ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

Πανεπιστήμιο Ιωαννίνων
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Πτυχιακή Εργασία

Χρονοπρογραμματισμός εξετάσεων

Βασίλειος Νάστος

Επιβλέπων: Χρήστος Γκόγκος
Αναπληρωτής καθηγητής Πανεπιστημίου Ιωαννίνων

29 Σεπτεμβρίου 2021

Uncapacitated Examination Timetabling

Εγκρίθηκε από τριμελή εξεταστική επιτροπή
Άρτα, //

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων Καθηγητής
   Χρήστος Γκόγκος

2. Μέλος Επιτροπής

3. Μέλος Επιτροπής

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 συνδιαστικής βελτιστοποιήσης που απασχολεί αρκετούς ερυνητές σε διάφορα ερευνητικά πεδία, ενώ ένας μεγάλος αριθμός από ερυνητικά άρθρα έχει σχηματιστεί βασισμένο 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 απλοποιήσης του προβλήματος, εξετάζονται οι τεχνικές δημιουργίας αρχικών λύσεων, και προτείνονται τεχνικές βελτιστοποιήσης του προβλήματος με χρήση της τεχνικής της τοπικής αναζήτησης της προσωμοιωμένης ανόπτησης καθώς και της αναρρίχεισης λόφων, εστιάζοντας και στην παραμετροποιήση των δύο αλγορίθμων.

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

Abstract

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 πολύμιτη 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,Laport και Lee [2], 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 οι οποίες μπορούν να φιλοξενίσουν μία εξέταση ή οι προτημήσεις του εισηγητή μίας εξέτασης όσον αφορά την περίοδο,κ.α. 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 ερυνητές 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 την βέλτιστη καθώς ο χρόνος υπολογισμού που απαιτείται είναι εκθετικά αυξανόμενος σε συνδυασμό με το μέγεθος των προβλημάτων. Στην παρούσα εργασία εξετάζεται η εφαρμογή της μέταευρετικής 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 την χρήση αλγορίθμων χρωματισμού γράφων. Προτείνονται επίσης τεχνικές ελαχιστοποιήσης του προβλήματος, οι οποίες και εφαρμόζονται ενώ παρουσίαζονται και τα αποτελέσματα που τα οποία προέκυψαν από την εφαρμογή τεχνικών βελτιστοποιήσης στο πρόβλημα.

1.1 Δομή εργασίας

Στο κεφάλαιο 2, αναφερόμαστε στις εννοίες P και NP, και περιγράφουμε ορισμένα προβλήματα χρονοπρογραμματισμού.Στο κεφάλαιο 3, πραγματοποιείται μία περιγραφή του προβλήματος του χρονοπρογραμματισμού εξετάσεων και των συνόλων δεδομένων που χρησιμοποιήθηκαν. 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 κεφαλαίο 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 προσωμοιωμέρνης ανόπτησης καθώς 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 των οποίων προέκυψαν, έπειτα από εφαρμογή των δύο τεχνικών. Τέλος στο κεφάλαιο 7, παρουσιάζονται τα συμπεράσματα που προέκυψαν από την εργασία καθως και οι μελλοντικές επεκτάσεις του προβλήματος.

Κεφάλαιο 2

Προβλήματα χρονοπρογραμματισμού

2.1 P και NP Προβλήματα

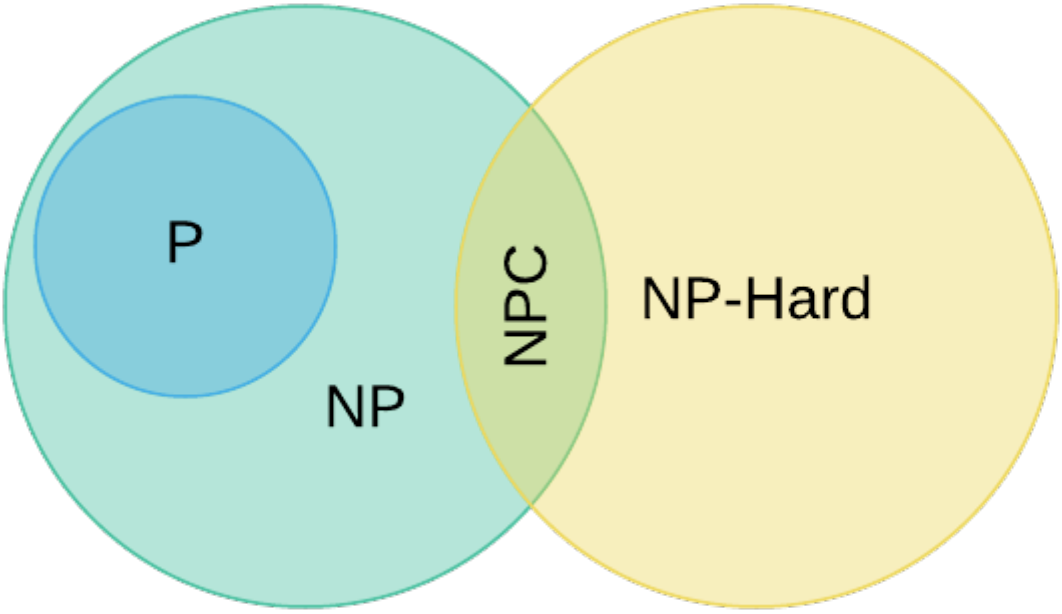
Η αποδοτικότητα ενός αλγορίθμου εξαρτάται από τον αριθμό των υπολογιστικών βημάτων,τα οποία θα πραγματοποιηθούν εως ότου επιλυθεί ένα πρόβλημα. Χωρίζουμε τα προβλήματα σε δύο κατηγορίες. Η πρώτη είναι τα εύκολα στην επίλυση τους (easy to solve)προβλήματα ενώ η δεύτερη τα προβλήματα τα οποία αναφέρονται ως δύσκολα στην επίλυση τους, κάτι που οδηγεί στην αύξηση των υπολογιστικών βημάτων 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 είναι ανάλογη του όγκου του αποτελέσματος του προβλήματος. Παραδειγμα προβλήματος που η δυσκολία του είναι αναμφίβολη απότελεί το πρόβλημα του πλανώδιου πωλητή(traveling salesman problem), το πρόβλημα εύρεσης κλίκας(clique) κ.α, προβλήματα αναζήτησης για τα οποία δεν φαίνεται να υπάρχει αποτελεσματικός τρόπός επύλισης.

2.1.1 Κλάση P

Η κλάση Π περιλαμβάνει το σύνολο των προβλήματων τα οποία μπορούν να επιβεβαιωθούν και να επιλυθούν σε πολυωνυμικό χρόνο. Τα προβλήματα που ανήκουν στην κλάση Π μπορούν να επιλυθούν σε χρόνο $\mathbb{O}(n^k)$ στην χειρότερη περίπτωση. Προβλήματα που ανήκουν στην κλάση είναι το πρόβλημα εύρεση των πρώτων αριθμών, το πρόβλημα εύρεσης του μέγιστου κοινού διαιρέτη, κ.α. Τα προβλήματα αυτά μπορούν να επιλυθούν από μία ντετερμινιστική μηχανή σε πολυωνυμικό χρόνο.

2.1.2 Κλάση NP

Η κλάση NP περιλαμβάνει τα προβλήματα που μπορούν επιβεβαιωθούν σε πολυωνιμικό χρόνο$\mathbb{O}(n^k)$, δωθέντος μίας λύσης τους. Τέτοια προβλήματα μπορούν να επιλυθούν από μία μη ντετερμινιστική μηχανή Turing σε πολυωνυμικό χρόνο. Ενα πρόβλημα που ανήκει σε αυτήν την κατηγορία είναι το Συδοκυ, καθώς η επιββεβαιωση μίας λύσης σε ένα ημι-συμπληρωμένο ταμλό μπορεί να πραγματοποιηθεί σε πολυωνυμικό χρονο.



Σχήμα 2.1: P VS NP [1]

2.1.3 NP-Hardness

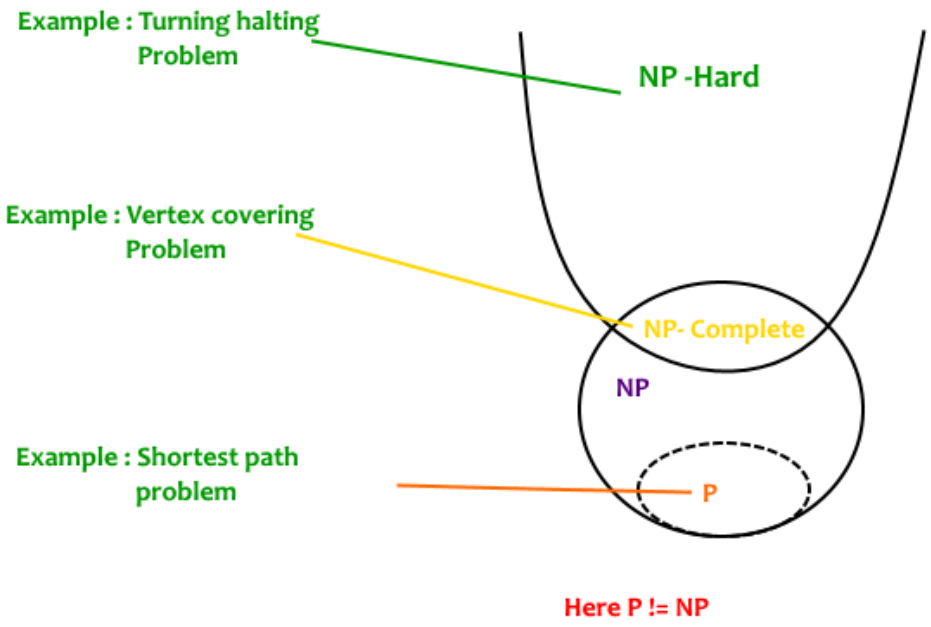
Ένα πρόβλημα H ανήκει στη κλάση NP-HARD, όταν κάθε πρόβλημα
*mathbb*L της κλάσης NP μπορεί να απλοποιηθεί σε ένα πολυωνυμικό πρόβλημα H. Μία παρανόηση είναι ότι το NP στο NP-HARD αντιστοιχεί στο μη πολυωνιμικό, ενώ πρακτικά αντιστοιχεί στον όρο μη ντετερμινιστικά πολυωνιμικά προβλήματα αποδοχής. Επίσης υπάρχει η υποψία ότι δεν υπάρχουν αλγόριθμοι που να επιλύουν σε πολυωνιμικό χρόνο τα προβλήματα της κλάσης NP-HARD είναι κάτι το οποίο δεν έχει αποδειχτεί [3]. Ένα πρόβλημα το οποίο ανήκει στην κατηγορία NP-HARD είναι το πρόβλημα εύρεσης υποσυνόλων.

