC2007_1	22	21
ITC2007_2	15	15
ITC2007_3	23	22
ITC2007_4	20	19
ITC2007_5	13	13
ITC2007_6	14	14
ITC2007_7	20	17
ITC2007_8	21	20
ITC2007_9	12	11
ITC2007_10	18	18
ITC2007_11	23	22
ITC2007_12	12	12

Πίνακας 3.12: Αποτελέσματα αλγορίθμου Smallest Last

3.6.3 Saturation Largest First(DSATUR)

Ο αλγόριθμος DSatur αποτελεί έναν ευρετικό αλγόριθμο χρωματισμού γράφων. Η ιδέα του αλγορίθμου αρχικά διατυπώθηκε από τον Daniel Brelaz, το 1979 και παρόμοια με τους άπληστους αλγορίθμους χρωματίζει τις κορυφές με την χρήση ένος χρώματος που δεν έχει χρησιμοποιηθεί για κάποια γειτονική κορυφή. Στο σχήμα 3.3 παρουσιάζονται τα βήματα εκτέλεσης του αλγορίθμου DSatur. Πρώτο βήμα αλγορίθμου αποτελεί η επιλογή της κορυφής $e \in V$ με τον μεγαλύτερο βαθμό. Το πρώτο χρώμα θα εφαρμοστεί στην πρώτη κορυφή που θα επιλεχθεί. Στην συνέχεια για τις υπόλοιπες κορυφές υπολογίζετε το επίπεδο κορεσμού και πραγματοποιήται επιλογή της κορυφής n_i με την υψηλότερο τιμή κορεσμού ($deg(n_i)$). Ως επίπεδο κορεσμού ορίζουμε τον συνολικό αριθμό τον γειτονικών κορυφών μίας κορυφής, για τις οποίες δεν έχει πραγματοποιηθεί απόδοση κάποιου χρώματος. Αν πολλές κορυφές έχουν την ίδια μέγιστη τιμή κορεσμού, τότε επιλέγεται τυχαία μία από τις κορυφές η κορυφή με τον μεγαλύτερο βαθμό. Στην κορυφή n_i εφαρμόζεται το πρώτο διαθέσιμο εφικτό χρώμα. Η διαδικασία συνεχίζει να εκτελείται επαναληπτικά εως ότου δεν απομένει καμία κορυφή προς εξέταση. Στον πίνακα 3.12 παρουσιάζονται οι συνολικές περίοδοι που θα χρησιμοποιηθούν για κάθε πρόβλημα στα δύο σύνολα δεδομένων, έπειτα από εφαρμογή του αλγορίθμου.

DSATUR.

Είσοδος Ο γράφος $\mathbb{G}$
Έξοδος Αποτελέσμα χρωματισμού c_v:$v \in V$
C:=$\emptyset$, U:=V, computedeg$_{G(U)}$ (συνάρτηση εύρεσης βαθμού κορεσμού κορυφής).
Επιλογή κορυφής με τον υψηλότερο βαθμό κορεσμού.
$c(v)$:=1, C:=$C \cup \{v\}$, U:=$U/\{v\}$·
Ενημέρωση βαθμών κορεσμού.
while $U \neq \emptyset$ **do**
    Εύρεση κορυφής $u_i \in U$ με μέγιστο βαθμό κορεσμού.
    Εύρεση του του πρώτου διαθέσιμου χρώματος k για χρωματισμό της κορυφής u_i.
    $c(u_i) := k$, C:=$C \cup \{u_i\}$, U:=$U/\{u_i\}$
    Ενημέρωση βαθμών κορεσμού των κορυφών του προβλήματος.
end while

Σχήμα 3.4: Ο Ευρετικός αλγόριθμος DSatur

Αρχείο δεδομένων	Αριθμός περιόδων
car92	30
car91	31
ear83	23
hec92	19
kfu93	19
lse91	19
pur93	35
rye93	22
sta83	13
tre92	23
uta92	31
ute92	10
yor83	20
ITC2007_1	21
ITC2007_2	15
ITC2007_3	22
ITC2007_4	18
ITC2007_5	13
ITC2007_6	14
ITC2007_7	19
ITC2007_8	21
ITC2007_9	12
ITC2007_10	18
ITC2007_11	22
ITC2007_12	12

Πίνακας 3.13: Μέγιστος αριθμός περιόδων ανά σύνολο δεδομένων

Συνολικά θα χρησιμοποιηθούν 7 στρατηγικές χρωματισμού καθώς και πέντε παραλλαγές των πέντε από τον επτά στρατηγικών. Οι τρεις στρατηγικές που επιτυγχάνουν έγκυρο χρωματισμό σε συνδυασμό με τον χρήση όσο το δυνατόν λιγότερων περιόδων στα περισσότερα από τα προβλήματα του συνόλου δεδομένων είναι οι στρατηγικές largest first,smallest last,saturation largest first, ενώ λόγο της δομής της καλύτερα αποτελέσματα θα παραχθούν και από τις στρατηγικές που χρησιμοποιούν εναλλακτικές στρατηγικές χρωματισμού. Στον πίνακα 3.14 παρουσιάζονται τα αποτελέσματα για τους υπόλοιπες στρατηγικές χρωματισμού που χρησιμοποιήθηκαν και εφάρμοσαν στα προβλήματα έναν έγκυρο αριθμό περιόδων. Αξίζει να σημειωθεί ότι η στρατηγική random sequential εισάγει τυχαιότητα στον τρόπο κατανομής των περιόδων, διαφοροποιώντας τα αποτελέσματα του αλγορίθμου ανά εκτέλεση της.

Αρχείο δεδομένων	Έγκυροι αλγόριθμοι χρωματισμού
car92	LF, SLF, LFI, SLI
car91	LF, SL, SLF, LFI, SLI
ear83	SL, SLF, LFI, SLI,
hec92	SLI
kfu93	LF, SL, SLF, LFI, SLI, CSBI
lse91	SL, LFI, SLI
pur93	LF, SL, SLF, LFI, SLI,
rye93	SL, SLF, LFI, SLI, CSBI
sta83	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
tre92	LF, SLF, LFI, SLI,
uta92	SL, SLF, LFI, SLI,
ute92	SL, SLF, LFI, RSI, SLI, CSBI,
yor83	SLF, SLI,
ITC2007_1	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_2	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_3	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_4	LF, SL, SLF, LFI, SLI
ITC2007_5	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_6	LF, SL, CSB, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_7	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_8	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_9	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_10	LF, RS, SL, IS, CSB, CSD, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_11	LF, SL, SLF, LFI, RSI, SLI, CSBI, CSDI
ITC2007_12	SL, SLF, LFI, SLI

Πίνακας 3.14: Στρατηγικές οι οποίες παράγουν εφικτό χρωματισμό

- LF: Largest First
- RS: Random Sequential
- SL: Smallest Last
- IS: Independent Set
- CSB: Connected Sequential Bfs
- CSD: Connected Sequential Dfs
- SLF: Saturation Largest First
- LFI :Largest First Interchange
- RSI: Random Sequential Interchange
- SLI: Smallest Last Interchange
- CSBI: Connected Sequential Bfs Interchange
- CSDI: Connected Sequential Dfs Interchange

Κεφάλαιο 4

Διαδικασίες επίλυσης

4.1 Εισαγωγή

Το πρόβλημα του χρονοπρογραμματισμού χωρίζεται σε δύο σταδια επίλυσης. Το πρώτο είναι το στάδιο της δημιουργίας αρχικής λύσης που πραγματοποιήται με την χρήση μεθόδων χρωματισμού γράφων, και δημιουργεί για το πρόβλημα μία αρχική λύση s_0. Το δεύτερο στάδιο επικεντρώνεται στην αξιολόγηση και την βελτιστοποίηση της αρχικής λύσης. Για την βελτιστοποιήση των λύσεων μας χρησιμοποιήθηκε ο μεταευρετικός αλγόριθμος της προσομοιωμένης ανόπτησης(simulated annealing) καθώς και η τεχνική της αναρρίχησης λόφων(hill climbing). Και οι δύο μέθοδοι αποτελούν τεχνικές μαθηματικής βελτιστοποιήσης.

Οι δύο τεχνικές βελτιστοποιήσης πραγματοποιούν αναζήτηση της βέλτιστης λύσης σε έναν μεγάλο χώρο αναζήτησης. Ως χώρο αναζήτησης ορίζουμε τις πιθανές εφικτές λύσεις στις οποίες μπορεί να πραγματοποιηθεί μετάβαση από την αρχική λύση, καθιστώντας τις ως υποψήφιες λύσεις. Κάθε υποψήφια λύση αξιολογείται ως προς την επιρροή της στην τρέχουσα λύση. Η αξιολόγηση των υποψήφιων λύσεων πραγματοποιήται με την βοήθεια της αντικειμενικής συνάρτησης. Ονομάζουμε την αντικειμενική τιμή η οποία θα προκύψει από την συνάρτηση, κόστος λύσης. Για τον υπολογισμό του κόστους της λύσης θεωρούμε ότι θα επιβάλλεται ποινή ανάμεσα σε δύο γειτονικές εξετάσεις, ανάλογα με την κατανομη των περιόδων τους. Γειτονικές εξετάσεις που έχουν προγραμματιστεί με διαφορά κατά απόλυτη τιμή μέχρι πέντε περιόδους, συμμετέχουν στην κοστολόγηση της λύσης. Για την κοστολόγηση μίας λύσης χρησιμοποιούνται οι εξής κανόνες:

- Για κάθε $x_i, x_j \in X$, αν η εξέταση x_i έχει κοινούς φοιτητές με την εξέταση x_j, και η περίοδος που προγραμματιστηκε η εξέταση x_i απέχει κατα απόλυτη τιμή 1, από την περίοδο που προγραμματτιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 16. Η συμμετοχή στην που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι $16 * W_{x_i x_j}$.
- Για κάθε $x_i, x_j \in X$, αν η εξέταση x_i έχει κοινούς φοιτητές με την εξέταση x_j, και η περίοδος που προγραμματιστηκε η εξέταση x_i απέχει κατα απόλυτη τιμή 2, από την περίοδο που προγραμματιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 8. Η συμμετοχή που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι $8 * W_{x_i x_j}$.

- Για κάθε $x_i, x_j \in X$, αν η εξέταση x_i έχει κοινούς φοιτητές με την εξέταση x_j, και η περίοδος που προγραμματιστηκε η εξέταση x_i απέχει κατα απόλυτη τιμή 3, από την περίοδο που προγραμματτιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 4. Η συμμετοχή που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 4 * $\mathbb{W}_{x_i x_j}$.
- Για κάθε $x_i, x_j \in X$, αν η εξέταση x_i έχει κοινούς φοιτητές με την εξέταση x_j, και η περίοδος που προγραμματιστηκε η εξέταση x_i απέχει κατα απόλυτη τιμή 4, από την περίοδο που προγραμματιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 2. Η συμμετοχή που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 2 * $W_{x_i x_j}$.
- Για κάθε $x_i, x_j \in X$, αν η εξέταση x_i έχει κοινούς φοιτητές με την εξέταση x_j, και η περίοδος που προγραμματιστηκε η εξέταση x_i απέχει κατα απόλυτη τιμή 5, από την περίοδο που προγραμματιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 1. Η συμμετοχή που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι $1 * W_{x_i x_j}$.

