ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΧΕΔΙΟ ΠΑΡΟΥΣΙΑΣΗΣ
ΑΤΟΜΙΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Μάιος 2010

Ατομική Διπλωματική Εργασία

ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΠΡΟΤΙΜΗΣΕΩΝ

Μαρία Χριστοφόρου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μέθοδοι Βελτιστοποίησης Προβλημάτων Προτιμήσεων

Μαρία Χριστοφόρου

Επιβλέπων Καθηγητής

Γιάννης Δημόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2014

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας Δρ. Γιάννη Δημόπουλο για την βοήθεια και τις συμβουλές που μου παρείχε καθ' όλη την διάρκεια της διπλωματικής μου εργασίας.

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την απέραντη συμπαράσταση και την στήριξη τους σε ολόκληρη τη διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου για την στήριξη τους, αλλά και για την βοήθεια που μου παρείχαν σε αυτό το σημαντικό κομμάτι της ζωής μου.

Περίληψη

Η παρούσα διπλωματική εργασία αποσκοπεί στην υλοποίηση ενός προγράμματος, και την χρησιμοποίηση διαφόρων εργαλείων ώστε να επιτευχθεί βελτιστοποίηση προβλημάτων προτιμήσεων. Μέσα από τη μελέτη διαφόρων άρθρων και την εξοικείωση με το εργαλείο το οποίο χρησιμοποιήθηκε, έγινε αντιληπτή η έννοια των CP-Networks και γενικά των προβλημάτων βελτιστοποίησης.

Ο στόχος της διπλωματικής εργασίας έχει υλοποιηθεί σε ένα ενιαίο πρόγραμμα. Για την υλοποίηση του προγράμματος αυτού χρησιμοποιήθηκε το εργαλείο Minisat+ το οποίο είχε σαν αποτέλεσμα τις βέλτιστες λύσεις του προβλήματος, και τον χρόνο τον οποίο χρειάστηκε για να βρεθούν οι λύσεις αυτές.

Στο πρώτο μέρος υλοποίησης υλοποιήθηκε ένα πρόγραμμα το οποίο δημιουργεί ένα τυχαίο cp-network, τις προτιμήσεις του χρήστη και τους περιορισμούς του προβλήματος. Σε ένα αρχείο, δημιουργείται αρχικά η εξίσωση με την οποία μεγιστοποιούσαμε το αποτέλεσμα, και μετά μπαίνουν οι προτιμήσεις και οι περιορισμοί σε μορφή εξίσωσης.
Το αρχείο αυτό, έχει συγκεκριμένη μορφή, έτσι ώστε να είναι η σωστή είσοδος για το Minisat+.
Στο συνέχεια, παίρνουμε την έξοδο του Minisat+, βρίσκουμε την βέλτιστη λύση του προβλήματος και την κάνουμε περιορισμό, ώστε να ξανατρέξουμε το αρχείο και να βρεθεί η δεύτερη βέλτιστη λύση.

Για να μην χρειάζεται ο χρήστης να τρέχει τα προγράμματα ξεχωριστά για να του εμφανίσει τις επόμενες βέλτιστες λύσεις, δημιουργήσαμε ένα bash αρχείο, το οποίο περιέχει όλες τις εντολές των προγραμμάτων που χρειαζόμαστε για να εμφανίσουμε τις βέλτιστες λύσεις. Το μόνο που χρειάζεται να κάνει ο χρήστης είναι να πληκτρολογήσει πόσες φορές θέλει να του εμφανιστεί η βέλτιστη λύση. Οι λύσεις αυτές, εμφανίζονται σε ένα αρχείο με την σειρά. Δηλαδή, η πρώτη λύση είναι η καλύτερη, η δεύτερη η δεύτερη καλύτερη και ούτω καθεξής.

Στο μέρος αυτό, έχουμε την επιλογή να δώσει και ο χρήστης δικά του αρχεία, το οποία θα διαβαστούν και θα του παρουσιάσουμε την βέλτιστη λύση του προβλήματος που δίνει. Το πρόγραμμα αυτό διαβάζει την είσοδο του χρήστη, δημιουργεί το αρχείο με την αντικειμενική συνάρτηση και τους περιορισμούς, και τέλος χρησιμοποιούμε το εργαλείο Minisat+, για να εμφανίσουμε τις λύσεις του προβλήματος.