2.1.4 NP-Completeness

Η κλάση NP-COMPLETE περιλαμβάνει το σύνολο των προβλημάτων που η κατάσταση τους παραμένει άγνωστη , καθώς κανένας αλγόριθμος επίλυσης σε πολυωνιμικό 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. 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 [4]. Άλλα προβλήματα της κλάσης NP-Complete είναι το πρόβλημα εύρεσης μονοπατιού η κυκλώματος Hamilton [5]. Επίσης στην κλάση NP-Complete ανήκει και το πρόβλημα του χρονοπρογραμματισμού εξετάσεων, με το οποία ασχολείται και η παρούσα εργασία.

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



Σχήμα 2.2: Κλάσεις Πολυπλοκότητας

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 Προβλήματα Χρονοπρογραμματισμού

Τα προβλήματα αναθέσεων αφορούν την κατανομή πόρων σε δραστηριότητες. Τα προβλήματα χρονοπρογραμματισμού δημιουργούνται συχνά από την ανάγκη του ανθρώπου για οργάνωση γεγονώτων. 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 ενδιαφέρωντος στον διεθνή διαγωνισμό χρονοπρογραμματισμού το 2010 και το 2014. [6].
- Χρονοπρογραπρογραμματισμός αθλητικών γεγονότων(Sports scheduling): Το πρόβλημα αφορά την δημιουργία προγραμμάτων για αθλητικά γεγονότα, με βάση τις ανάγκες και τους περιορισμούς που μπορούν να προκύψουν για το κάθε αθλητι-

κό γεγονός, οι οποίες διαφέρουν ανάλογα και με την φύση του αθλήματος. Ένα κλασικό παράδειγμα στο χώρο της επιχειριασιακής έρευνας

- 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. [7]

Κεφάλαιο 3

Χρονοπρογραμματισμός εξετάσεων

3.1 Περιγραφή προβλήματος

Το πρόβλημα του χρονοπρογραμματισμού εξετάσεων χωρίς περιορισμούς χωρητηκότητας 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 χρησιμοποιηθεί ο μεταευρετικός αλγόριθμος τοπικής αναζήτησης της προσομοιωμένης ανόπτησης και ο αλγόριθμος βελτιστοποιήσης της αναρρίχεισης λόφων.

3.2 Μοντελοποιήση προβλήματος

Σε αυτή την παράγραφο, ορίζουμε τους όρους διατύπωσης που πρόκειται να χρησιμοποιηθούν αργότερα στην τρέχουσα εργασία. Ως G, ορίζουμε τον γράφο ο οποίος θα μοντελοποιεί το πρόβλημα μας.Ως σύνολο $\mathbb{S}$, ορίζουμε το σύνολο των σπουδαστών, ως σύνολο $\mathbb{X}$ το σύνολο των εξετάσεων και ως σύνολο P το σύνολο των περιόδων. Ως **XS**, s∈ 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 αναπαριστούμε το πρόβλημα, στον οποίο ορίζουμε ως $\mathbb{V}$ τον αριθμό των κορυφών και ως $\mathbb{E}$ τον αριθμό των ακμών. Η τιμή του βάρους ακμής για δύο εξετάσεις $x_i, x_j \in X$ ορίζεται ως $\mathbb{W}_{x_i,x_j}$. Για μία εξέταση $x_i \in \mathbb{V}$, ορίζουμε ως $\ltimes_{x_i}$, το σύνολο των εξετάσεων με τις οποίες η εξέταση x_i έχει κοινούς φοιτητές. Επίσης για μία εξέταση x_i, εξετάσεις οι οποίες έχουν κοινούς φοιτητές με την εξέταση x_i, τις ονομάζουμε γειτονικές εξετάσεις. 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 πραγματικού χρόνου, 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 αριθμητικά δεδομένα. Η πρώτη γραμμή του συνόλου δεδομένων περιέχει τρία αριθμητικά δεδομένα, όπου το πρώτο αντοιστοιχεί 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,Laport και Lee [;]. Τα προβλήματα αντιστοιχούν σε εξετάσεις πραγματικού χρόνου από τρία καναδέζικα λύκεια,πέντε καναδέζικα,1 αμερικάνικο και 1 αγγλικό πανεπιστήμιο καθώς και 1 πανεπιστήμιο της μέσης ανατολής. 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 περιόδων 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 CE/LV^2. Ως CE ορίζεται ο συνολικός αριθμός των συγκρούσεων που θα δημιουργηθούν ανάμεσα στις εξετάσεις , ενώ ως LV ο συνολικός αριθμός εξετάσεων που υπάρχουν στο πρόβλημα. Ο συντελεστής πυκνότητας θα αντιστοιχεί σε μία δεκαδική τιμή $d \leq 1$.

3.3.2 ITC Datasets

Το σύνολο δεδομένων περιέχει δώδεκα προβληματα. Τα δωδεκα αυτά προβλήματα προβλήματα χρησιμοποιήθηκαν στον διεθνή διαγωνισμό χρονοπρογραμματισμού [8] για τα οποία πηγή προέλευσης αποτελούν διάφορα πανεπιστήμια. Το σύνολο δεδομένων περι-

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

έχει δώδεκα προβλήματα σε πραγματικό χρόνο. Παρόμοια και με το σύνολο δεδομένων Carter, κάθε γραμμή που ξεκινάει με το γραμμα s αντιστοιχεί σε εγγραφή ενός φοιτητή σε μία εξέταση ενώ σε αντίθετη περίπτωση η γραμμή αντιστοιχεί σε μία εξέταση, για την οποία δίνεται και πληροφορία για τον συνολικό αριθμό εγγραφών της, πληροφορία 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 αναπαραστηθούν με την χρήση γράφου. Η μοντελοποιήση αυτή των προβλημάτων μας επιτρέπει να αναζητήσουμε στον γράφο

μας σύνολα κορυφών $\mathbb{SE} \subseteq \mathbb{V}$, οι κορυφές των οποίων συνδέονται μεταξύ τους, χωρίς καποία από τις κορυφές $x_i \in \mathbb{SE}$ του συνόλου να δημιουργεί σύνδεση με κάποια από τις υπόλοιπες κορυφές του γραφήματος κάτι που 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 [9]:

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

Η ύπαρξη συνεκτικών τμημάτων, οδηγεί στην αποσύνθεση του προβλήματος, την δημιουργία επί μέρους προβλήματων και την επίλυση του κάθε τμήματος ξεχωριστά. Για την εύρεση των συνεκτικών τμημάτων χρησιμοποιήθηκε η μέθοδος connected_components($\mathbb{G}$) της networkx [10]. 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 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 στο γράφημα μας ορίζεται μία ακμή η οποία αν αφαιρεθεί οδηγεί στην δημιουργία περαιτέρω συνεκτικών τμημάτων. Για την εύρεση των γεφυρων στα προβλήματα των συνολων δεδομένων χρησιμοποιήθηκε η συνάρτηση της networkx [10], bridges($\mathbb{G}$). Επείτα αφαιρέθηκαν οι ακμές αυτές που οδήγησαν 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 πρακτικά το πρόβλημα κάτι που οδήγησε και στην απόρριψη της ιδέας. Ωστόσο δεν αποκλείται το γεγονός ότι εξαιτίας της φύσης του προβλήματος πιθανή αφαίρεση τέτοιων από διαφορετικά σύνολα δεδομένων να οδηγούσε σε καλύτερα αποτελέσματα.

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 όποιαδήποτε περίοδο και να διεξαχθούν θα έχουν την ίδια επιρροή στο πρόβλημα. Ως άνευ σημασίας σπουδαστές,όπως παρουσιάζεται και στο άρθρο [11], 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 ως προς το συνολικό κόστος της λύσης του προβλήματος. Στον πίνακα 3.4 παρουσιάζονται οι άνευ σημασίας σπουδαστές και για όλα τα προβλήματα των συνόλων δεδομένων.

Αρχείο δεδομένων	**Συνολικός αριθμός σπουδαστών**	**Άνευ 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: Άνευ σημασίας σπουδαστέςανά 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 [11] μία εξέταση θεωρείται 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 περίδο, χωρίς να επηρεαστεί η ποιότητα της λύσης. Μία εξέταση που χαρακτηρίζεται άνευ σημασίας, δεν πρόκυται 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 εξαιρεσουμε 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. Στον πίνακα 3.5 παρουσιάζονται οι άνευ σημασίας εξετάσεις ανά πρόβλημα και για τα δύο σύνολα δεδομένων.

Αρχείο δεδομένων	**Εξετάσεις**	**Άνευ σημασίας εξετάσεις**
car92	543	1
car91	682	4
ear83	190	0
hec92	81	0
kfu93	461	17
lse91	381	2
pur93	2419	6
rye93	486	1
sta83	139	0
tre92	261	1
uta92	622	0
ute92	184	0
yor83	181	0
ITC2007_1	607	0
ITC2007_2	870	0
ITC2007_3	934	19
ITC2007_4	273	0
ITC2007_5	1018	16
ITC2007_6	242	5
ITC2007_7	1096	59
ITC2007_8	598	13
ITC2007_9	169	4
ITC2007_10	214	1
ITC2007_11	934	19
ITC2007_12	78	4

Πίνακας 3.5: Άνευ 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 ένα ορίο μεγέθος για κάθε τμήμα το οποίο θα προκύψει σε ένα πρόβλημα. Κάθε τμήμα του προβλήματος $\mathbb{SG} \in \mathbb{G}$, 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 στο συνολικό κοστος της λύσης. 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 που κατηγοριοποιούνται ως άνευ σημασίας με βάση τον περιορισμό μεγέθους των συνεκτικών τμημάτων.