Εφαρμόζωντας τους παραπάνω κανόνες σε όλα τα ξεύγη γειτονικών εξετάσεων, υπολογίζεται το κόστος της λύσης, που αντιστοιχεί σε μία ακέραια τιμή, και μπορεί να κανονικοποιηθεί ανά σπουδαστή.

4.2 Περιγραφή αλγορίθμου προσομοιωμένης ανόπτησης

Μια πολύ σημαντική μέθοδος στην επίλυση προβλημάτων βελτιστοποιήσης είναι η μέθοδος της προσομοιωμένης ανόπτησης. Ο όρος της ανόπτησης χρησιμοποιήται κατά κύριο λόγο στην θερμοδυναμική και αντιστοιχεί στην θέρμανση του υλικού μέχρι το σημείο τήξεως του και εν συνεχεία την αργή και ελεγχόμενη ψύξη του υλικού με σκοπό το υλικό να πέσει στο χαμηλότερο στάδιο ενέργειας όταν τελειώση η ψύξη, και την μεταβολή τον φυσικών ιδιοτήτων του υλικού.

Η διαδικασία της προσομοιωμένης ανόπτησης εκκεινεί με μία συγκεκριμένη υψηλή τιμή θερμοκρασίας. Σε κάθε επανάληψη η προσομοιωμένη ανόπτηση χρησιμοποιεί μία γειτονιά καινούργιων υποψήφιων λύσεων $N(s)$ που μπορούν να δημιουργηθούν με βάση την τρέχουσα λύση s_0, ενώ ως s_{best} ορίζουμε την βέλτιστη λύση για το πρόβλημα. Η επιλογή μίας υποψήφιας λύσης $c_s \in N(s)$ πραγματοποιήται με τυχαίο τρόπο. Για την δημιουργία υποψήφιων λύσεων χρησιμοποιούνται οι τελεστές γειτνίασης, οι οποίοι εκτελούν μεταβολές περιόδων, πραγματοποιώντας κινήσεις οι οποίες θα οδηγήσουν στην δημιουργία νέων υποψήφιων λύσεων. Από την παραπάνω διαδικασία θα προκύψει μία υποψήφια λύση $s_n \in N(s)$. Η νέα αυτή λύση θα αξιολογηθεί με βάση την αντικειμενική συνάρτηση, κάτι που οδηγήσει στον υπολογισμό του κόστους της λύσης s_n. Σε περίπτωση που το κόστος της λύσης s_n είναι μικρότερο από το κόστος της λύσης s_{best}, τότε ορίζουμε ως καλύτερη λύση την υποψηφια s_n. Σε αντίθετη περίπτωση υπολογίζεται η διαφορά d του κόστους της λύσης s_n, από το αντίστοιχο κόστος της λύσης s_0. Αν $d > 0$, τότε ορίζεται μία πιθανότητα αποδοχής της υποψήφιας λύσης ($s_0 = s_n$), κάτι που θα οδηγήσει στην αύξηση του κόστους λύσης. Η πιθανότητα αυτή προέρχεται από τους νόμους της θερμοδυναμικής και προκύπτει από τον τύπο:

$$p = e^{(-d/t)} \tag{4.1}$$

όπου το t αντιστοιχει στην τιμή της θερμοκρασίας στον τρέχον βήμα του αλγορίθμου. Με τον τρόπο αυτό η διαδικασία της προσομοιωμένης ανόπτησης επιτρέπει την αποδοχή καταστάσεων οι οποίες δεν οδηγούν σε μείωση του κόστους λύσης. Παρατηρείται ότι σε υψηλότερες θερμοκρασίες, αυξάνεται η πιθανότητα να γίνουν αποδεκτές λύσεις που αυξάνουν την τιμή το κόστος. Στην συνέχεια του αλγορίθμου πραγματοποιήται μείωση της θερμοκρασίας με βάση τον συντελεστή ψύξης a, ο οποίος και συνιθίζεται να έχει τιμές εύρους [0.8-0.9999]. Η καινούργια τιμή της θερμοκρασίας θα προκύψει έπειτα από πολλαπλασιασμό του συντελεστή ψύχρανσης με την τιμή θερμοκρασίας κατά την προηγούμενη επανάληψη. Όσο η θερμοκρασία μειώνεται τόσο δυσχαιρένει η δυνατότητα ελαχιστοποιήσης του κόστους λύσης. Ο αλγόριθμος ολοκληρώνεται είτε επειτα από ένα συγκεκριμένο αριθμό επαναλήψεων, είτε μετά από συγκεκριμένο χρονικό διάστημα το οποίο έχει οριστεί κατά την εκκίνηση του.

Κατα την διάρκεια της διαδικασίας της προσομοιωμένης ανόπτησης υπάρχουν πολλοί παράγοντες που μπορούν να ρυθμιστούν και να δοκιμαστούν, παράγοντες όπως ο καθορισμός της γειτονιας των εξετάσεων που θα πραγματοποιηθεί η αναζήτηση βέλτιστης λύσης, η μέθοδος με την οποία θα μειώνεται η τιμή της θερμοκρασίας ή ο αριθμός των εσωτερικών επαναλήψεων. Με την αποθήκευση της καλύτερης λύσης διασφαλίζεται ότι ο αλγόριθμος δεν θα τερματίσει σε πιθάνη εύρεση λύσης που αποτελεί τοπικό ακρότατο και θα συνεχίσει στην αναζήτηση ολικού ακρότατου. Στο σχήμα 4.1 περιγράφεται η διαδικασία της προσομοιώμενης ανόπτησης. Η συνάρτηση stop_function() ορίζει την σύνθηκη τερματισμού της διαδικασίας, ενώ η συνάρτηση is_feasible(), ελέγχει την εφικτότητα μίας υποψήφιας λύσης.

1: Εύρεση αρχικής λύσης s_0
2: Ορισμός αρχικής θερμοκρασίας $t_0 > 0$
3: $s_{best} = s_0$
4: Αρχικοποιήση παράγοντα ψύχρανσης a (συνήθως τιμή εύρους [0.8-0.99])
5: Αρχικοποίηση θερμοκρασίας διαδικασίας $t = t_0$
6: **while** !stop_function() **do**
7:     Επιλογή κατάστασης $s \in NS$, (όπου NS το σύνολο των λύσεων που προέκυψαν με βάση τους τελεστές γειτνίασης).
8:     **if** !is_feasible(s) **then**
9:         Επιστροφή στο βήμα 6:
10:     **end if**
11:     $DE = OB(s) - OB(s_0)$
12:     **if** $DE \leq 0$ and $OB(s) \leq OB(s_{best})$ **then**
13:         $s_{best} = s$
14:     **end if**
15:     **if** $DE \geq 0$ and $(random(0, 1) \geq e^{(-DE/t)})$ **then**
16:         $s_0 = s$
17:     **end if**
18:     $t = t * a$
19: **end while**

Σχήμα 4.1: Ψευδοκώδικας διαδικασίας προσομοιωμένης ανόπτησης

4.3 Περιγραφή αλγορίθμου αναρρίχησης λόφων

Η διαδικασία της αναρρίχησης λόφων αποτελεί διαδικασία βελτιστοποιήσης όπως και η διαδικασία της προσομοιωμένης ανόπτησης. Ο αλγόριθμος ανήκει στην κατηγορία αλγορίθμων τοπικής αναζήτησης. Ο αλγόριθμος εναλλάσεται από λύση σε λύση, με σκοπό την βελτίωση της τρέχουσας λύσης. Τα προβλήματα για τα οποία πραγματοποιεί βελτιστοποιήση ο αλγόριθμος χωρίζονται σε δύο κατηγορίες, τα προβλήματα ελαχιστοποιήσης και τα προβλήματα μεγιστοποιήσης. Για την παραγωγή υποψήφιων λύσεων, χρησιμοποιούμε τους τελεστές γειτνίασης. Ο αλγόριθμός μεταβάνει σε μία υποψήφια λύση μόνο αν ελαχιστοποιεί το κόστος της καλύτερης λύσης. Κάθε λύση η οποία εξετάζεται ως υποψήφια, συγκρίνεται και με την βέλτιστη λύση που έχει προκύψει εως την συγκεκριμένη επανάληψη, δίνοντας την δυνατότητα στον αλγόριθμο να ξεφύγει από τοπικά άκρα, κατά την αναζήτηση του ολικού άκρου, παρόμοια και με τον αλγόριθμο της προσωμοιωμένης ανόπτησης. Η αντικειμενική συνάρτηση είναι η ίδια που χρησιμοποιήθηκε στον αλγόριθμο της προσομοιωμένης ανόπτησης, και υπολογίζει το κόστος της λύσης. Ο αλγόριθμός εκτελείται επαναληπτικά μέχρι να ικανοποιήθει κάποια συνθήκη τερματισμού που έχει οριστεί. Αν δεν υπάρχει κάποια συνθήκη τερματισμού η οποία έχει οριστεί κατά την εκκίνηση του αλγορίθμου.

Το πρόβλημα του χρονοπρογραμματισμού εξετάσεων απότελει ένα πρόβλημα ελαχιστοποίησης. Σκοπος της βελτιστοποίησης για το πρόβλημα μας, είναι η μείωση του κόστους της λύσης. Η δημιουργία υποψήφιων λύσεων πραγματοποιήται με την βοήθεια των τελεστών γειτνίασης. Η επιλογή μίας λύσης πραγματοποιείται με τυχαίο τρόπο, ενώ μία υποψήφια λύση αντικαθιστά την τρέχουσα λύση σε περίπτωση μείωσης του κόστους λύσης. Στο σχήμα 4.2 παρουσιάζονται τα βήματα της διαδικασίας της αναρρίχησης λόφων. Ως stop_function(), ορίζουμε την συνάρτηση τερματισμού του προβλήματος, ενώ η συνάρτηση OB, αποτελεί την αντικειμενική συνάρτηση και υπολογίζει την τιμή κόστος μίας λύσης.