Η διπλωματική εργασία έχει υλοποιηθεί σε γλώσσα προγραμματισμού C. Επίσης, χρησιμοποιήθηκαν τα UNIX του Πανεπιστημίου Κύπρου για την μεταγλώττιση των προγραμμάτων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Οι προτιμήσεις παίζουν σημαντικό ρόλο στη ζωή μας. Προτιμήσεις πάνω σε ένα πεδίο πιθανών επιλογών, ταξινομούν τις προτιμήσεις αυτές έτσι ώστε μια προτιμώμενη επιλογή να προηγείται μιας λιγότερο προτιμώμενης επιλογής.

Οι προτιμήσεις μπορούν να αντιπροσωπευτούν σε δύο τρόπους: σε ποσοτικές («σε μια κλίμακα από το 0 μέχρι το 1 προτιμώ την Cyprus Airways στο επίπεδο 0.8») ή ποιοτικές («Προτιμώ την Cyprus Airways από την Ryanair»). Επίσης, μπορούν να εκφραστούν χωρίς όρους («Ανεξαρτήτως ώρας, προτιμώ την Cyprus Airways από την Ryanair») ή με όρους(«Αν είναι αργά το βράδυ, προτιμώ την Cyprus Airways από την Ryanair»). Οι χρήστες όπως είναι γνωστό, νιώθουν πιο άνετα να χρησιμοποιούν ποιοτικές και με όρους προτιμήσεις.

1.2 Προτασιακή Ικανοποιησιμότητα και Προβλήματα ικανοποιήσης

Μια έκφραση προτασιακής λογικής, που μπορεί επίσης να ονομαστεί Boolean έκφραση, μπορεί να δομηθεί από μεταβλητές, τελεστές και παρενθέσεις. Οι τελεστές που μπορούν να χρησιμοποιηθούν είναι: AND (σύζευξη, και συμβολίζεται με $\wedge$) OR (διάζευξη, και συμβολίζεται με $\vee$) και NOT (άρνηση, και συμβολίζεται με $\neg$).

Μια έκφραση είναι ικανοποιήσιμη, αν μπορεί η τιμή της να είναι αληθής (TRUE), όταν ανατεθούν οι κατάλληλες τιμές (TRUE ή FALSE) στις μεταβλητές.

Στα προβλήματα προτασιακής ικανοποιησιμότητας (Boolean Satisfiability Problem ή SAT), προσπαθεί να καθοριστεί εάν υπάρχει μία λύση η οποία ικανοποιεί ένα συγκεκριμένο Boolean πρόβλημα. Δηλαδή, ορίζει εάν οι μεταβλητές του συγκεκριμένου Boolean προβλήματος, μπορεί να ανατεθούν με τέτοιο τρόπο ώστε η λύση του προβλήματος να είναι αληθής (TRUE). Αν δεν μπορεί να γίνει καμία τέτοια ανάθεση, τότε η λύση του προβλήματος είναι Ψευδής (FALSE).
Στην πρώτη περίπτωση, αν υπάρχει δηλαδή κάποια ανάθεση των μεταβλητών, το πρόβλημα είναι ικανοποιήσιμο, αλλιώς είναι μη-ικανοποιήσιμο (δεύτερη περίπτωση).
Για παράδειγμα, η έκφραση “a AND NOT b”, είναι ικανοποιήσιμη επειδή οι τιμές του a και b μπορεί να είναι TRUE και FALSE, αντίστοιχα. Έτσι, το “a AND NOT b”, ισούται με TRUE.
Σε αντίθεση, η έκφραση “a AND NOT a ”, είναι ένα μη – ικανοποιήσιμο πρόβλημα. Αν το a έχει την τιμή TRUE, τότε το πρόβλημα “a AND NOT a ” θα έχει την τιμή FALSE. Το ίδιο ισχύει και αν το a έχει την τιμή FALSE.