Αρχείο δεδομένων	**Άνευ 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
lse	168,256
pur93	552,1454,976,1520,983,2131, 340,341,342,343,2133,153
rye93	304
sta83	∅
tre	186
uta92	∅
ute92	∅
yor83	∅

Πίνακας 3.6: Άνευ σημασίας εξετάσης με βάση το μέγεθος των συνεκτικών τμημάτων

3.4.3 Άνευ σημασίας εξετάσεις με βάση των βαθμό τους

Η τρίτη κατηγορία άνευ σημασίας εξετάσεων βασίζεται στον βαθμό τους, έχοντας μοντελοποιήσει το πρόβλημα υπό την μορφή ενός γραφήματος. Ο βαθμός μίας κορυφής σε ένα γράφημα είναι ο συνολικός αριθμός εισερχομένων της. Στο πρόβλημα του χρονοπρογραμματισμού εξετάσεων, ως βαθμό μίας εξέτασης ορίζουμε τον συνολικό αριθμό των εξετάσεων με τις οποίες έχει κοινούς σπουδαστές. Για παράδειγμα, αν μία εξέταση $e \in \mathbb{V}$, συνδέεται με τις εξετάσειςe_1, e_2, e_3, όπου $e_1, e_2, e_3 \in en_e$, τότε ο βαθμός της εξέτασης e είναι 3. Με βάση το άρθρο [9], αν μία εξέταση e έχει βαθμό μικρότερο από $\lfloor \frac{P}{11} \rfloor$, τότε θεωρείται άνευ σημασίας, καθώς η κάθε εξέταση προγραμματισμένη σε μία περίοδο $p \in \mathbb{P}$, συμμετέχει στο κόστος της λύσης με γειτονικές εξετάσεις, οι οποίες έχουν προγραμματιστεί σε διαφορετική πέριοδο εύρους $[p-5, p+5]$. Επομένως

καταλήγουμε στο συμπέρασμα ότι η κάθε περίοδος αλληλεπιδρά με 10 διαφορετικές περιόδους, ενώ στην περίοδο την οποία έχει προγραμματιστεί η εξέταση δεν μπορεί να προγραμματιστεί άλλη γειτονική εξέταση. Πιθανός βαθμός εξέτασης $\leq \lfloor \frac{P}{11} \rfloor$, επιτρέπει τον προγραμματισμό της εξέτασης και των γειτονικών εξετάσεων της, σε χρονικές περιόδους η οποίες θα αποτρέψουν την συμμετοχή της εξέτασης στο συνολικό κόστος της λύσης, καταλήγωντας στο συμπέρασμα ότι αντίστοιχες εξετάσεις θεωρούνται άνευ σημασίας. Η διαδικασία εύρεσης τέτοιων εξετάσεων επαναλαμβάνεται εως ότου δεν βρεθεί εξέταση στο πρόβλημα η οποία να μπορει να θεωρηθεί άνευ σημασίας. Σε κάθε επανάληψη αφαιρούμε από τον γράφο μας εξετάσεις που προηγουμένως έχουν οριστεί ως άνευ σημασίας, και υπολογίζουμε τον βαθμό των υπόλοιπων εξετάσεων που υπάρχουν στον γράφο. Στον πίνακα 3.7 παρουσιάζοναι οι εξετάσεις οι οποίες κατηγοριοποιούνται ως άνευ σημασίας λόγω του βαθμού τους, για τα δύο σύνολα δεδομένων.

Αρχείο δεδομένων	**Θορυβώδεις 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	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 3.7: Άνευ σημασίας εξετάσεις με βάση τον βαθμό των εξετάσεων

Παρουσιάστηκαν τρεις κατηγορίες εξετάσεων άνευ σημασίας, οι οποίες οδηγούν στην απλοποιήση του γράφου μας σε ένα γράφο μικρότερου μεγέθους. 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

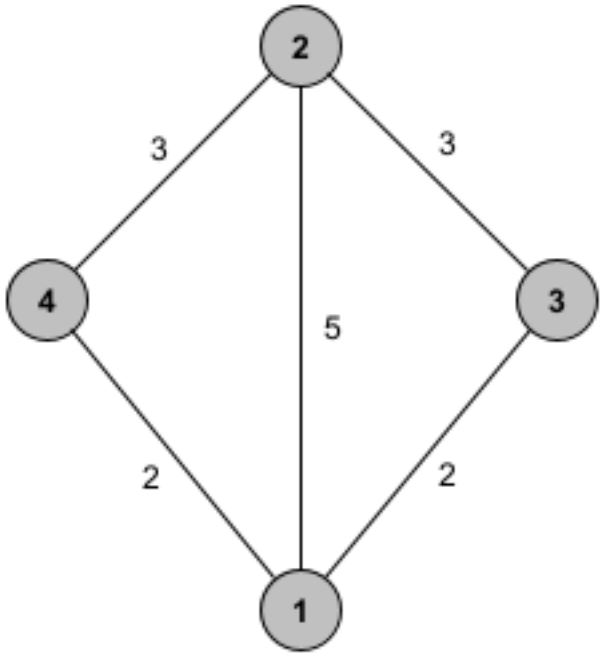
ο συνολικός αριθμός άνευ σημασίας εξετάσεων, και ο συνολικός αριθμός για κάθε κατηγορία, για τα σύνολα δεδομένων. Σημειώνεται ότι μία εξέταση μπορεί να ανήκει σε δύο κατηγορίες άνευ σημασίας εξετάσεων. Οι εξετάσεις αυτές έχουν αφαιρεθεί από τον γράφο. Κατα την διαδικασία βελτιστοποιήσης της λύσης ενός 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 μπορούν να μεταβάλλουν το κόστος της λύσης μας.

Αρχείο δεδο-μένων	**Άνευ σημασίας εξετάσεις βάση των σπουδαστών**	**Άνευ σημασίας εξετάσεις βάση των συνεκτικών τμημάτων**	**Άνευ σημασίας εξετάσεις βάση βαθμού**	**Συνολικός αριθμός Άνευ σημασίας εξετάσεων**
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				

Πίνακας 3.8: Συνολικός άνευ σημασίας εξετάσεις στα συνόλων δεδομένων

3.5 Συμμετρικές εξετάσεις

Σε αυτήν την παράγραφο θα ασχοληθούμε με τις πανομοιότυπες εξετάσεις, εξετάσεις οι οποίες μπορούν να προγμαμματιστούν στην ίδια περίοδο και έχουν κοινά χαρακτηριστικά. Το πρώτο σύνολο πανομοιότυπων εξετάσεων με τις οποίες θα αχοληθούμε αφορούν εξετάσεις η οποίες δεν έχουν κοινούς φοιτητές μεταξύ τους, ωστόσο έχουν κοινές γειτονικές εξετάσεις και για κάθε μία από τις εξετάσεις που συνδέονται έχουν τον ίδιο αριθμό κοινών φοιτητών. Αυτό το σύνολο συμμετρικών εξετάσεων το ονομάσουμε $\mathbb{I}$. Αυτές οι εξετάσεις θα προγραμματιστούν την ίδια βέλτιστη χρονική στιγμή σε μία βέλτιστη λύση, μιας και δεν συγκρούονται μεταξύ τους και έχουν τις ίδιες γειτονικές εξετάσεις με τα ίδια βάρη. Ένα παράδειγμα πανομοιότυπων εξετάσεων φαίνεται στην εικόνα ανάμεσα στις εξετάσεις 3 και 4. Εξατάσεις που παρασουσιάζουν την συγκεκριμένη μορφή συμμετρίας μπορούν να προγραμματιστούν με κοινό τρόπο , τοποθετούμενες σε μία βέλτιστη περίοδο. Άλλη μία μορφή συμμετρικών εξετάσεων η οποία υπολογίζεται είναι οι εξετάσεις που έχουν εγγραφεί οι ίδιοι φοιτητές. Αυτές οι εξετάσεις θα κατανέμονται πάντα σε διαφορετική περίοδο, στοχεύοντας πάντα στην κατανομή που θα επιτύχει το μικρότερο κόστος. Αναζητώντας συμμετρικές εξετάσεις, επιτυνχάνουμε την μείωση του χώρου αναζήτησης, δουλεύωντας με ένα μικρότερο σύνολο εξετάσεων. Ο



Σχήμα 3.1: Εξετάσεις 3,4$\epsilon\mathbb{I}$

αριθμός των εξετάσεων οι οποίες ανήκουν στο σύνολο I καθώς και ο αριθμός των εξετάσεων που έχουν τους ίδιους συμμετέχοντες παρουσιάζετε στον πίνακα και για τα σύνολα δεδομένων.

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 ς χρωμάτων στους κόμβους του γραφήματος τηρώντας πάντα τον περιορισμό κοινού χρωματισμού ανέμεσα σε γειτονικές κορυφές. Ως χρωματική κλάση X_i ορίζουμε το σύνολο των κόμβων που θα χρωματιστουν με το χρώμα. Ως χρωματικό αριθμό ορίζουμε το σύνολο των ελέχιστων χρωμάτων που θα χρησιμοποιηθούν στην διαδικασία ενός έγκυρου χρωματισμού. Μερικές από τις εφαρμογές που χρησιμοποιήται ο χρωματισμός γράφων είναι:

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

- **Δημιουργία προγραμματων και χρονοπρογμαμμάτων**
- **Κατανομή ραδιωφωνικών συχνοτήτων(Mobile Radio Frequency Assignment) [12]**.
- **Sudoku**

Στο πρόβλημα του χρονοπρογραματισμού εξετάσεων τα χρώματα αντιστοιχούν στις διαθέσιμες χρονικές περιόδους. Από την μοντελοποιήση του προβλήματος ως γράφο, οι εξετάσεις αντιστοιχούν στις κορυφές και τοι κοινούς φοιτητες μεταξύ εξετάσεων αντιστοιχούν στις ακμές όπως παρουσιάστηκε 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 εξέταση. Με τη μέθοδο του χρωματισμού επιτυνχανουμε την δημιουργία ένος αρχικού προγράμματος των εξετάσεων. Για να θεωρηθεί έγκυρή μία λύση πρέπει ο αριθμός των περιόδων να είναι μικρότερος από τον όριο που ορίζεται στον πίνακα για κάθε πρόβλημα, και να παραβιάζεται ο ισχυρός περιορισμός τον οποίο ορίσαμε. Για την εκτέλεση χρωματισμού στους γράφους στα προβλημάτα των συνόλων δεδομένων, χρησιμοποιήθηκε η συνάρτηση της βιβλιοθήκης networkx [10] greedy_color($\mathbb{G}$). Οι αλ-

γόριθμοι που χρησιμοποιούνται για τον χρωματισμό των γράφων είναι άπληστοι κάτι που σημαίνει ο αριθμός των περιόδων που θα χρησιμοποιήσει θα είναι κατα μέγιστο d+1, όπου το d όρίζεται ως ο μέγιστος βαθμός εξέτασης στον γράφο των οποίο εξετάζουμε. Οι στρατιγικές 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. Δοθέντως μίας συνάρτησης χρωματισμού $\mathbb{CF}$ και μίας εξέτασης , όπου $\mathbb{CF}() \in [i,j]$, τότε οι τεχνικές εναλλακτικού χρωματισμού επιτρέπουν τον επανορισμό της συνάρτησης χρωματισμού, ώστε αν $\mathbb{CF}() \in i$, τότε με τον επανορισμό της συνάρτησης $\mathbb{CF}() \in j$, δίνοτας την δυνατότητα στην συνάρτηση χρωματισμού να αποφύγει ένα καινούργιο χρώμα , εναλλάσοντας χρώματα ανά τους υπογράφους $(SG) \in \mathbb{G}$ όταν είναι εφικτό, τροποποιώντας την υπαρχουσα λύση [13]. 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.