Δημιουργία αρχικής λύσης s_0
while stop_function() **do**
    Παραγωγή καταστάσεων $s \in NS$ (σύνολο υποψήφιων λύσεων)
    if $OB(s) \leq OB(s_0)$ **then**
        $s_0 = s$
    end if
end while

Σχήμα 4.2: Αλγόριθμος αναρρίχησης λόφων

4.4 Τελεστές γειτνίασης

Οι τελεστές γειτνίασης χρησιμοποιούνται για την παραγωγή υποψήφιων λύσεων, κατά την εκτέλεση των διαδικασιών βελτιστοποιήσης, οι οποίες έπειτα από αξιολόγηση μπορούν να αντικαταστήσουν την τρέχουσα λύση. Οι τελεστές γειτνίασεις πραγματοποιούν μεταβολές στις περιόδους των εξετάσεων, με σκοπό την δημιουργία νέων υποψήφιων λύσεων. Στην τρέχουσα επίλυση χρησιμοποιήθηκαν 7 τελεστές γειτνίασης.

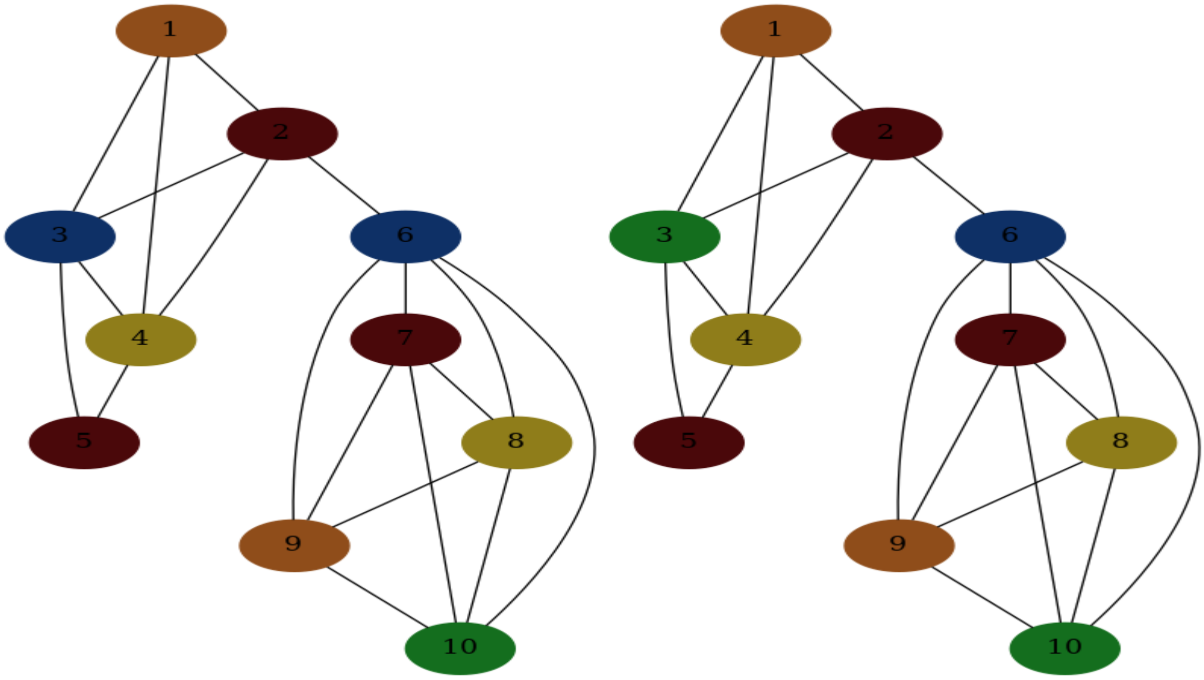
- Μετακίνηση περιόδου εξετάσης
- Εναλλαγή περιόδων εξετάσεων
- Ανταλλαγή εξετάσεων μεταξύ περιόδων
- Ολίσθηση εξετάσεων ανά περίοδο
- Αλυσίδες Kempe
- Εξαγωγή εξάτασης
- Διπλή εξαγωγή εξέτασης

Σε κάθε βήμα επιλέγεται μία τυχαία εξέταση $e \in X$. Για να επιλεχθεί μία εξέταση πρεπει να έχει γειτονικές εξετάσεις και να συμμετέχει στην αντικειμενική συνάρτηση, να υπάρχει δηλαδή τουλάχιστουν μία γειτονική εξέταση, οπου οι περίοδοι τους έχουν διαφορά κατά απόλυτη τιμή εως 5 περιόδους. Επίσης απαραίτητη προυπόθεση είναι η εφικτότητα της κίνησης.

4.4.1 Μετακίνηση περιόδου εξετάσης

Επιλογή μίας τυχαίας περιόδου p και μετακίνηση της εξέτασης e στην περίοδο p. Για να πραγματοποιηθεί η μετακίνηση της εξέτασης e στην περίοδο p, πρέπει να ισχύουν οι εξής περιορισμοί:

- $p \neq F(e)$
- Η κίνηση που θα εκτελεστεί να είναι εφικτή.



Σχήμα 4.3: Μετακίνηση εξέτασης 3 σε διαφορετική περίοδο

4.4.2 Εναλλαγή περιόδων εξετάσεων

Για κάθε τυχαία εξέταση e, επιλέγεται τυχαία μία διαφορετική εξεταση $e_2 \in X$ και στην συνέχεια πραγματοποιήται εναλλαγή των περιόδων τους. Οι περιορισμοί που θα πρέπει να ισχύουν για να εκτελεστεί η κίνηση είναι οι εξής:

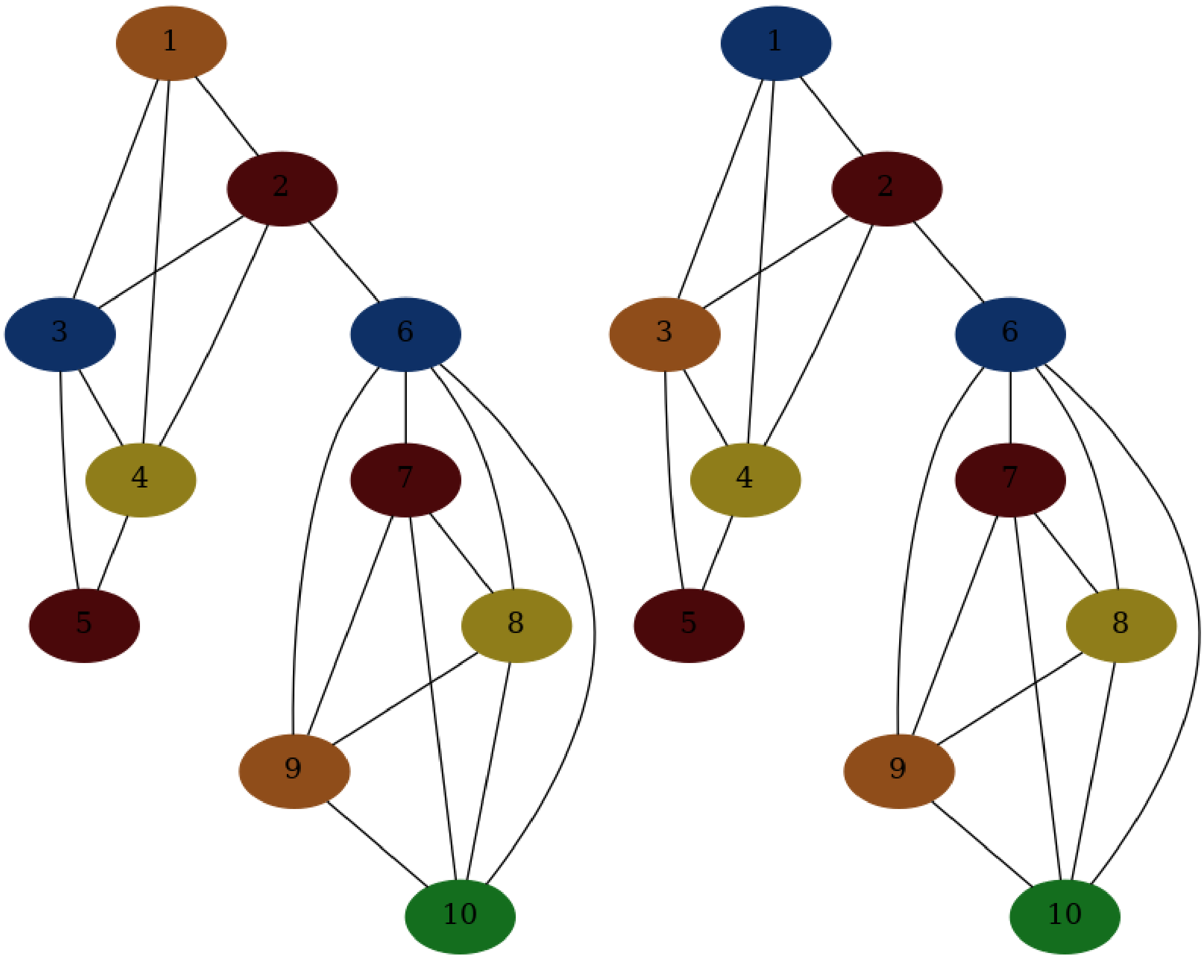
- Για κάθε εξέταση e η οποία επιλέγεται πρέπει να ισχύει:

$$\mathbb{W}_{e_1 e_2} \neq 0 \tag{4.2}$$

- Οι δύο εξετάσεις που επιλέχθηκαν θα πρέπει να συμμετέχουν στην αντικειμένική συνάρτηση και κατ᾽ επέκταση στο κόστος.