Το SAT, είναι ένα από τα παραδείγματα ενός προβλήματος NP-πληρότητας. Αυτό σημαίνει ότι δεν υπάρχει κανένας γνωστός αλγόριθμος που να μπορεί να λύσει όλες τις περιπτώσεις του SAT.
Μια κατηγορία αλγορίθμων ονόματι SAT solvers, μπορούν να επιλύσουν αποτελεσματικά ένα αρκετά μεγάλο υποσύνολο των παραδειγμάτων αυτών.

Μια διαφορά των προβλημάτων ικανοποίησης περιορισμών (CSPs – Constraint Satisfaction Problems) και της προτασιακής ικανοποιησιμότητας (SAT), είναι στις τιμές που παίρνουν. Στα προβλήματα προτασιακής ικανοποιησιμότητας, έχουμε δυαδικό πεδίο τιμών (δηλαδή παίρνει τιμές 0 ή 1), και μη δυαδικούς περιορισμούς, ενώ στα προβλήματα ικανοποίησης περιορισμών, έχουμε δυαδικούς περιορισμούς και μη δυαδικό πεδίο τιμών.
Ένα πρόβλημα ικανοποίησης περιορισμών, περιέχει ένα σύνολο από μεταβλητές, η κάθε μια με ένα πεδίο τιμών και ένα σύνολο από περιορισμούς στις επιτρεπόμενες τιμές για συγκεκριμένα υποσύνολα μεταβλητών.
Δοθέντος ενός προτασιακού τύπου, το πρόβλημα προτασιακής ικανοποιησιμότητας καθορίζει αν υπάρχει ανάθεση των τιμών αληθείας στις μεταβλητές που κάνουν ολόκληρο το τύπο αληθή(TRUE).

Μπορούμε να κωδικοποιήσουμε προβλήματα ικανοποίησης περιορισμών σε προβλήματα προτασιακής ικανοποιησιμότητας, χρησιμοποιώντας άμεση κωδικοποίηση, με το πιο κάτω τρόπο:

Συσχετίζουμε μια προτασιακή μεταβλητή X_{ij} με κάθε τιμή j που μπορεί να ανατεθεί στην CSP μεταβλητή X_i. Επίσης, έχουμε διατάξεις που εξασφαλίζουν ότι κάθε CSP μεταβλητή θα πάρει μία τιμή. Προαιρετικά, έχουμε διατάξεις που εξασφαλίζουν ότι κάθε μεταβλητή θα πάρει μία και μόνο μία μεταβλητή, Τέλος, έχουμε δυαδικές διατάξεις που αποκλείουν τις τιμές που δεν μας εξυπηρετούν. Για παράδειγμα, αν $X_1 = 2$ και $X_3 = 1$ απαγορεύεται, τότε έχουμε τη διάταξη $\neg X_{12} \vee \neg X_{31}$. [3]

Ένα κυρίαρχο πρόβλημα στη Τεχνητή Νοημοσύνη (Artificial Intelligence) είναι η επίλυση προβλημάτων. Αυτό που ζητάμε από το σύστημα, είναι να καταλαβαίνει τις προτιμήσεις μας έναντι των εναλλακτικών επιλογών και να βρίσκει τις καλύτερες εφικτές επιλογές για εμάς.

Προτιμήσεις πάνω σε κάποιο πεδίο δυνατών επιλογών, διατάσσουν τις επιλογές αυτές έτσι ώστε μια πιο επιθυμητή επιλογή να προηγείται μιας λιγότερο επιθυμητής. Για διαφορετικά πεδία δυνατών επιλογών, έχουμε διαφορετικά αποτελέσματα. [2]

1.3 Σκοπός Διπλωματικής Εργασίας

Σκοπός της διπλωματικής εργασίας ήταν η βελτιστοποίηση των προβλημάτων προτιμήσεων. Δηλαδή, δοθέντος ενός CP-Net, να βρεθούν όλες οι πιθανές βέλτιστες λύσεις του προβλήματος αυτού. Μέσα από τη μελέτη διαφόρων άρθρων και την εξοικείωση με το εργαλείο το οποίο χρησιμοποιήθηκε, έγινε αντιληπτή η έννοια των CP-Networks και γενικά των προβλημάτων βελτιστοποίησης.
Για την υλοποίηση του προγράμματος αυτού χρησιμοποιήθηκε το εργαλείο Minisat+ το οποίο είχε σαν αποτέλεσμα τις βέλτιστες λύσεις του προβλήματος, και τον χρόνο τον οποίο χρειάστηκε για να βρεθούν οι λύσεις αυτές.