3.6.1 Largest First

Ο αλγόριθμος largest first αποτελεί έναν από τους πιο δημοφηλής 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 $\mathbb{C}$ 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 $\mathbb{C}$. Ο αλγόριθμος αναλαμβάνει την κατασκευή συνόλων που θα

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

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 $\mathbb{U}$, περιέχει τις κορυφές που δεν έχει χρωματίσει ο αλγόριθμος, ενώ το σύνολο $\mathbb{W}$, τις κορυφές $e \in \mathbb{U}$, οι οποίες συνδέονται με την κορυφή u. Ο αλγόριθμος επιλέγει την κορυφή που έχει τους περισσότερους γείτονες στο σύνολο $\mathbb{W}$. Έπειτα πραγματοποιήται ο χρωματισμός της κορυφής και η διαδικασία επαναλαμβάνεται όσο $\mathbb{U} \neq \emptyset$ [14]. Για να εκτελέσουμε χρωματισμό γράφων στο παράδειγμα μας με χρήση του αλγορίθμου largest first χρησιμοποιήσαμε την συνάρτηση greedy˜color της βιβλιοθήκης networkx [10] . Η στρατιγική largest˜first υπάρχει ενσωματωμένη σαν στρατιγική 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 $u \in \mathbb{U}$.
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 $\mathbb{U}$.
Όσο $\mathbb{U} \neq \emptyset$

- 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 $\mathbb{U}$ στο σύνολο C_v, και μετικίνηση των γειτονικών κορυφών $w \varepsilon \mathbb{U}$ της κορυφής u στο σύνολο $\mathbb{W}$.

Τέλος επανάληψης

Σχήμα 3.2: 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 επιτευχθει χρωματισμός γράφων είναι η στρατιγική 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 στρατιγική smallest last.

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 $e \in \mathbb{V}$ με τον μεγαλύτερο βαθμό. Ο βαθμος μίας κορυφής ισούται με τον αριθμό των συνδέσεων μίας κορυφής με τις υπόλοιπες κορυφές του γραφήματος. 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 nv με την υψηλότερο τιμή κορεσμού ($deg(nv)$). 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. Στην κορυφή nv εφαρμόζεται το πρώτο διαθέσιμο χρώμα. Η διαδικασία συνεχίζει να εκτελείται επαναληπτικά εως ότου δεν απομένει καμία κορυφή προς εξέταση. Στον πίνακα 3.12 παρουσιάζονται οι συνολικές περιόδοι που θα χρησιμοποιηθούν για κάθε πρόβλημα στα δύο σύνολα δεδομένων, έπειτα από εφαρμογή του αλγορίθμου DSATUR.

1: **Είσοδος** Ο γράφος $\mathbb{G}$
2: **Έξοδος** Αποτελέσμα χρωματισμού c_v:$v \in V$
3: C:=$\emptyset$, U:=V, computedeg$_{G(U)}$
4: Επιλογή μίας κορυφής v, όπου $v \in \mathbb{V}$ και
5: $max_{v \in \mathbb{U}}\{deg_{G(U)}()\}$
6: $c(v)$:=1, C:=$C \cup \{v\}$, U:=$U/\{v\}$·
7: Ενημέρωση βαθμών κορεσμού.
8: **while** $U \neq \emptyset$ **do**
9: Εύρεση κορυφής v όπου $max_{v \in \mathbb{U}}\{dsatur_{G(C)}(v)\}$
10: Εαν υπάρχει σύνολο κορυφών που έχουν τον ίδιο μέγιστο βαθμό κορεσμού, επιλέγεται τυχαία μία κορυφη.
11: Εύρεση του του πρώτου διαθέσιμου χρώματος k για χρωματισμό της κορυφής v.
12: $c(v) := k$· C:=$C \cup \{v\}$· U:=$U/\{v\}$·
13: Ενημέρωση βαθμών κορεσμού $deg_{G(U)}$
14: **end while**

Σχήμα 3.3: 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 στρατιγηκές χρωματισμού καθώς και πέντε παραλλαγές των πέντε από τον επτά στρατηγικών. Οι τρεις στρατηγικές που επιτυνχάνουν 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. Στον πίνακα 3.14 παρουσιάζονται τα αποτελέσματα για τους υπόλοιπες στρατηγικές χρωματισμού που χρησιμοποιήθηκαν. Με έντονο χρώμα σημειώνονται οι στρατηγικές που επέτυχαν έγκυρο αριθμό περιόδων σύμφωνα με τον πίνακα. Αξίζει να σημειωθεί ότι η στρατιγική random sequential εισάγει τυχαιότητα στον τρόπο επίλυσης του, οδηγώντας σε τυχαία παραγωγή εφικτών και μη εφικτών λύσεων.

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 τοπικής αναζήτησης της προσωμοιωμένης ανόπτησης(simulated annealing) καθώς και η τεχνική βελτιστοποιήσης της αναρρίχεισης 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 καταστάσεις στις οποίες μπορεί να πραγματοποιηθεί μετάβαση από την αρχική λύση, καθιστώντας όλες τις καταστάσεις ως πιθανές λύσεις. Κάθε κατάσταση αξιολογείται ως προς την επιρροή της στην τρέχουσα λύση. Η αξιολόγηση των καταστάσεων πραγματοποιήται με την βοήθεια της αντικειμενικής συνάρτησης. Ονομάζουμε την αριθμητική μεταβλητή, η οποία θα καθορίζει την επιρροή που θα έχει μία κατάσταση στην τρέχουσα λύση ως 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 $x_i, x_j \in \mathbb{X}$, 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 προγραμματιστηκέ 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 * $\mathbb{W}_{x_i x_j}$.
- Για κάθε $x_i, x_j \in \mathbb{X}$, 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 στην που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 8 * $\mathbb{W}_{x_i x_j}$.

- Για κάθε $x_i, x_j \in \mathbb{X}$, 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 στην που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 4 * $\mathbb{W}_{x_i x_j}$.
- Για κάθε $x_i, x_j \in \mathbb{X}$, 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 προγραμματτιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 2. Η συμμετοχή στην που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 2 * $\mathbb{W}_{x_i x_j}$.
- Για κάθε $x_i, x_j \in \mathbb{X}$, 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 προγραμματτιστηκέ η εξέταση x_j, η ποινή που παράγουν οι δύο εξετάσεις είναι 1. Η συμμετοχή στην που θα έχουν στην αντικειμενική συνάρτηση οι δύο εξετάσεις θα είναι 1 * $\mathbb{W}_{x_i x_j}$.

Εφαρμόζωντας τους παραπάνω κανόνες σε όλα τα ξεύγη γειτονικών εξετάσεων, υπολογίζεται το κόστος της λύσης, που αποτελεί ένα ακέραιο.

4.2 Περιγραφή αλγοριθμου προσωμοιωμένης ανόπτησης

Μια πολύ σημαντική μέθοδος στην επίλυση προβλημάτων βελτιστοποιήσης είναι η μέθοδος της προσωμοιωμένης 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 ψήξη 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 προσωμοιωμένης ανόπτησης εκκεινεί με μία συγκεκριμένη υψηλή τιμή θερμοκρασίας. Σε κάθε επανάληψη η προσωμοιωμένη ανόπτηση χρησιμοποιεί μία γειτονιά καινούργιων καταστάσεων $N(s)$ που μπορούν να δημιουργηθούν με βάση την τρέχουσα λύση s_0. Η επιλογή της καινούργιας κατάστασης πραγματοποιήται με τυχαίο τρόπο. Για την δημιουργία καινούργιων λύσεων χρησιμοποιούνται οι τελεστές γειτνίασης, οι οποίοι εκτελούν μεταβολλές περιόδων στην αρχική λύση, πραγματοποιώντας κινήσεις οι οποίες θα οδηγήσουν στην δημιουργία νέων πιθανών 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 λύσηςs_n. Σε περίπτωση που το κόστος της λύσης s_n είναι μικρότερο από το κόστος της λύσης s_0, τότε πραγματοποιήται αντικατάσταση της τρέχουσας λύσης, από την υποψήφια λύση($s_0 = s_n$).Σε αντίθετη περίπτωση υπολογίζεται η διαφορά d του κόστους της λύσης s_n, από την λύση s_0.Αν $d > 0$, τότε ορίζεται μία πιθανότητα αποδοχής της υποψήφιας λύσης, 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 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 συντελεστής ψήξης με την τιμή θερμοκρασίας κατά την προηγούμενη επανάληψη. Όσο η θερμοκρασία μειώνεται τόσο δυσχερένει 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 ορίσει ο χρήστης. Επίσης σε κάθε βήμα του αλγορίθμου ελέγχεται η λύση η οποία προτάθηκε ως προς την βέλτιστη λύση που έχει βρεθεί μέχρι εκείνο το σημείο. Στο τέλος κάθε επανάληψης επιλέγεται η βέλτιστη λύση που έχει βρεθεί μέχρι εκείνο το σημείο, δίνοντας την δυνατότητα διαφυγής από λύσεις που αποτελούν τοπικά ακρά και βοηθόντας τον αλγόριθμο στην αναζήτηση ολικού άκρου. Κατα την διάρκεια της διαδικασίας της προσωμοιωμένης ανόπτησης υπάρχουν πολλοί παράγοντες που μπορούν να ρυθμιστούν και να δοκιμαστούν, παράγοντες όπως ο καθορισμός της γειτονιας των εξετάσεων που θα πραγματοποιήθεί η αναζήτηση βέλτιστης λύσης, η μέθοδος με την οποία θα μειώνεται η τιμή της θερμοκρασίας ή ο αριθμός των εσωτερικών επαναλήψεων. Στο σχήμα 4.1 περιγράφεται η διαδικασία της προσωμοιώμενης ανόπτησης. Η συνάρτηση stop_function() είναι η συνάρτηση που ορίζει την σύνθηκη τερματισμού της διαδικασίας, ενώ η συνάρτηση is_feasible(), ελέγχει την εφικτότητα μίας υποψήφιας λύσης.