$$\left| F_{e_1} - F_{e_2} \right| <= 5 \tag{4.3}$$

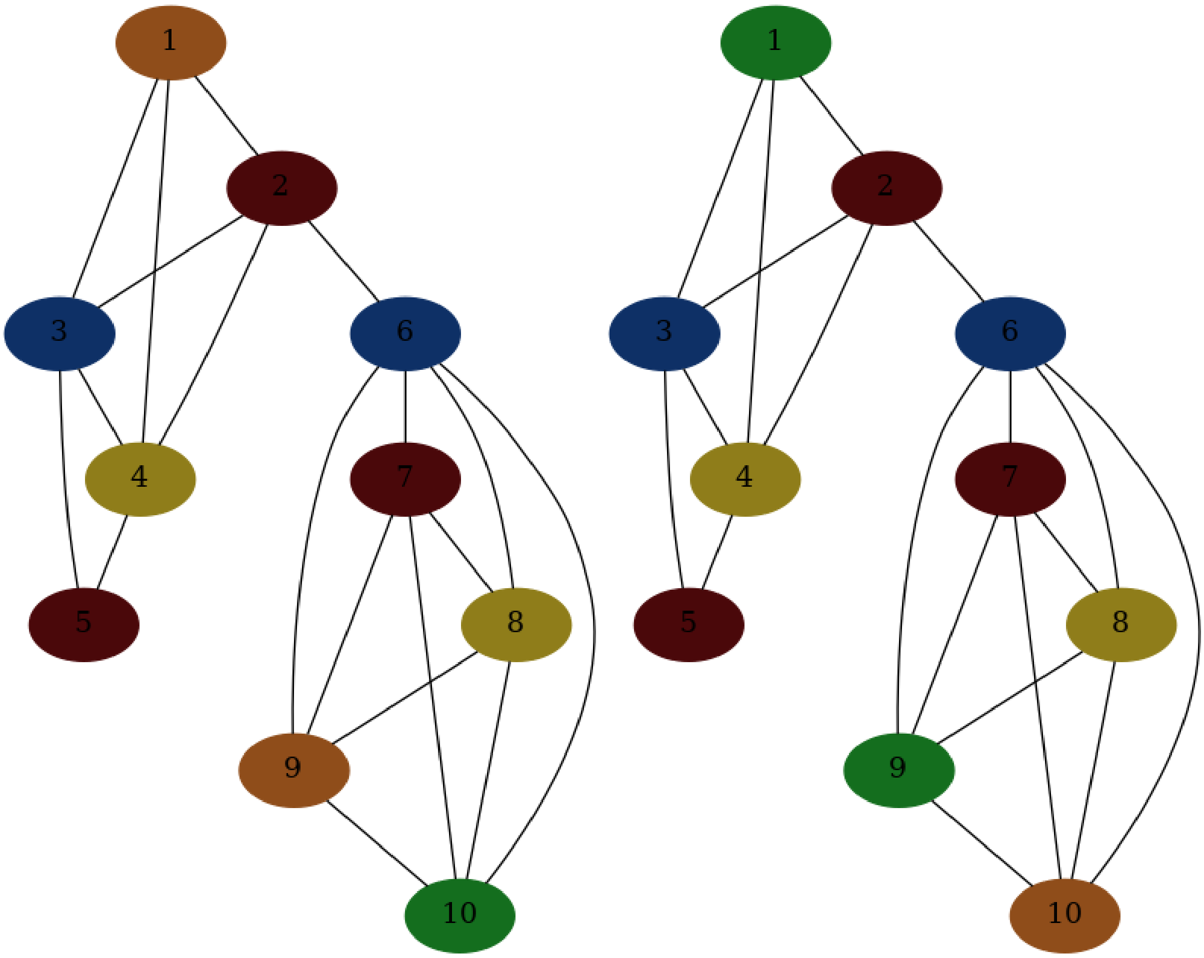
- Η κατονομή των περιόδων στις εξετάσεις πρέπει να είναι εφικτή



Σχήμα 4.4: Ανταλλαγή περιόδων μεταξύ εξετάσεων 1 και 3

4.4.3 Ανταλλαγή εξετάσεων μεταξύ περιόδων

Σε κάθε εκτέλεση της κίνησης,επιλέγονται τυχαία δύο περίοδοι p_i, p_j και κάθε εξέταση που είχε προγραμματιστεί στην περίοδο p_i, προγραμματίζεται εκ νέου στην περίοδο p_j και αντίστοιχα οι εξετάσεις που προγραμματίστηκαν στην περίοδο p_j, προγραμματίζονται εκ νέου στην περίοδο p_i. Αξιοσημείωτο είναι το γεγονός ότι η συγκεκριμένη κίνηση επιτυγχάνει να δημιουργήσει χαμηλότερου κόστους υποψήφιες λύσεις κατά τα αρχικά στάδια της διαδικασίας βελτιστοποίησης. Όσο οι μεταβολλές των καταστάσεων αυξάνονται, η κίνηση αποτυγχάνει να δημιουργήσει εφικτές καταστάσεις $s_n \in N(s)$, καθώς μειώνεται η πιθανοτητα εφικτής ανταλλαγής περιόδων ανάμεσα σε δύο σύνολα εξετάσεων.



Σχήμα 4.5: Εναλλαγή περιόδων (Πορτοκαλί-Πράσινο)

4.4.4 Ολίσθηση εξετάσεων ανά περίοδο

Σε κάθε εκτέλεση του τελεστή γειτνίασης, επιλέγονται τυχαία δύο περίοδοι p_i, p_j, όπου $p_i < p_j$. Ξεκινώντας από την περίοδο p_{i+1}, τα σύνολα εξετάσεων της κάθε περιόδου, προγραμματίζονται μία περίοδο πριν, κάτι που θα συμβεί μέχρι και την εξέταση p_j. Οι εξετάσεις που προγραμματίστηκαν στην περίοδο p_i θα προγραμματιστούν στην περίοδο p_j. Αντίστοιχα και με την κίνηση της ανταλλαγής εξετάσεων μεταξύ περιόδων, η κίνηση της ολίσθησης εξετάσεων ανα περίοδο, δεν θα έχει μεγάλη αποτελεσματικότητα, έπειτα από ένα συγκεκριμένο αριθμό επαναλήψεων που θα πραγματοποιηθούν στην διαδικασία βελτιστοποιήσης.

4.4.5 Αλυσίδες Kempe

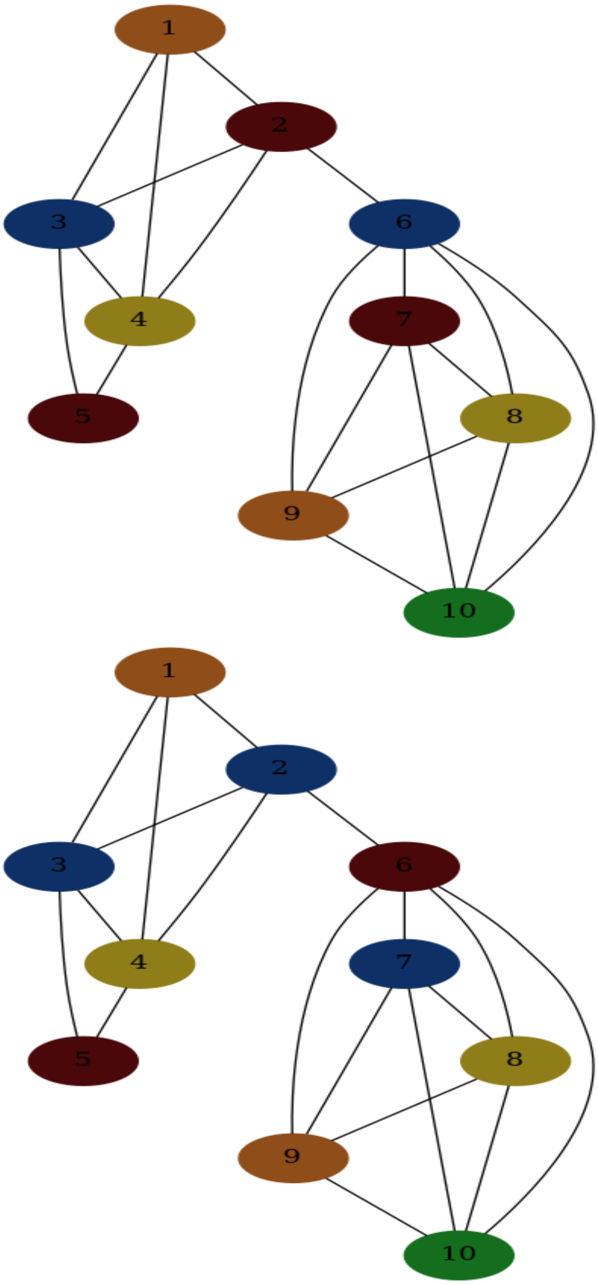
Σε ένα γράφο μία αλυσίδα kempe, αποτελειται από ένα σύνολο κορυφών συνδεδεμένες μεταξύ τους, για τις οποίες εξετάζοντας τις συσχετιζόμενες κορυφές αναδρομικά, παρατηρείται εναλλαγή μεταξύ συγκεκριμένων περιόδων (χρωμάτων). Σε μαθηματικούς όρους μία αλυσίδα kempe είναι μία συσκευή [15] που χρησιμοποιήται για την μελέτη του Θεωρήματος τεσσάρων χρωμάτων(four color theorem) [16]. Το θεώρημα των τεσσάρων χρωμάτων διατύπωνε την ιδέα ότι οποιοσδήποτε γράφος που αποτελείται από κόμβους και ακμές θα μπορούσε να χρωματιστεί με λιγότερα από τεσσερα χρώματα, με τέτοιο τρόπο ωστε δύο συνδεδεμένες κορυφές να έχουν διαφορετικό χρώμα. Το θεώρημα των τεσσάρων χρωμάτων αποτέλεσε το πρώτο σημαντικό θεώρημα που αποδυκνύεται με την χρήση υπολογιστή. Η υπόθεση του θεωρήματος των τεσσάρων χρωμάτων αναπτύχθηκε προτάθηκε αρχικά από τον φοιτητή Francis Guthrie, οποίος προσπάθησε να χρωματίσει τον χάρτη των περιφερειών της Αγγλίας. Μία απόδειξη του θεωρήματος δόθηκε από τον Alfred Kempe, το 1879 η οποία επικροτήθε εως ότου 11 χρόνια αργότερα, το 1890, παρουσιαστεί ως ανακριβής από τον Percy Heawood. Τελικά η απόδειξη kempe, κρίθηκε ως ακριβής το 1976 από τους Kenneth Appel και Wolfgang Haken από το πανεπιστήμιο του Ιλλινόις, με την βοήθεια του john koch σε ένα πείραμα διάρκειας 1200 ωρών.

Σαν τελεστής γειτνίασης στο πρόβλημα μας οι αλυσίδες kempe εκτελούν την εξής διαδικασία:
Επιλέγονται δύο εξετάσεις e_1, e_2, με χρονικές περιόδους $F(e_1), F(e_2)$. Οι δύο αυτές εξετάσεις έχουν έχουν προγραμματιστεί σε διαφορετικές χρονικές περιόδους. Εκτελείται εναλλαγή στη χρονική περίοδο των δύο αυτών εξετάσεων έτσι ώστε $e_1- > F(e_2)$ και $e_2- > F(e_1)$. Στην συνέχεια ελέγχονται όλες οι γειτονικές εξετάσεις της εξέτασης e_2. Για κάθε γειτονική εξέταση n_i, της κορυφής e_1 όπου $F(n_i) = F(e_2)$, πραγματοποιήται εναλλαγή της περιόδου εξέτασης η οποία προγραμματίζεται στην περίοδο $F(e_1)$. Αντίστοιχα εξέταζονται και οι γειτονικές εξετάσεις της εξέτασης e_1 και οποίες από τις εξετάσεις, ανήκουν στην περίοδο $F(e_1)$, προγραμματίζονται εκ' νέου στην περίοδο $F(e_2)$, όπως παρουσιάζετε και στις εξισώσεις (4.4, 4.5).