1.4 Δομή Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία έχει χωριστεί σε τρεις φάσεις μελέτης. Στην πρώτη φάση, ήταν η μελέτη διαφόρων άρθρων για να γίνουν κατανοητές οι βασικές έννοιες και οι ορισμοί. Η μελέτη ήταν αρχικά για την κατανόηση των προτιμήσεων του χρήστη και των CP-Nets. Στη συνέχεια έγινε εξοικείωση με το εργαλείο Minisat+, τρέχοντας δικά μου έτοιμα παραδείγματα, ώστε να γίνει κατανοητή η έξοδος του εργαλείου αυτού.

Στη δεύτερη φάση, έγινε η ανάπτυξη και η υλοποίηση των μεθόδων βελτιστοποίησης των διάφορων προβλημάτων προτιμήσεων όπως περιγράφονται στο Κεφάλαιο 5 . Η φάση αυτή υλοποιήθηκε σε διάφορα μέρη.

Στο πρώτο μέρος, υλοποιήθηκε η δημιουργία ενός τυχαίου CP-Net. Δημιουργείται επίσης, ένα αρχείο με τις προτιμήσεις του χρήστη για το συγκεκριμένο CP-Net και ένα αρχείο με περιορισμούς για τις προτιμήσεις αυτές. Συγκεκριμένα, δίνει ο χρήστης τον αριθμό που θα αποτελείται το CP-Net, και αυτό δημιουργείται τυχαία. Στο τέλος του προγράμματος αυτού, σαν έξοδο έχουμε ένα αρχείο το οποίο περιέχει την αντικειμενική εξίσωση μαζί με περιορισμούς, το οποίο θα μπει σαν είσοδο στο εργαλείο Minisat+.
Επίσης, εκτός από το τυχαίο CP-Net που μπορεί να δημιουργηθεί, έχουμε την επιλογή, ο χρήστης να δώσει το αρχείο προτιμήσεων. Δηλαδή, ο χρήστης μας δίνει το δικό του CP-Net, και εμείς από αυτό, εξάγουμε όλες τις εφικτές λύσεις αλλά και την βέλτιστη. Έχοντας το αρχείο των προτιμήσεων και το αρχείο τον περιορισμών, μπορούμε να δημιουργήσουμε το αρχείο το οποίο θα μπει στο εργαλείο Minisat+.
Στο δεύτερο μέρος, μπαίνει σαν είσοδος το αρχείο στο Minisat+. Σαν αποτέλεσμα, έχουμε όλες τις εφικτές λύσεις του προβλήματος, την βέλτιστη λύση και το χρόνο που χρειάστηκε Minisat+ για να υπολογίσει την κάθε λύση ξεχωριστά.
Στο τρίτο μέρος της υλοποίησης, παίρνουμε την εφικτή και βέλτιστη λύση και την κάνουμε περιορισμό, έτσι ώστε το εργαλείο Minisat+ να βρει την δεύτερη καλύτερη εφικτή και βέλτιστη λύση.
Τέλος, δημιουργήθηκε ένα αρχείο, ώστε ο χρήστης να έχει την δυνατότητα να επιλέξει πόσες φορές θέλει να τρέξει το πρόγραμμα για να βρεθούν οι βέλτιστες λύσεις του προβλήματος.

Μετά από την αξιολόγηση έγινε η πειραματική αξιολόγηση. Με βάση τα αποτελέσματα της πειραματικής αξιολόγησης έγινε εξαγωγή συμπερασμάτων. Επιπρόσθετα, σημαντικό ρόλο για τις μετρήσεις αυτές έχει παίξει και ο χρόνος εκτέλεσης των πειραμάτων. Δηλαδή, τον χρόνο που χρειαζόταν κάθε φορά το εργαλείο Minisat+ να υπολογίσει όλες τις εφικτές λύσεις και την βέλτιστη.

Η παρούσα διπλωματική εργασία έχει υλοποιηθεί σε γλώσσα προγραμματισμού C.
Επίσης, χρησιμοποιήθηκαν τα UNIX του πανεπιστημίου Κύπρου για την μεταγλώττιση των προγραμμάτων.