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

Σχήμα 4.1: Ψευδοκώδικας διαδικασίας προσωμοιωμένης ανόπτησης

4.3 Περιγραφή αλγορίθμου αναρρίχισης λόφων

Η διαδικασία της αναρρίχισης λόφων αποτελεί διαδικασία βελτιστοποιήσης όπως και η διαδικασία της προσωμοιωμένης ανόπτηση. Ο αλγόριθμος ανέκει στους αλγορίθμους τοπικής αναζήτησης. Ο αλγόριθμος παρόμοια και με τον αλγόριθμο της προσωμοιωμένης εναλλάσεται από κατάσταση σε κατάσταση, με σκοπό την βελτίωση της τρέχουσας κατάστασης. Τα προβλήματα για τα οποία πραγματοποιεί βελτιστοποιήση ο αλγόριθμος χωρίζονται σε δύο κατηγορίες, τα προβλήματα ελαχιστοποιήσης και τα προβλήματα μεγιστοποιήσης. Για την παραγωγή καινούργιων καταστάσεων, χρησιμοποιούμε τους τελεστές γειτνίασης. Η μεταβολή της λύσης, πραγματοποιείται μόνο σε περίπτωση βελτιστοποιήσης της αντικειμενικής τιμής του προβλήματος. Κάθε λύση η οποία εξετάζεται ως υποψήφια, συγκρίνεται και με την βέλτιστη λύση που έχει προκύψει εως την συγκεκριμένη επανάληψη, δίναντοας 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 προσωμοιωμένης ανόπτησης, και υπολογίζει το κόστος της λύσης. Για να βελτιωθεί η αρχική λύση, πρέπει η καινούργια κατάσταση που επιλέγεται, να ελαχιστοποιεί το κόστος της λύσης. Ο αλγόριθμός εκτελείται επαναληπτικά μέχρι να ικανοποιήθει κάποια συνθήκη τερματισμού που έχει οριστεί. Αν δεν υπάρχει κάποια συνθήκη τερματισμού τότε ολγόριθμος διακόπτει την λειτουργία του σε περίπτωση που η μετάβαση σε μία κατάσταση δεν οδηγήσει σε ελαχιστοποιήση του κόστους του προβλήματος.

Στο πρόβλημα του χρονοπρογραμματισμού εξετάσεων απότελει ένα πρόβλημα ελαχιστοποίησης. Σκοπος της βελτιστοποίησης που θα πραγματοποιήσουμε είναι η μείωση του κόστους. Η δημιουργία νέων καταστάσεων πραγματοποιήται με την βοήθεια των τελεστών γειτνίασης. Η επιλογή μίας κατάστασης πραγματοποιείται τυχαία, ενώ η κατάσταση αντικαθιστά την τρέχουσα κατάσταση εαν υπάρξει ελαχιστοποίηση του κόστους λύσης. Στο σχήμα 4.2 παρουσιάζονται τα βήματα της διαδικασίας της αναρρίχισης λόφων.

1: Δημιουργία αρχικής λύσης s_0
2: **while** stop_function() **do**
3: Παραγωγή καταστάσεων $s \in NS$(σύνολο υποψήφιων λύσεων)
4: **if** $OB(s) \leq OB(s_0)$ **then**
5: $s_0 = s$
6: **end if**
7: **end while**

Σχήμα 4.2: Αλγόριθμος αναρρίχισης λόφων

4.4 Τελεστές γειτνίασης

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

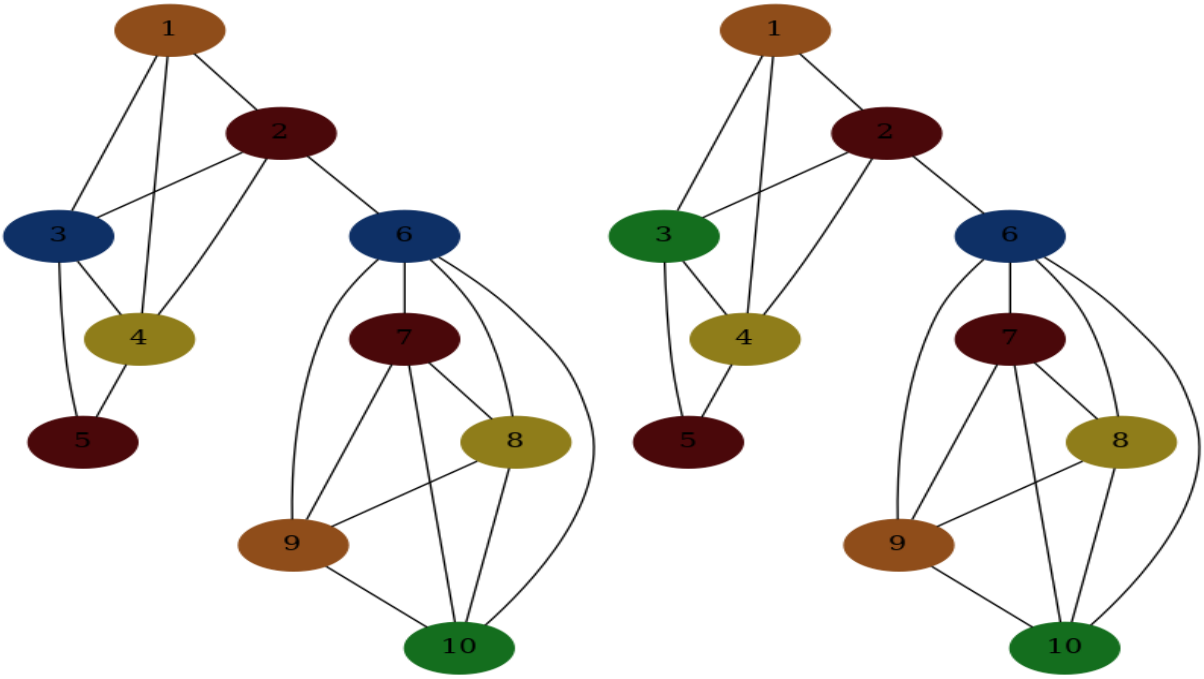
- Μετακίνηση περιόδου εξετάσης
- Εναλλαγή περιόδων εξετάσεων
- Ανταλλαγή εξετάσεων μεταξύ περιόδων
- Ολίσθηση εξετάσεων ανά περίοδο
- Αλυσίδες Kempe
- Εξαγωγή εξάτασης
- Διπλή εξαγωγή εξέτασης

Σε κάθε βήμα επιλέγεται μία τυχαία εξέταση $e \in \mathbb{X}$. 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 απέχουν 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:

- $p \neq F_e$
- Πρέπει να μην παραβιάζεται ο περιορισμός, του μη κοινού χρωματισμού μεταξύ εξετάσεων με κοινούς φοιτητές.



Σχήμα 4.3: Μετακίνηση εξέτασης 3 σε διαφορετική περίοδο

4.4.2 Εναλλαγή περιόδων εξετάσεων

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

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

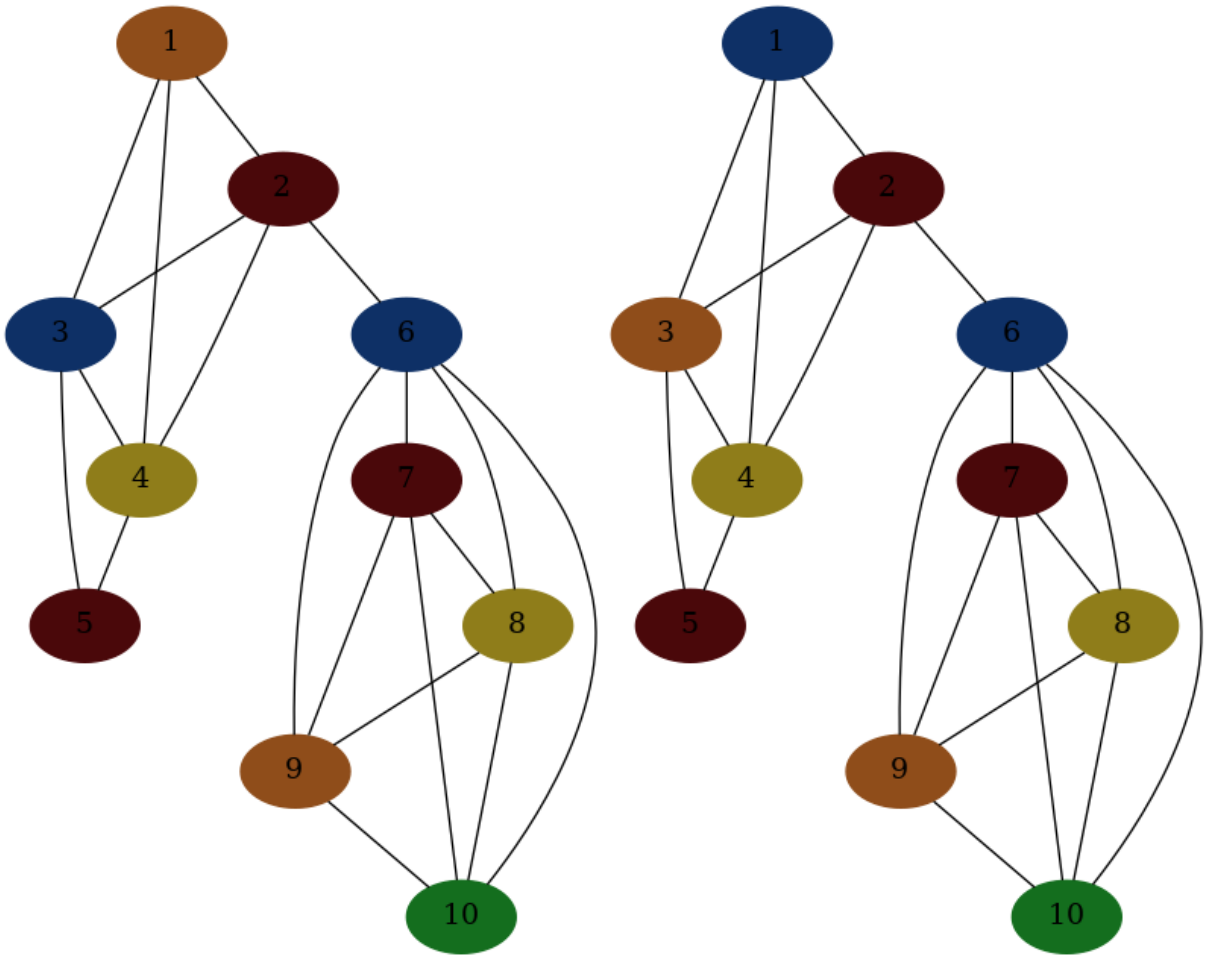
$$\mathbb{W}_{ee_n} \neq 0 \tag{4.2}$$

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

$$\left|F_e - F_{e_n}\right| <= 5 \tag{4.3}$$

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. Σημαντική παρατήρηση αποτελεί το γεγονός ότι η συγκεκριμένη κίνηση επιτυνχανει να δημιουργήσει χαμηλότερου κόστους κατάστασεις κατά τα αρχικά στάδια της διαδικασίας βελτιστοποίησης. Όσο οι μεταβολλές των καταστάσεων αυξάνονται, η κίνηση αποτυνχάνει να δημιουργήσει εφικτές καταστάσεις $s_n \in N(s)$, καθώς μειώνεται η πιθανοτητα εφικτής ανταλλαγής περιόδων ανάμεσα σε δύο σύνολα εξετάσεων.