- $$F(e_1) \qquad \forall n_i \text{ and } F(n_i) = F(e_2) \tag{4.4}$$

- $$F(e_2) \qquad \forall n_i \text{ and } F(e_1) = F(n_i) \tag{4.5}$$

Η διαδικασία εκτελείται επαναληπτικά ελέγχοντας για κάθε κορυφή που πραγματοποιήθηκε μεταβολή τις γειτονικές κορυφές της, εως ότου δεν υφίσταται η δυνατότητα να πραγματοποιηθούν εναλλαγές μεταξύ των περιόδων $F(e_1), F(e_2)$. Στο σχήμα 4.1, παρουσιάζεται η εκτέλεση μίας αλυσίδας kempe βήμα-βήμα. Στον γράφο επιλέγεται η κορυφή 6 και χρωματίζεται με το κόκκινο ενώ η κορυφή 2 με μπλε. Αυτή η εναλλαγή επηρεάζει και τις κορυφές 7 και 2 που θα χρωματιστούν με χρώμα μπλε. Εξαιτίας της κορυφής 2 επηρεάζεται η κορυφή 3 και χρωματίζεται με χρώμα κόκκινο, η οποία με την σειρά της επηρεάζει την κορυφή 5 που θα χρωματιστεί με χρώμα μπλε. Η αλυσιδωτή αυτή αντίδραση οδηγεί στην δημιουργία μίας αλυσίδας kempe.

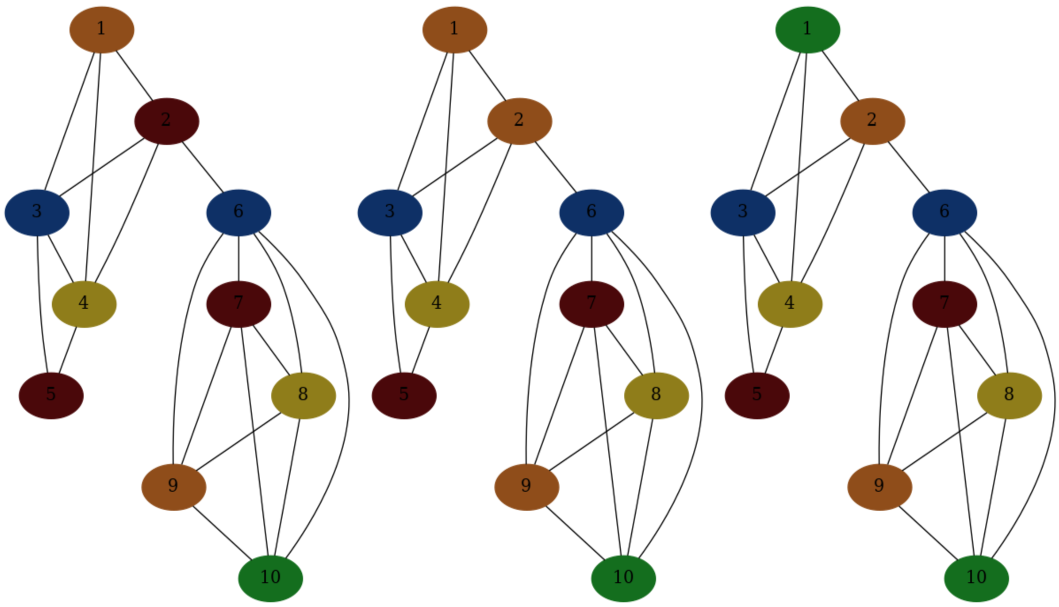


Σχήμα 4.6: Παράδειγμα αλυσίδας kempe (Κόκκινο-μπλε χρώμα)

4.4.6 Εξαγωγή εξέτασης

Επιλογή δύο εξέτασεων $e1, e2 \in V$. Στην εξέταση $e1$ ανατείθεται η περίοδος της εξέτασης e2 ενώ στην εξέταση $e2$ επιλέγεται η κατάλληλη παρίοδος στην οποία μπορεί να προγραμματιστεί η εξέταση. Όπως αναφέρεται και στο άρθρο [17], οι περιορισμοί που πρέπει να ισχύουν για να εκτελεστεί η κίνηση είναι οι εξής:

- $W_{e1e2} \neq 0$
- $F_{e1} \neq F_{e2}$
- $|F_{e1} - F_{e2}| \leq 5$
- Οι κινήσεις που θα πραγματοποιηθούν πρέπει να είναι εφικτές.

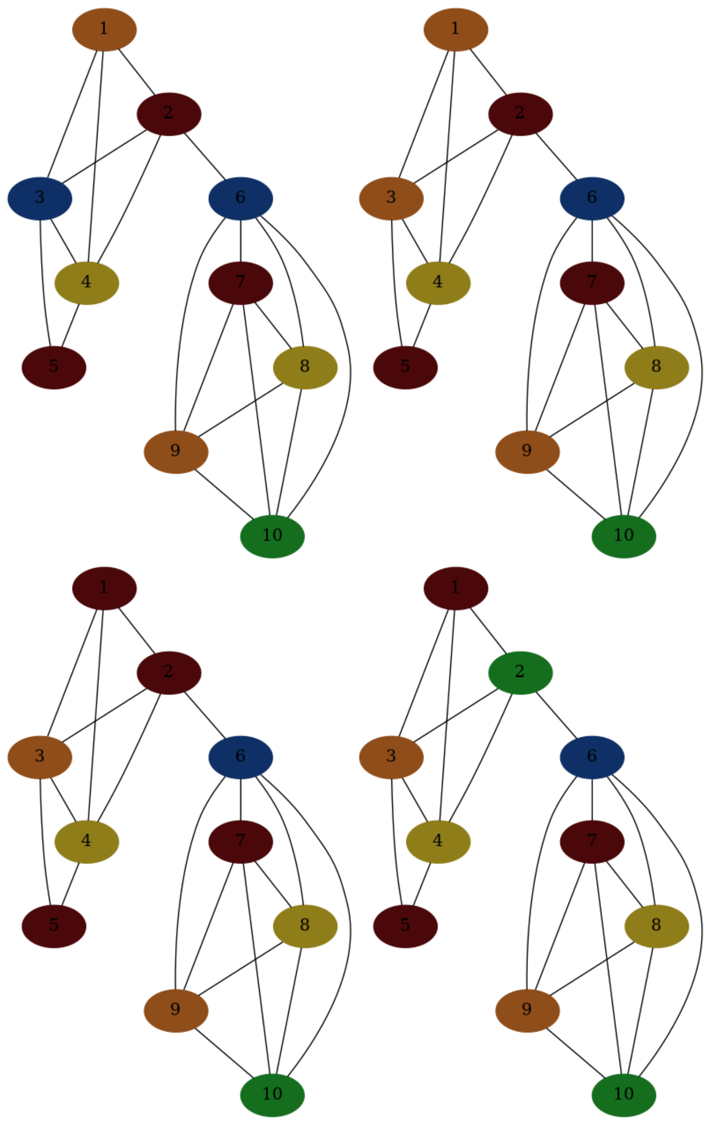


Σχήμα 4.7: Εκτέλεση κίνησης εξαγωγής εξέτασης μεταξύ εξετάσεων 2, 1

4.4.7 Διπλή εξαγωγή εξέτασης

Η κίνηση της διπλή εξαγωγής εξετάσεων, αποτελεί επέκταση της ιδέας της κίνησης της εξαγωγής εξετάσεων καθώς επεκτείνει τον όγκο των κινήσεων οι οποίες θα πραγματοποιηθούν. Η ακολουθία εκτέλεσης είναι παρόμοια με την απλή εξαγωγή εξέτασης, με την σημαντική διαφορά της εισαγωγής μίας τρίτης εξέτασης στα απαιτούμενα βήματα εκτέλεσης. Συνοπτικά επιλέγονται τρεις εξετάσεις $e1, e2, e3 \in X$. Η εξέταση $e1$ θα μετακινηθεί στην περίοδο F_{e2}, η εξέταση $e2$ θα μετακινηθεί στην περίοδο F_{e3}, και θα πραγματοποιηθεί μετακίνηση σε μία εφικτή περίοδο p της εξέτασης e3. Όπως περιγράφεται και στο άρθρο [17], οι περιορισμοί που πρέπει να ισχύουν για να εκτελεστεί η κίνηση είναι οι εξής:

- $W_{e1e2} \neq 0$ and $W_{e2e3} \neq 0$
- $|F_{e1} - F_{e2}| <= 5$
- Οι κινήσεις που θα πραγματοποιηθούν πρέπει να είναι εφικτές.



Σχήμα 4.8: Εκτέλεση κίνησης διπλής εξαγωγής εξέτασης για τις εξετάσεις 3,1,2

4.4.8 Διαδικασίες βελτιστοποιήσης

Εκτός από τους τελεστές γειτνίασης χρησιμοποιούνται και ορισμένες διαδικασίες με σκοπό την μείωση του κόστους λύσης ενός προβλήματος. Η πρώτη διαδικασία γενικεύει τις κινήσεις των τελεστών γειτνίασης εξετάζοντας για κάθε τελεστή γειτνίασης την επιρροή που έχει ξεχωριστά στην κάθε εξέταση του προβλήματος. Σε πιθανή περίπτωση μείωσης κόστος της λύσης η εξέταση προγραμματίζεται εκ' νέου σε νέα περίοδο. Η διαδικασία εκτελείται μέχρι την λήξη ενός συγκεκριμένου χρονικού ορίου το οποίο ορίζει ο χρήστης. Άλλη μία διαδικασία βελτιστοποιήσης που μπορεί να χρησιμοποιηθεί είναι η εφαρμογή ενός ελαστικού περιορισμού (soft constraint), ο οποίος ορίζει ότι εξετάσεις με υψηλό παράγοντα λαμβάνουν προτεραιότητα προγραμματισμού. Ως παράγοντα ορίζουμε τον αριθμό των σπουδαστών που συμμετέχουν σε μία εξέταση. Παραμετροποιώντας τον συγκεκριμένο περιορισμό προγραμματίζουμε εξετάσεις οι οποίες έχουν υψηλό αριθμό κοινών σπουδαστών προγραμματίζονται με κατάλληλο τρόπο ώστε να έχουν την μικρότερη δυνατή συμμετοχή η στην βέλτιστη περίπτωση καμία συμμετοχή στο κόστος.

Συνοπτικά στον πίνακα 4.1 παρουσιάζεται ο συνολικός αριθμός των εξετάσεων που επηρεάζει η κάθε κίνηση. Οι κινήσεις που περιλαμβάνουν ανταλλαγές με βάση την περίοδο διαφέρουν ανάλογα τον όγκο του προβλήματος και την περίοδο που θα επηρεαστεί από την κίνηση. Μία ιδέα που έχει προταθεί στο άρθρο [17] είναι ο ορισμός μικρής πιθανότητας εκτέλεσης της κίνησης της διπλής εξαγωγής εξέτασης, καθώς αποτελεί κίνηση που ικανοποιήται από μικρό συνδυασμό εξετάσεων λόγω της πολυπλοκότητας των περιορισμών της. Επίσης η αντιστοίχηση πιθανοτήτων εκτέλεσης σε κάθε κίνηση είναι μία ιδέα που έχει προταθεί ξανά στο άρθρο [17]. Και οι δύο προτάσεις που αναφέρθηκαν μπορούν να επηρεάσουν την απόδοση των τελεστών γειτνίασης καθώς και την τελική λύση.