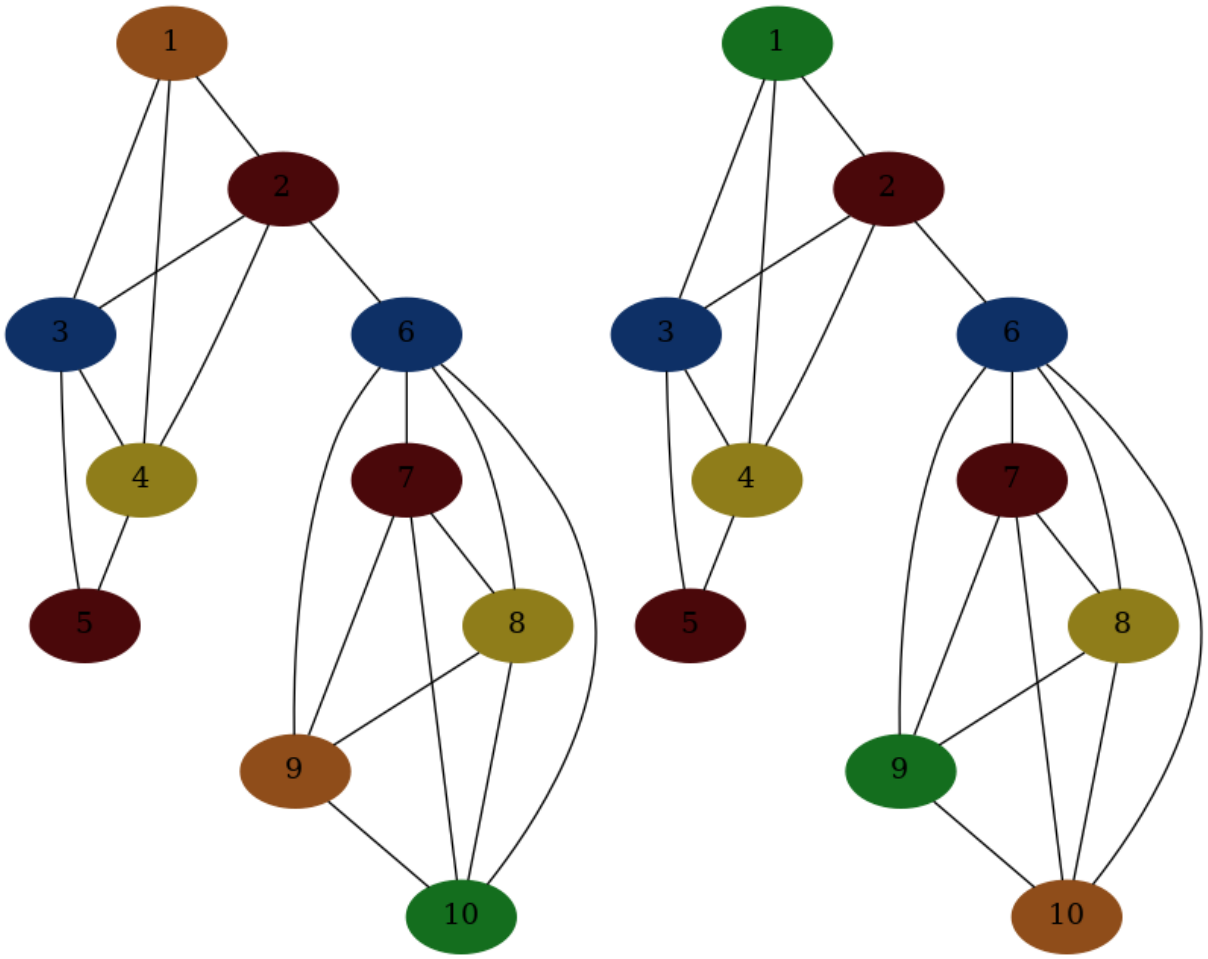
Σχήμα 4.4: Ανταλλαγή περιόδων μεταξύ εξετάσεων 1 και 3

4.4.4 Ολίσθηση εξετάσεων ανά περίοδο

Σε κάθε εκτέση 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 πραγματοποιώντας διάσχιση, παρατηρείται εναλλαγή μεταξύ συγκεκριμένων 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



Σχήμα 4.5: Εναλλαγή περιόδων (Πορτοκαλί-Πράσινο)

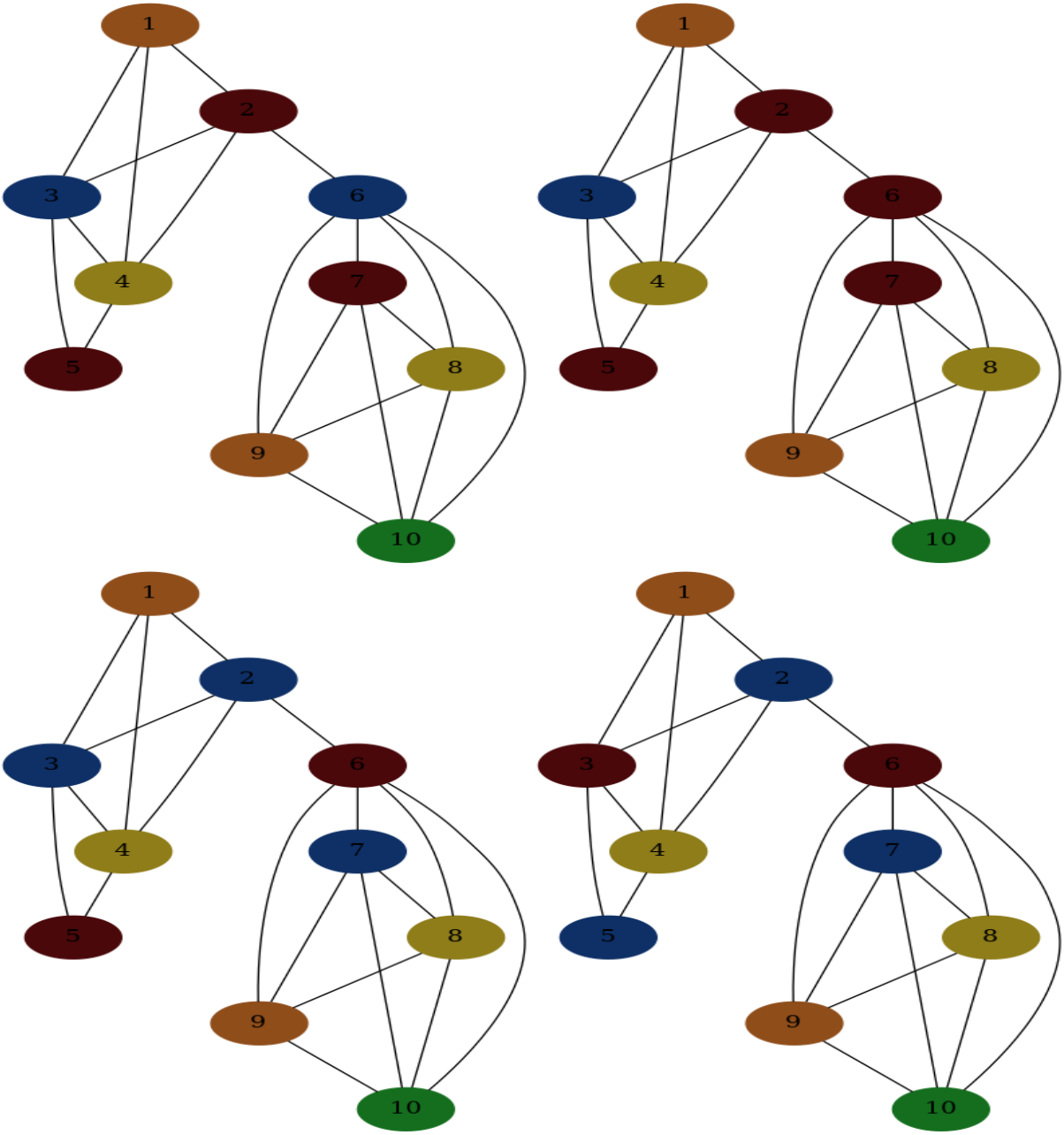
αποτέλεσε το πρώτο σημαντικό θεώρημα που αποδυκνύεται με την χρήση υπολογιστή. Η υπόθεση του θεωρήματος των τεσσάρων χρωμάτων αναπτύχθηκε προτάθηκε αρχικά από τον φοιτητή Francis Guthrie, οποίος προσπάθησε να χρωματίσει τον χάρτη των περιφερειών της Αγγλίας. Μία απόδειξη του θεωρήματος δώθηκε από τον 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 απ το πανεπιστήμιο του Ιλλινόις, με την βοήθεια του john koch σε ένα πείραμα διάρκεια 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 $F(e_1) = F(e_2)$ και $F(e_1) = F(e_2)$. Στην συνέχεια ελέγχονται όλες οι γειτονικές εξετάσεις της εξέτασης e_2. Για κάθε γειτονική εξέταση e_{ni}, της κορυφής e_1 όπου $F(e_{ni}) = F(e_2)$, 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 $F(e_2)$.

$F(e_1)$	$\forall e_{ni}$ and $F(e_{ni}) = F(e_2)$	(4.4)
$F(e_2)$	$\forall e_{ni}$ and $F(e_1) = F(e_{ni})$	(4.5)

Σχήμα 4.6: Περιορισμοί κίνησης kempe_chain

Η διαδικασία εκτελείται επαναληπτικά 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.

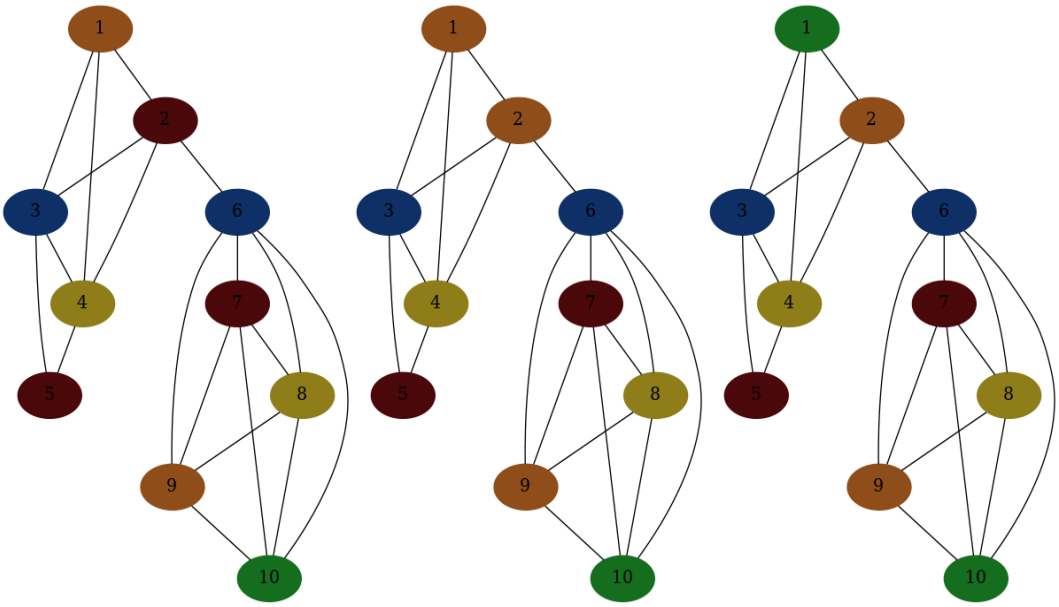


Σχήμα 4.7: Παράδειγμα αλυσίδας kempe (Κόκκινο-μπλε χρώμα)

4.4.6 Εξαγωγή εξέτασης

Επιλογή δύο εξέτασεων $e1 \in V, e2 \in V$. 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:

- $W_{e1e2} \neq 0$
- $F_{e1} \neq F_{e2}$
- $|F_{e1} - F_{e2}| <= 5$

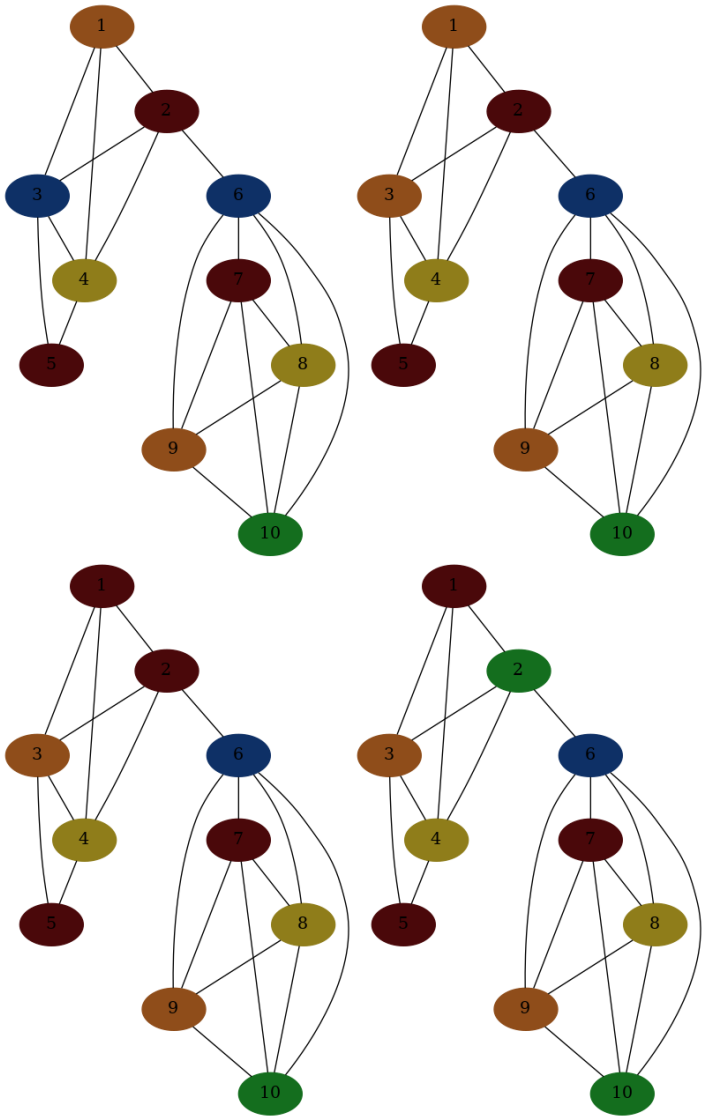


Σχήμα 4.8: Εκτέλεση κίνησης εξαγωγής εξέτασης μεταξύ εξετάσεων 2,1

4.4.7 Διπλή εξαγωγή εξετάσης

Η κίνηση της διπλή εξαγωγής εξετάσεων, αποτελεί επέκταση της ιδέας της κίνησης απλής εξαγωγής εξετάσεων καθώς επεκτείνει τον όγκο των κινήσεων. 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 $e1, e2, e3 \in V$.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 4.9: 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 σε όλες τις εξετάσεις ενός προβλήματος. Σε πιθανή περίπτωση βελτιστοποιήσης 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 επη-

ρεάζει 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 συγκεντρώνει αρκετές πιθανότητες λόγω της πολυπλοκότητας των περιορισμών της, να μην επηρεάσει κάποια εξέταση. Επίσης η αντιστοίχη πιθανοτήτων εκτέλεσης με την κάθε κίνηση είναι μία ιδέα που έχει προταθεί στο άρθρο [17]. Και οι δύο προτάσεις που αναφέρθηκαν μπορούν να επηρεάσουν την απόδοση των τελεστών γειτνίασης.

Αξίζει να σημειωθεί ότι κινήσεις που αλληλεπιδρούν με περιόδους και όχι συγκεκριμένες εξετάσεις επηρεάζουν διαφορετικό αριθμό εξετάσεων ανά κίνηση, αριθμός ο οποίος εξαρτάται από την τεχνική χρωματισμού που έχει εμφανιστεί καθώς και από κινήσεις βελτιστοποιήσης της αρχικής λύσης που εφαρμόστηκαν σε προηγούμενη επανάληψη.

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 για την δημιουργία των βασικών οντοτήτων του προβλήματος. Στον πίνακα παρουσιάζονται τα στοιχεία του υπολογιστή που χρησιμοποιήθηκε για την εκτέλεση των πειραμάτων.

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 αποθηκεύωντας πληροφορίες σε σχέση με 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 φοιτητές μεταξύ δύο στιγμυοτύπων εξετάσεων και θα χρησιμοποιηθεί για την εύρεση των βαρών των κορυφών, όταν μοντελοποιήσουμε το προβλήμαμε 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, ο οποίος θα χρησιμοποιηθεί στην διαδικασία της προσωμοιωμένης ανόπτησης καθώς και στην διαδικασία της αναρρίχισης λόφων. Η κλάση αυτή αποθηκεύει πληροφορίες σχετικά με τα μαθήματα, τους φοιτητές, τον συνολικό αριθμό εγγραφών, και των αριθμό των περιόδων που χρησιμοποιήθηκαν σε κάθε πρόβλημα Στο σχήμα 5.1 παρουσιάζεται ο κώδικας που χρησιμοποιήθηκε για την κατασκευή γράφου με την χρήση της βιβλιοθήκης της networkx.

Βασική κλάση η οποία κατασκευάστηκε είναι η κλάση solution, η οποία δημιουργεί αντικείμενα τα οποία θα διαχειρίζονται, ένα πρόβλημα και θα συμμετέχουν στην

διαδικασία βελτιστοποιήσης του. Στην κλάση υλοποιούνται οι επτά τελεστές γειτνίασης οι οποίοι ορίζονται ως συναρτήσεις. Επίσης ορίζεται η μεταβλητή cost, η οποία θα μεταβάλλεται ανάλογα με την τιμή της αντικειμενικής συνάρτησης, καθώς και η δομή p_periods, που για κάθε περίοδο αποθηκεύει τις εξετάσεις οι οποίοιες 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 στο κόστος, η μέθοδος can_be_moved, ελέγχει την εγκυρότητα μίας κίνησης και η μέθοδος select_move, επιλέγει και εφαρμόζει τυχαία έναν τελεστή γειτνίασης.

```
    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 τον αριθμό των κορυφών του γράφου. Μαζί με την αφαίρεση τον άνευ σημασίας εξετάσεων πραγματοποιήται και εύρεση των εξετάσεων που παρουσιάζουν ομοιότητα στις συγκρούσεις. Αυτές οι εξετάσεις θα προγραμματιστούν μία φορά, στην ίδια χρονική περίοδο.

```
    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 καταστευάστηκε η συνάρτηση compute_cost(), η οποία χρησιμοποιήται για τον υπολογισμό της αντικειμενικής τιμής της εκάστοτε κατάστασης. Ο κώδικας για την συνάρτηση παρουσιάζεται στο σχήμα 5.3. Η συνάρτηση compute_cost(), υπολογίζει το συνολικό κόστος, ενώ η συνάρτηση compute_normalized_cost(), κανονικοποιεί το κόστος ανα φοιτητή.

```
    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 πανομοιώτυπων φοιτητών. Η συνάρτηση identical_exams() είναι υπεύθυνη για την εύρεση τον πανομοιώτυπων εξετάσεων. Η πρώτη κατηγορία πανομοιώτυπων εξετάσεων που υπολογίζεται και αποθηκεύεται στην λίστα identical_type_1, αφορά τις εξετάσεις πού έχουν τις ίδιες γειτονικές εξετάσεις , με την ίδια τιμή βάρους ανά σύνδεση, μη έχοντας κοινούς φοιτητές ανάμεσα τους. Η δεύτερη κατηγορία πανομοιότυπων εξετάσεων που υπολογίζεται και αποθηκεύεται στην λίστα identical_type_2, αφορά τις εξετάσεις που που έχουν κοινές γείτονικές εξετάσεις, με ίδια τιμή βάρους ανά σύνδεση, έχοντας κοινό αριθμό φοιτητών μεταξύ τους, ενώ η τρίτη κατηγορία αφορά τις εξετάσεις που έχουν κοινούς φοιτητές,έχουν τις ίδιες γειτονικές εξετάσεις, μη διαθέτωντας την ίδια τιμή βάρους ανά σύνδεση. 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']:
                            ident_ipp=False
                if ident_ipp:
                    type_3.append((node_1,node_2))
```

```
29
30          for node_1,node_2 in type_1:
31              found_in=False
32              for index,fs in enumerate(identical_type_1):
33                      if node_1 in fs or node_2 in fs:
34                          identical_type_1[index].add(node_1)
35                          identical_type_1[index].add(node_2)
36                          found_in=True
37                          break
38
39              if not found_in:
40                  identical_type_1.append({node_1,node_2})
41
42          for node_1,node_2 in type_2:
43              found_in=False
44              for index,fs in enumerate(identical_type_2):
45                  if node_1 in fs or node_2 in fs:
46                      found_in=True
47                      identical_type_2[index].add(node_1)
48                      identical_type_2[index].add(node_2)
49                      break
50              if not found_in:
51                  identical_type_2.append({node_1,node_2})
52
53          for node_1,node_2 in type_3:
54              found_in=False
55              for index,fs in enumerate(identical_type_3):
56                  if node_1 in fs or node_2 in fs:
57                      found_in=True
58                      identical_type_3[index].add(node_1)
59                      identical_type_3[index].add(node_2)
60                      break
61              if not found_in:
62                  identical_type_3.append({node_1,node_2})
63          return identical_type_1,identical_type_2,identical_type_3
64
```