Αξίζει να σημειωθεί ότι κινήσεις που αλληλεπιδρούν με περιόδους και όχι συγκεκριμένες εξετάσεις επηρεάζουν διαφορετικό αριθμό εξετάσεων ανά εκτέλεση τους, αριθμός ο οποίος εξαρτάται από την τεχνική χρωματισμού που έχει εμφανιστεί καθώς και από κινήσεις βελτιστοποιήσης που εφαρμόστηκαν εως και την εκτέλεση του κάθε τελεστή.

Κίνηση	**Αριθμός εξετάσεων που συμμετέχουν**
Μετακίνηση εξέτασης	1
Ανταλλαγή εξετάσεων	2
Ανταλλαγή περιόδων	Διαφέρει ανά κίνηση
Ολίσθηση περιόδων	Διαφέρει ανά κίνηση
Αλυσίδες Kempe	Διαφέρει ανά κίνηση
Εξαγωγή εξέτασης	2
Διπλή εξεγωγή εξέτασης	3

Πίνακας 4.1: Εξετάσεις που επηρεάζονται ανά κίνηση

Κεφάλαιο 5

Υλοποίηση αλγορίθμων βελτιστοποιήσης

Η δημιουργία των μεταευρετικών αλγορίθμών πραγματοποιήθηκε με χρήση της γλώσσας προγραμματισμού python, όπως και η δημιουργία των βασικών οντοτήτων του προβλήματος.

5.1 Βασικές οντότητες προβλήματος

Οι τρεις βασικές οντότητες που χρησιμοποιήθηκαν είναι οι οντότητες Exam,Student,Problem. Η οντότητα Exam μοντελοποιεί μία εξέταση. Η αναπαράσταση των οντοτήτων πραγματοποιήται με αντικειμενοστραφής τρόπους σχεδίασης και συγκεκριμένα την χρήση κλάσεων. Οι πληροφορίες που αποθηκεύονται στην κλάση Exam, είναι το αναγνωριστικό του μαθήματος, καθώς και οι φοιτητές που συμμετέχουν. Αντίστοιχα η κλάση Student, μοντελοποιεί έναν φοιτητή αποθηκεύοντας πληροφορίες χρησιμοποιώντας ως αναγνωριστικό τον αύξοντα αριθμό του εκάστοτε φοιτητή, που προκύπτει με βάση την σειρά ανάγνωσης από το αρχείο δεδομένων του προβλήματος. Επίσης αποθηκεύονται με την χρήση μίας λίστας, οι εξετάσεις τις οποίες συμμετέχει ένας φοιτητής. Σημείο ανοφοράς για την κλάση Exam, αποτελεί η συνάρτηση common_students(Student other), η οποία βρίσκει τους κοινούς σπουδαστές μεταξύ δύο στιγμυοτύπων εξετάσεων και θα χρησιμοποιηθεί για την εύρεση των βαρών των κορυφών, όταν μοντελοποιήσουμε το προβλήμα με την χρήση γράφήματος.

Η τρίτη κλάση που έχει και την μεγαλύτερη επιρροή στο πρόβλημα είναι η κλάση Problem, η οποία αποσυνθέτει το πρόβλημα μετατρέπωντας το σε γράφο. Η κλάση αυτή αποθηκεύει πληροφορίες σχετικά με τα μαθήματα, τους σπουδαστές, τον συνολικό αριθμό εγγραφών, και των αριθμό των περιόδων που χρησιμοποιήθηκαν σε κάθε πρόβλημα. Με χρήση της μεθόδου create _graph(), κατασκευάζεται ο γράφος που θα μοντελοποιήσει το πρόβλημα. Ο κώδικας της συγκεκριμένης μεθόδου παρουσιάζεται στο σχήμα 5.1. Για την κατασκευή του γράφου χρησιμοποιήθηκε η βιβλιοθήκη της networkx, η οποία αποτελεί ιδανική επιλογή για διαχείριση γράφων.

Βασική κλάση που κατασκευάστηκε είναι η κλάση solution, η οποία δημιουργεί αντικείμενα τα οποία θα διαχειρίζονται ένα πρόβλημα για το οποίο μέσω μεθόδων θα μπορουμε να διαχειριζόμαστε το πρόβλημα καθόλη την διαδικασία βελτιστοποιήσης του και να διαχειριζόμαστε τις μεταβολές στην λύση του προβλήματος, υπολογίζοντας και τις επιρροές στο συνολικό κόστος της λύσης. Επίσης στην κλάση solution υλοποιούνται οι επτά τελεστές γειτνίασης οι οποίοι ορίζονται ως μέθοδοι της κλάσης. Επίσης ορίζεται η μεταβλητή cost, η οποία θα μεταβάλλεται ανάλογα με την τιμή της αντικειμενικής συνάρτησης, και αποτελεί την αντικειμενική τιμή, καθώς και η δομή δεδομένων p_periods, που για κάθε περίοδο αποθηκεύει τις εξετάσεις οι οποίες έχουν προγραμματιστεί σε αυτήν. Τέλος ορίζονται οι μέθοδοι reposition,can_be_moved και select_move. Η μέθοδος reposition, επαναπρογραμματίζει μία εξέταση σε διαφορετική περίοδο υπολογίζοντας την μεταβολή που θα έχει η αλλαγή αυτή στο κόστος, η μέθοδος can_be_moved, ελέγχει την εγκυρότητα μίας κίνησης και η μέθοδος select_move, επιλέγει και εφαρμόζει τυχαία έναν τελεστή γειτνίασης.

```python
    def create_graph(self):
        self.G=nx.Graph()
        self.G.add_nodes_from([exam.id for exam in self.exams])
        for index_i in range(len(self.exams)):
            for index_j in range(index_i+1,len(self.exams)):
                cs=self.exams[index_i].common_students(self.exams[index_j])
                if cs>0:
                    self.G.add_edge(self.exams[index_i].id,self.exams[index_j].id,weight=cs)

```

Σχήμα 5.1: Κώδικας δημιουργίας γράφου με την χρήση της βιβλιοθήκης networkx

Στο σχήμα 5.2 παρουσιάζετε ο κώδικας, που χρησιμοποιήθηκε για την αφαίρεση των άνευ σημασίας εξετάσεων από το γράφο, διαδικασία η οποία μείωσε το μέγεθος του προβλήματος. Μαζί με την αφαίρεση τον άνευ σημασίας εξετάσεων πραγματοποιήται και εύρεση των εξετάσεων που παρουσιάζουν συμμετρία. Αυτές οι εξετάσεις θα προγραμματιστούν μία φορά, στην ίδια χρονική περίοδο.

```python
    def noise_out(self):
        self.Graph_copy=deepcopy(self.G)
        self.fixed_exams=dict()
        _,ident_type_2,_=self.identical_exams()
        for identical in ident_type_2:
            key=-1
            for index,node in enumerate(identical):
                if index==0:
                    key=node
                    self.ident_coloring_exams[key]=list()
                    continue
                self.ident_coloring_exams[key].append(node)
                self.fixed_exams.update({node:self.s_periods[key]})
        self.G.remove_nodes_from(self.noisy_exams)

```

Σχήμα 5.2: Κώδικας αφαίρεσης άνευ σημασίας εξετάσεων από τον γράφο

Επίσης κατασκευάστηκε η συνάρτηση compute_cost(), η οποία χρησιμοποιήται για τον υπολογισμό της αντικειμενικής τιμής μίας λύσης, αποτελώντας την αντικειμενική συνάρτηση του προβλήματος μας. Ο κώδικας για την μέθοδο υπολογισμού κόστους παρουσιάζεται στο σχήμα 5.3. Η συνάρτηση compute_cost(), υπολογίζει το συνολικό κόστος λύσης, ενώ η συνάρτηση compute_normalized_cost(), κανονικοποιεί το κόστος ανα σπουδαστή.

```
    def compute_cost(self):
        return sum([penalty[abs(self.s_periods[node1]-self.s_periods[node2])-1] for node1,node2 in self.G.edges])

    def compute_normilized_cost(self):
        return self.compute_cost()/self.normalized

```

Στο σχήμα 5.3, παρουσιάζεται ο κώδικας εύρεσης των συμμετρικών εξετάσεων, μέσω της μεθόδου identical _exams(). Η πρώτη κατηγορία συμμετρικών εξετάσεων, αποθηκεύεται στην λίστα identical_type_1, αφορά ζεύγη εξετάσεων πού έχουν τις ίδιες γειτονικές εξετάσεις, με την ίδια τιμή βάρους ανά σύνδεση, χωρίς την ύπαρξη κοινών εξετάσεων μεταξύ τους. Η δεύτερη κατηγορία συμμετρικών εξετάσεων που υπολογίζεται και αποθηκεύεται στην λίστα identical_type_2, αφορά τις εξετάσεις που που έχουν τις ίδιες γείτονικές εξετάσεις, με ίδια τιμή βάρους ανά σύνδεση, έχοντας κοινό αριθμό φοιτητών μεταξύ τους, ενώ η τρίτη κατηγορία αφορά τις εξετάσεις που έχουν κοινούς σπουδαστές, έχουν τις ίδιες γειτονικές εξετάσεις, ωστόσο δεν έχουν οι συνδέσεις τους με τις γειτονικές εξετάσεις δεν παρουσιάζουν απόλυτη ομοιότητα. Οι τρεις αυτές λίστες διατηρούν ως πληροφορία συνόλα εξετάσεων τα οποία ανήκουν στην εκάστοτε κατηγορία.

```
     def identical_exams(self):
        exam_combinations=combinations([exam.id for exam in self.exams],2)
        identical_type_1,identical_type_2,identical_type_3=list(),list(),list()
        type_1,type_2,type_3=list(),list(),list()
        for node_1,node_2 in exam_combinations:
            if self.exams[self.exams.index(node_1)].students==self.exams[self.exams.index(node_2)].students:
                type_1.append((node_1,node_2))

            node_a_neighbors=set(self.G.neighbors(node_1))
            node_b_neighbors=set(self.G.neighbors(node_2))
            if len(node_a_neighbors)==0 and len(node_b_neighbors)==0: continue
            ident_neighbor_exams=True
            if node_a_neighbors==node_b_neighbors:
                for neighbor in node_a_neighbors:
                    if self.G[node_1][neighbor]['weight']!=self.G[node_2][neighbor]['weight']:
                        ident_neighbor_exams=False
                        break
                if ident_neighbor_exams:
                    type_2.append((node_1,node_2))

            elif node_a_neighbors.symmetric_difference(node_b_neighbors)=={node_1,node_2}:
                ident_ipp=True
                for neighbor in node_a_neighbors:
                    if neighbor!=node_2:
                        if self.G[node_1][neighbor]['weight']!=self.G[node_2][neighbor]['weight']:
```

```python
                            ident_ipp=False
                if ident_ipp:
                    type_3.append((node_1,node_2))

        for node_1,node_2 in type_1:
            found_in=False
            for index,fs in enumerate(identical_type_1):
                    if node_1 in fs or node_2 in fs:
                        identical_type_1[index].add(node_1)
                        identical_type_1[index].add(node_2)
                        found_in=True
                        break

            if not found_in:
                identical_type_1.append({node_1,node_2})

        for node_1,node_2 in type_2:
            found_in=False
            for index,fs in enumerate(identical_type_2):
                if node_1 in fs or node_2 in fs:
                    found_in=True
                    identical_type_2[index].add(node_1)
                    identical_type_2[index].add(node_2)
                    break
            if not found_in:
                identical_type_2.append({node_1,node_2})

        for node_1,node_2 in type_3:
            found_in=False
            for index,fs in enumerate(identical_type_3):
                if node_1 in fs or node_2 in fs:
                    found_in=True
                    identical_type_3[index].add(node_1)
                    identical_type_3[index].add(node_2)
                    break
            if not found_in:
                identical_type_3.append({node_1,node_2})
        return identical_type_1,identical_type_2,identical_type_3
```