Σχήμα 5.3: Κώδικας εύρεσης πανομοιώτυπων εξετάσεων

5.2 Προσωμοιωμένη ανόπτηση

Στο σχήμα 5.4, παρουσιάζεται η διαδικασία της προσωμοιωμένης ανόπτησης υλοποιημένη με την βοήθεια της γλώσσας python. Η διαδικασία ορίστηκε ως μία συνάρτηση η οποία δεχόμενη ως όρισμο ένα αλφαριθμητικό που αντιστοιχεί στο όνομα ενός προβλήματος, πραγματοποιεί βελτιστοποιήση της λύσης του. Η μεταβλητή temp αντιστοιχεί στην τιμή θερμοκρασίας, η μεταβλήτη alpha, στον τελεστή ψύξης 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 θα έχει συμμετοχή στην μείωση της θερμοκρασίας, και ορίζει την καθυστέρηση στην διαδικασία μείωσης που θα προκύψει όταν έχει βρεθεί καλύτερη λύση στο πρόβλημα, καθυστερώντας την μείωση της θερμοκρασίας και δοκιμάζοντας αναζήτηση λύσεων με την ίδια τιμή θερμοκρασίας. Επίσης σε κάθε κύκλο της προσωμοιωμένης

ανόπτησης στον οποίο έχει πραγματοποιηθεί βελτιστοποίηση του προβλήματος, πραγματοποιήται ανθέρμανση της θερμοκρασίας. Η μεταβλητή temp επαναρχικοποιήται με την τιμή πέντε, συνεχίζοντας την διαδικασία στον ίδιο γύρω. Η μεταβλητή delta κρατάει την διαφορα στο κόστος μεταξύ δύο λύσεων και η μεταβλητή plateu, ελέγχει τον ρυθμό βελτίωσης ανά κύκλο. Η μεταβλητή delta, αποθηκεύει την τιμή της διαφοράς κόστους μίας υποψήφιας λύσης από την υπάρχουσα βέλτιστη λύση και σε περίπτωση ελαχιστοποιήσης του κόστους, πραγματοποιήται και αντικατάσταση της λύσης. Αν δεν παρατηρηθεί ελαχιστοποιήση στο κόστος και έχουν ολοκληρωθεί 20 κύκλοι εκτέλεσης ο αλγόριθμός τερματίζει την λειτουργεία του. Τέλος, ως συνθήκη τερματισμού έχει οριστεί ένα πεπερασμένο χρονικό όριο που εισάγει ο χρήστης. Η διαδικασία τερματίζει σε περίπτωση που ο συνολικός χρόνος εκτέλεσης υπερβαίνει το χρονικό όριο.

```
        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 προσωμοιωμένης ανόπτησης

5.3 Αναρρίχιση λόφων

Στο σχήμα 5.5, παρουσιάζεται η διαδικασία της αναρρίχισης λόφων με την βοήθεια της γλώσσας python. Με χρήση της μεθόδου execute_moves, πραγματοποιήται δημιουργία λύσης με την χρήση τελεστή γειτνίασης, η οποία και θα εκτελεστεί. Αν το κόστος της καινούργιας λύσης που δημιουργήθηκε είναι μεγαλύτερο του κόστους της τρέχουσας λύσης, τότε επαναφέρω την τρέχουσα λύση., ειδάλλως μετακινούμε στην καινούργια αυτή λύση. Αντίστοιχα και με την διαδικασία της προσωμοιωμένης ανόπτησης ως συνθήκη τερματισμού ορίζεται η υπέρβαση ένος πεπερασμένου χρονικού ορίου. Τέλος με την βοήθεια της μεθόδου permuting_periods, πραγματοποιήται μία αναδιάταξη των περίοδων με σκοπό την ελαχιστοποιήση του κόστους πριν την έναρξη της διαδικασίας της αναρρίχισης λόφων.

```
        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()
```

```
6             logo=' Permuting Periods '
7             print('-'*5+logo+'-'*5)
8             best=sol.permuting_periods(best)
9             print('-'*(len(logo)+10),end='\n\n')
10            while True:
11                moves=sol.execute_moves()
12                if len(moves)==0:
13                    if time()-start_timer>exec_time:
14                        break
15                    continue
16                rollback=dict()
17                for exam in moves:
18                    rollback[exam]=sol.solutions[exam]
19                sol.reposition(-1,-1,moves)
20                if sol.cost<best:
21                    best=sol.cost
22                    logging.info("Hill Climbing|New best solution found S
    ={} and time T={}`s".format(sol.cost,time()-start_timer))
23                else:
24                    sol.reposition(-1,-1,rollback)
25                if time()-start_timer>exec_time:
26                    break
27            sol.renew_solution(sol.solutions)
28
```

Σχήμα 5.5: Διαδικασία αναρείχισης 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. Ως χρόνος

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

Πίνακας 6.1: Χαρακτηριστικά υπολογιστικής μονάδας εκτέλεσης πειραμάτων

εκτέλεσης για κάθε πρόβλημα ορίστηκαν τα 1000 δευτερόλεπτα, ενώ διάφορα αποτελέσματα για διαφορετικούς χρόνους εκτέλεσης εμφανίζονται στον ιστοχώρο [18].

6.1 Αποτελέσματα προσομοιωμένης ανόπτησης

Αρχείο δεδομένων	Κόστος Λύσης
car92	116368(6.87)
car91	98103(5.32)
ear83	48823(43.39)
hec92	30360(10.75)
kfu93	82043(15.33)
lse91	34312(12.58)
pur93	253584(8.44)
rye93	128746(11,21)
sta83	95959(157.05)
tre92	45025(10.32)
uta92	100995(4.74)
ute92	73746(26.82)
yor83	47502(50.48)
ITC2007_1	98850(12.53)
ITC2007_2	88140(7.06)
ITC2007_3	115189(7.03)
ITC2007_4	88478(20.01)
ITC2007_5	208595(23.92)
ITC2007_6	55167(6.97)
ITC2007_7	170403(12.35)
ITC2007_8	98167(12.71)
ITC2007_9	18964(30.39)
ITC2007_10	48228(34.08)
ITC2007_11	
ITC2007_12	

Πίνακας 6.2: Αποτελέσματα προσωμοιωμένης ανόπτησης

6.2 Αποτελέσματα αλγορίθμου αναρρίχισης λόφων

Αρχείο δεδομένων	Κόστος Λύσης
car92	
car91	
ear83	
hec92	
kfu93	
lse91	
pur93	
rye93	
sta83	
tre92	
uta92	
ute92	
yor83	
ITC2007_1	
ITC2007_2	
ITC2007_3	
ITC2007_4	
ITC2007_5	
ITC2007_6	
ITC2007_7	
ITC2007_8	
ITC2007_9	
ITC2007_10	
ITC2007_11	
ITC2007_12	

Πίνακας 6.3: Αποτελέσματα αναρρίχεισης 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 προσωμοιωμένης ανόπτησης. Με την εισαγωγή των εξετάσεων και σπουδαστων οι οποίοι χαρακτηρίστηκαν ως άνευ σημασίας, επιτευχθηκέ μείωση του όγκου των προβλημάτων, ενώ προτάθηκε και ο τρόπος επίλυσης των προβλημάτων με χρήση των συνεκτικών τμημάτων, κάτι που οδήγησε στην διάσπαση προβλημάτων σε επημέρους υποπροβλήματα. Τέλος εξετάσθηκαν οι τελεστές γειτνίασης που χρησιμοποιήθηκαν για την δημιουργία καινούργιων λύσεων και προτάθηκαν κάποιες παραμέτροι στον αλγόριθμο της προσωμοιωμένης ανόπτησης με σκοπό την βελτίωση της απόδοσης του αλγορίθμου.

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 χρονοπραγαμματισμού εξετάσεων φυσικού περιεχομένοιυ, 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 αναρχά 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 επρόκειτω 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 παραμετροποιήται και θα δημιουργήται μέσω της εφαρμογης. Στο πρόβλημα θα μπορούσαμε να προσθέσουμε και τις δυνατότητες του προγραμματισμού με περιο-

ρισμούς(constaint programming) στην διαδικασία βελτιστοποιήσης του προβλήματος, κάτι που δύναται στο μέλων να οδηγήσει στην δημιουργία ενός προγράμματος εξετάσεων για κάποιο ίδρυμα.

Βιβλιογραφία

[1] "P and NP Problems." [Online]. Available: https://miro.medium.com/max/1400/1*CZCgxZBSnrUUW2qsXiKXQA.png

[2] S. Y. L. M. W. Carter, G. Laporte, "Examination timetabling: Algorithmic strategies and applications," *Journal of the Operational Research*, vol. 47, p. 373–383, 02 1996.

[3] "NP Hard Problems." [Online]. Available: http://www.scottaaronson.com/blog/?p=1720

[4] 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.

[5] "Hamilton Path." [Online]. Available: https://nitsri.ac.in/Department/Computer%20Science%20&%20Engineering/Lec4.pdf

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

[7] 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

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

[9] Vol. 1, 09 2021.

[10] "NetworkX." [Online]. Available: https://networkx.org/

[11] 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.

[12] "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

[13] "screte optimization algorithms with pascal programs," pp. 415–424, 1982.

[14] “A new efficient rlf-like algorithm for the vertex coloring problem,” November 2015.

[15] “kempe_chain_description.” [Online]. Available: https://planetmath.org/kempechain

[16] “Every planar map is four-colorable,” *Contemporary Mathematics, 98, With the collaboration of J. Koch., Providence, RI: American Mathematical Society*, 1989.

[17] “Sa. schaerf “two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling,” *Computers Operations Research*, 2021.

[18] “Vn thesis page.” [Online]. Available: https://github.com/vasnastos/Examination_Timetabling_DIT_UOI