Σχήμα 5.3: Κώδικας εύρεσης συμμετρικών εξετάσεων

5.2 Προσομοιωμένη ανόπτηση

Στο σχήμα 5.4, παρουσιάζεται η διαδικασία της προσομοιωμένης ανόπτησης υλοποιημένη με την βοήθεια της γλώσσας python. Για την εκτέλεση της διαδικασίας κατασκευάστηκε μία συνάρτηση η οποία δεχόμενη ως όρισμα ένα αλφαριθμητικό που αντιστοιχεί στο όνομα ενός προβλήματος, δημιουργεί το πρόβλημα με κατασκευή αντικειμένου της κλάσης Problem και εκτελεί την διαδικασία της προσομοιωμένης ανόπτησης. Η μεταβλητή temp αντιστοιχεί στην τιμή θερμοκρασίας, η μεταβλήτη alpha, στον συντελεστή ψύχρανσης και η μεταβλητή freeze στην θερμοκρασία ψύξης. Με βάση τον τελεστή ψύξης θα πραγματοποιήθει και η μειώση της θερμοκρασίας. Ορίζουμε επίσης και την μεταβλητή temp_delay_counter, η οποία συμμετέχει στην μείωση της θερμοκρασίας, και ορίζει την καθυστέρηση στην διαδικασία μείωσης που θα προκύψει όταν έχει βρεθεί καλύτερη λύση στο πρόβλημα, καθυστερόντας την μείωση της θερμοκρασίας και δοκιμάζοντας αναζήτηση λύσεων με την ίδια τιμή θερμοκρασίας για έναν συγκεκριμένο αριθμό επαναλήψεων. Επίσης σε κάθε κύκλο επανάληψης όπου έχει πραγματοποιήθηκε μείωση του κόστους, επαναφέρουμε την θερμοκρασία σε μία σταθερή τιμή και χωρίς να μεταβούμε σε επόμενο κύκλο επανάληψης. Ως κύκλο επανάληψης ορίζουμε ένα αριθμό επαναλήψεων οι οποίες εκτελέστηκαν εως την μείωση της τιμής της θερμοκρασίας από την αρχική της σε τιμή χαμηλότερη η ίση της τιμής της θερμοκρασίας ψύξης.

Η μεταβλητή temp επαναρχικοποιήται συγκεκριμένα με την σταθερή τιμή πέντε, συνεχίζοντας την διαδικασία στον ίδιο κύκλο επανάληψης. Η μεταβλητή delta κρατάει την διαφορα στο κόστος που προέκυψε ανάμεσα στην υποψήφια και την τρέχουσα λύση. Αν η διαφορά είναι μικρότερη του μηδέν η υποψήφια λύση γίνεται αποδεκτή, αντικαθιστώντας την προηγούμενη τρέχουσα λύση. Η μεταβλητή plateu, ελέγχει τον ρυθμό βελτίωσης ανά κύκλο επανάληψης. Αν σε έναν κύκλο επανάληψης δεν έχει πραγματοποιηθεί καμία βελτίωση του κόστους λύσης, η μεταβλητή plateu αυξάνει την τιμή της κατά μία μονάδα. Αν δεν παρατηρηθεί μείωση κόστος λύσης και έχουν ολοκληρωθεί 20 κύκλοι επανάληψης, ο αλγόριθμός τερματίζει την λειτουργεία του. Τέλος, ως συνθήκη τερματισμού έχει οριστεί ένα πεπερασμένο χρονικό όριο. Η διαδικασία τερματίζει σε περίπτωση που ο συνολικός χρόνος εκτέλεσης της διαδικασίας υπερβαίνει το συγκεκριμένο χρονικό όριο. Η μεταβλητή που αποθηκεύει την τιμή του χρονικού ορίο είναι η μεταβλητη exec_time. Αξίζει να αναφερθεί ότι για την εμφάνισει των μηνυμάτων κατά της εκτέλεση του αλγορίθμου της προσομοιωμένης ανόπτησης χρησιμοποιήθηκε το πακέτο της python logging, που παρέχει μεθόδους για ιεραρχική εμφάνιση μηνυμάτων [18].

```
    def simulated_annealing(exec_time,dataset):
        logging.basicConfig(level=logging.INFO,format='%(asctime)s \t %(message)s')
        temp=1000
        start_temp=1000
        alpha=0.9999
        freeze=0.0001
        plateu=0
        solution_improvement_made=False
        temp_delay_counter=DEF_DATA.BEST_SOL_ITERATIONS
        s=solution(dataset)
        best=s.cost
        best_sol=s.solutions
        start_timer=time()
        number_of_reheats=3
        best,best_sol=s.eject_vertices(best,best_sol,start_time=start_timer)
        while True:
            moves=s.select_move()
            if len(moves)==0:
                continue
            pcost=s.cost
            rollback=dict()
            for exam in moves:
                rollback[exam]=s.solutions[exam]
            s.reposition(-1,-1,moves)
            delta=s.cost-pcost
```

```
            if delta<0:
                if s.cost<best:
                    best=s.cost
                    best_sol=deepcopy(s.solutions)
                    plateu=0
                    solution_improvement_made=True
                    logging.info(f"Simulated Annealing|New best solution found:S={best} T={temp}")
            elif delta>0:
                acceptance_propability=math.exp(-delta/temp)
                if acceptance_propability>random.random():
                    pass
                else:
                    s.reposition(-1,-1,rollback)

            if temp_delay_counter>0:
                temp_delay_counter-=1
            else:
                temp*=alpha
            if temp<freeze:
                if solution_improvement_made==True and number_of_reheats>0:
                    solution_improvement_made=False
                    number_of_reheats-=1
                    temp=DEF_DATA.REHEAT_TEMPERATURE
                    logging.info('Simulated Annealing|Reheating temperature-New value T={}'.format(temp))
                    continue

                # previous_best=best
                # best,best_sol=s.depth_moves(best,best_sol)
                # if previous_best>best:
                #     logging.info('Simulated Annealing|New best solution found S={} T={}'.format(best,temp))
                #     plateu=0
                #     continue
                plateu+=1
                number_of_reheats=3
                if plateu==10:
                    logging.info('Simulated Annealing| After {plateu} iterations no improvement made,canceling procedure')
                    break
            if time()-start_timer>exec_time:
                break
        s.reposition(-1,-1,best_sol)
        logging.info('Simulated Annealing| After {} seconds of execution best solution found S={}'.format(exec_time,s.cost))
        s.renew_solution(best_sol)

```

Σχήμα 5.4: Διαδικασία προσομοιωμένης ανόπτησης

5.3 Αναρρίχηση λόφων

Στο σχήμα 5.5, παρουσιάζεται η διαδικασία της αναρρίχησης λόφων με την βοήθεια της γλώσσας python. Με χρήση της μεθόδου execute_moves, πραγματοποιήται δημιουργία μίας υποψήφιας λύσης με την χρήση τελεστή γειτνίασης, η οποία και θα εκτελεστεί. Αν το κόστος της υποψήφιας λύσης είναι μεγαλύτερο του κόστους της βέλτιστης λύσης, η υποψήφια λύση απορρίπτεται. Σε διαφορετική περίπτωση μεταβαίνουμε στην υποψήφια λύση ως λύση του προβλήματος μας. Αντίστοιχα και με την διαδικασία της προσομοιωμένης ανόπτησης ως συνθήκη τερματισμού ορίζεται η υπέρβαση ένος πεπερασμένου χρονικού ορίου. Τέλος με την βοήθεια της μεθόδου permuting_periods, πραγματοποιήται μία αναδιάταξη των περίοδων, κατανέμοντας τις εξετάσεις σε περιόδους, ώστε να έχουν την μικρότερη συμμετοχή στο κόστος της λύσης.

```
def hill_climbing(dataset,exec_time):
    logging.basicConfig(level=logging.INFO,format='%(asctime)s\t%(message)s')
    sol=psolution(dataset)
    best=sol.cost
    start_timer=time()
    logo=' Permuting Periods '
    print('-'*5+logo+'-'*5)
    best=sol.permuting_periods(best)
    print('-'*(len(logo)+10),end='\n\n')
    while True:
        moves=sol.execute_moves()
        if len(moves)==0:
            if time()-start_timer>exec_time:
                break
            continue
        rollback=dict()
        for exam in moves:
            rollback[exam]=sol.solutions[exam]
        sol.reposition(-1,-1,moves)
        if sol.cost<best:
            best=sol.cost
            logging.info("Hill Climbing|New best solution found S={} and time T={}`s".format(sol.cost,time()-start_timer))
        else:
            sol.reposition(-1,-1,rollback)
        if time()-start_timer>exec_time:
            break
    sol.renew_solution(sol.solutions)
```

Σχήμα 5.5: Διαδικασία αναρίχησης λόφων

Κεφάλαιο 6

Αποτελέσματα αλγορίθμων επίλυσης

Στους πίνακες 6.1, 6.2 παρουσιάζονται οι τελικές τιμές κόστους που θα προκύψουν για όλα τα προβλήματα και στα δύο σύνολα δεδομένων (itc,carter). Το υλικό που χρησιμοποιήθηκε για την εκτέλεση των πειραμάτων έχει τα εξής χαρακτηριστικά.

Επεξεργαστής	Πυρήνες	Νήματα	Μνήμη
Intel Xeon Processor (Skylake, IBRS)	32	32	32GB

Πίνακας 6.1: Χαρακτηριστικά υπολογιστικής μονάδας εκτέλεσης πειραμάτων

Ως χρόνος εκτέλεσης για κάθε πρόβλημα ορίστηκαν τα 1000 δευτερόλεπτα, ενώ διάφορα αποτελέσματα για διαφορετικούς χρόνους εκτέλεσης εμφανίζονται στον ιστοχώρο [19]. Τα καλύτερα αποτελέσματα που έχου επιτευχθει μέχρι και σήμερα βρίσκονται στην ιστοσελίδα [20].

6.1 Αποτελέσματα προσομοιωμένης ανόπτησης

Αρχείο δεδομένων	Κόστος Λύσης	Βέλτιστη λύση	Ποσοστιαία διαφορά
car92	116368(6.87)	4.24	62.02
car91	98103(5.32)	3.64	46.15
ear83	48823(43.39)	32.42	33.83
hec92	30360(10.75)	10.03	7.17
kfu93	82043(15.33)	12.8	16.5
lse91	34312(12.58)	9.77	28.76
pur93	253584(8.44)	4	111
rye93	128746(11,21)	7.84	42.89
sta83	95959(157.05)	157.03	0.012
tre92	45025(10.32)	7.59	35.96
uta92	100995(4.74)	2.95	60.67
ute92	73746(26.82)	24.76	8.31
yor83	47502(50.48)	34.4	46.74
ITC2007_1	14539(1.84)	0.71	159
ITC2007_2	10439(0.83)	0.12	591
ITC2007_3	57759(3.47)	1.27	173
ITC2007_4	73734(16.67)	10.83	53.92
ITC2007_5	21739(2.49)	0.18	1283
ITC2007_6	55435(7.009)	3.84	82.52
ITC2007_7	17832(1.29)	0.02	6350
ITC2007_8	13044(1.69)	0.05	3280
ITC2007_9	20002(32.05)	4.68	584
ITC2007_10	30139(21.29)	8.63	146
ITC2007_11	66325(4.05)	3.32	21.98
ITC2007_12	30631(18.53)	6.43	188

Πίνακας 6.2: Αποτελέσματα προσομοιωμένης ανόπτησης

6.2 Αποτελέσματα αναρρίχησης λόφων

Αρχείο δεδομένων	Κόστος Λύσης	Βέλτιστη λύση	Ποσοστιαία διαφορά(%)
car92	117019(6.91)	4.24	62.97
car91	100000(5.42)	3.64	48.90
ear83	49123(43.66)	32.42	34.66
hec92	32000(11.23)	10.03	11.96
kfu93	84981(15.88)	12.8	24.06
lse91	34792(12.76)	9.77	30.6
pur93	255438(8.5)	4	112.5
rye93	129801(11.3)	7.84	44.13
sta83	95961(157.05)	157.03	0.012
tre92	45031(10.32)	7.59	82.52
uta92	102100(4.81)	2.95	63.05
ute92	74000(26.9)	24.76	8.64
yor83	47618(50.60)	34.4	47.09
ITC2007_1	15515(1.96)	0.71	176.05
ITC2007_2	12312(0.98)	0.12	716.66
ITC2007_3	58002(3.48)	1.27	174.01
ITC2007_4	73739(16.67)	10.83	53.92
ITC2007_5	22121(2.53)	0.18	1305
ITC2007_6	56432(7.13)	3.84	85.67
ITC2007_7	18000(1.30)	0.02	6400
ITC2007_8	14004(1.81)	0.05	3520
ITC2007_9	22002(35.25)	4.68	6.53
ITC2007_10	30240(21.37)	8.63	147.62
ITC2007_11	66400(4.05)	3.32	21.98
ITC2007_12	32521(19.67)	6.43	205.9

Πίνακας 6.3: Αποτελέσματα αναρρίχησης λόφων

Κεφάλαιο 7

Επίλογος

7.1 Συμπεράσματα

Οληκληρώντας την παρούσα εργασία, αναλύσαμε το συνδυαστικό πρόβλημα του χρονοπρογραμματισμού εξετάσεων, παρουσιάζοντας δύο τεχνικές επίλυσης, την τεχνική τοπικής αναζήτησης της αναρρίχισης λόφων καθώς και την τεχνική καθολικής βελτιστοποιήσης της προσομοιωμένης ανόπτησης. Με την εισαγωγή των εξετάσεων και σπουδαστων οι οποίοι χαρακτηρίστηκαν ως άνευ σημασίας, επιτευχθηκέ μείωση του μεγέθους των προβλημάτων, ενώ προτάθηκε και ο τρόπος επίλυσης των προβλημάτων με χρήση των συνεκτικών τμημάτων, κάτι που οδήγησε στην διάσπαση προβλημάτων σε επιμέρους υποπροβλήματα. Τέλος εξετάσθηκαν οι τελεστές γειτνίασης που χρησιμοποιήθηκαν για την δημιουργία καινούργιων λύσεων και προτάθηκαν παραμετροποιήσεις στον αλγόριθμο της προσομοιωμένης ανόπτησης με σκοπό την βελτίωση της απόδοσης του αλγορίθμου, και αντίστοιχα και της αναρρίχισης λόφων ενώ παρουσιάτηκαν αποτελέσματα για τα προβλήματα των συνόλων δεδομένων carter και ITC, έπειτα από χρηση των δύο αλγορίθμων βελτιστοποιήσης.

7.2 Μελλοντικές επεκτάσεις

Σχετική μελλοντική επέκταση, η οποία θα μπορούσε να προταθεί στο συγκεκριμένο πρόβλημα είναι η ενσωμάτωση περισσότερων περιορισμών που προέρχονται από προβλήματα χρονοπρογραμματισμού εξετάσεων εκπαιδευτικών ιδρυμάτων, όπως ο περιορισμός των προτιμήσεων των εισηγητών μίας εξέτασης ή ο περιορισμός στον χώρο διεξαγωγής των εξετάσεων και αντίστοιχα περιορισμός όπως, η εξομάλυνση του προγράμματος των φοιτητών ώστε να αποφεύγεται η συμμετοχή τους σε ένα άναρχα κατανεμμημένο πρόγραμμα, δηλαδή να παραμετροποιείται και ο αντίστοιχος χρόνος τον οποίο θα χρειαστεί ένας σπουδαστής ανάμεσα σε ένα σύνολο εξετάσεων που θα του επιτρέψει να προετιμαστεί για όλες τις εξετάσεις στις οποίες θα επρόκειτο να συμμετέχει. Με χρήση παρόμοιων περιορισμών θα μπορούσαν να προκύψουν νέοι τελεστές γειτνίασης, η να πραγματοποιηθεί τροποποιήση των τελεστών γειτνίασης οι οποίοι παρουσιάστηκαν στο συγκεκριμένο πρόβλημα. Τέλος η κατασκευή ένος ολοκληρωμένου Framework, σε συνδυασμό με την ανάγκη οπτικοποιήσης του προβλήματος, θα μπορούσαν να οδηγήσουν στον σχεδιασμό μίας ολοκληρωμένης εφαρμογής με σκοπό την κατασκευή προγράμματος εξετάσεων με βάση τους περιορισμούς ένος εκπαιδευτικού ιδρύματος, κάτι που θα παραμετροποιήται και θα δημιουργήται μέσω της εφαρμογης. Στο πρόβλημα θα μπορούσαμε να προσθέσουμε και τις δυνατότητες του προγραμματισμού με περιορισμούς(constaint programming) στην διαδικασία βελτιστοποιήσης του προβλήματος, κάτι που δύναται στο μέλλον να οδηγήσει στην δημιουργία ενός προγράμματος εξετάσεων για κάποιο εκπαιδευτικό ίδρυμα.

Βιβλιογραφία

[1] M. W. Carter, G. Laporte, and S. Y. Lee, “Examination timetabling: Algorithmic strategies and applications,” *Journal of the operational research society*, vol. 47, no. 3, pp. 373–383, 1996.

[2] “The Scientific Case for P ≠ NP.” [Online]. Available: http://www.scottaaronson.com/blog/?p=1720

[3] E. Ogheneovo, “Revisiting cook-levin theorem using np-completeness and circuit-sat,” *International Journal of Advanced Engineering Research and Science*, vol. 7, pp. 206–213, 01 2020.

[4] “Hamilton Circuits/Graphs.” [Online]. Available: https://nitsri.ac.in/Department/Computer%20Science%20&%20Engineering/Lec4.pdf

[5] A. Meisels and A. Schaerf, “Modelling and solving employee timetabling problems,” *Annals of Mathematics and Artificial Intelligence*, vol. 39, 10 2001.

[6] J. H. Kingston, *Educational Timetabling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 91–108. [Online]. Available: https://doi.org/10.1007/978-3-642-39304-4_4

[7] “International timetabling competition.” [Online]. Available: http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index.htm

[8] Christos Gogos, Angelos Dimitsas, Vasileios Nastos and Christos Valouxis, “Some insights about the uncapacitated examination timetabling problem,” vol. 1, 09 2021.

[9] “Networkx-Network Analysis in Python.” [Online]. Available: https://networkx.org/

[10] P. Alefragis,C. Gogos,C. Valouxis,E. Housos, “A multiple metaheuristic variable neighborhood search framework for the uncapacitated examination timetabling problem,” *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, vol. 1, pp. 159–171, 2021.

[11] “Martin grötschel konrad-zuse-zentrum für informationstechnik berlin (zib) dfg research center matheon “mathematics for key technologies” institut für mathematik technische universität berlin groetschel@zib.de http://www.zib.de/groetschel graph colouring and frequency assignment.” [Online]. Available: https://www.zib.de/groetschel/teaching/SS2012/GraphCol%20and%20FrequAssignment.pdf

[12] N. Deo, J. S. Kowalik *et al.*, *Discrete optimization algorithms: with Pascal programs.* Courier Corporation, 2006.

[13] M. Adegbindin, A. Hertz, and M. Bellaïche, "A new efficient rlf-like algorithm for the vertex coloring problem," *Yugoslav Journal of Operations Research*, vol. 26, no. 4, pp. 441–456, 2016.

[14] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 417–427, 1983.

[15] "kempe_chain." [Online]. Available: https://planetmath.org/kempechain

[16] K. I. Appel and W. Haken, *Every planar map is four colorable.* American Mathematical Soc., 1989, vol. 98.

[17] R. Bellio, S. Ceschia, L. Di Gaspero, and A. Schaerf, "Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling," *Computers & Operations Research*, vol. 132, p. 105300, 2021.

[18] "logging — logging facility for python." [Online]. Available: https://docs.python.org/3/library/logging.html

[19] "Vn thesis page." [Online]. Available: https://github.com/vasnastos/Examination_Timetabling_DIT_UOI

[20] "Optimization hub." [Online]. Available: https://opthub.uniud.it/instance/timetabling/uncap-examtt/5fd0b6fced2d9f93276b4df0