Κεφάλαιο 2

Προβλήματα Προτιμήσεων και Ικανοποίησης Περιορισμών

2.1 Ορισμός Προβλημάτων Προτιμήσεων

Ένα κυρίαρχο πρόβλημα στη Τεχνητή Νοημοσύνη (Artificial Intelligence) είναι η επίλυση προβλημάτων. Αυτό που ζητάμε από το σύστημα, είναι να καταλαβαίνει τις προτιμήσεις μας έναντι των εναλλακτικών επιλογών και να βρίσκει τις καλύτερες εφικτές επιλογές για εμάς.

Προτιμήσεις πάνω σε κάποιο πεδίο δυνατών επιλογών, διατάσσουν τις επιλογές αυτές έτσι ώστε μια πιο επιθυμητή επιλογή να προηγείται μιας λιγότερο επιθυμητής. Για διαφορετικά πεδία δυνατών επιλογών, έχουμε διαφορετικά αποτελέσματα.
Σαν αποτέλεσμα , αναφερόμαστε στα στοιχεία που βρίσκονται στο σετ των επιλογών. Το σετ των αποτελεσμάτων αλλάζει από ένα πεδίο σε άλλο. Παραδείγματα σετ πιθανών αποτελεσμάτων μπορούν να είναι πιθανές πτήσεις, κινητά τηλέφωνα, φωτογραφικές. Μια ταξινόμηση μπορεί να είναι ολική (όλα τα ζευγάρια αποτελεσμάτων είναι συγκρίσιμα), μερική(δεν υπάρχουν δύο αποτελέσματα που να είναι ίσα προτιμώμενα) ή ασθενές.

Μια σχέση προτίμησης σε ένα σετ Ω είναι μια μεταβατική δυαδική σχέση $\geq$ στο Ω. Αν για κάθε ο, ό $\in$ Ω, ή ο $\geq$ ό ή ό $\geq$ ο, τότε το $\geq$ είναι ολική ταξινόμηση. Αλλιώς, είναι μερική ταξινόμηση (για παράδειγμα, κάποια αποτελέσματα δεν είναι συγκρίσιμα). Μια σχέση προτίμησης είναι ισχυρή αν είναι αντι-συμμετρική, δηλαδή, αν ο $\geq$ ό τότε ό $\not\geq$ ο. Αλλιώς, είναι ασθενής.
Δοθέντος μίας ασθενής ταξινόμησης $\geq$, μπορούμε να ορίσουμε μία ισχυρή ταξινόμηση > ως εξής: ο > ό αν και μόνο αν ο $\geq$ ό, αλλά όχι ό $\not\geq$ ο.

Δυστυχώς όμως, το να δουλεύεις με προτιμήσεις δεν είναι πάντα εύκολο για πάρα πολλούς λόγους. Κάποιοι από αυτοί είναι:

Η γνωστική δυσκολία να προσδιορίσεις μια σχέση προτίμησης. Αν μας ενδιαφέρει μόνο ένα στοιχείο, τότε δεν δημιουργείται κανένα πρόβλημα. Για παράδειγμα, αν κάποιος θέλει να αγοράσει ένα συγκεκριμένο iPhone κινητό τηλέφωνο, τότε το μόνο που τον ενδιαφέρει είναι η τιμή. Ωστόσο, οι προτιμήσεις συνήθως είναι πιο περίπλοκες. Ακόμα και στο πιο πάνω παράδειγμα, μας ενδιαφέρει η εγγύηση, η αποστολή του κινητού τηλεφώνου. Όταν αρχίσουν να έχουν σημασία για εμάς, πολλές πτυχές του αποτελέσματος, τότε η ταξινόμηση ακόμα και δύο αποτελεσμάτων μπορεί να είναι γνωστικά δύσκολη, επειδή πρέπει να έχουμε υπόψη μας όλες τις εξαρτήσεις μεταξύ των μεταβλητών.

2.2 Μοντέλα Προτιμήσεων

Όταν έχουμε μοντελοποίηση προτιμήσεων, προκύπτουν δύο ερωτήσεις:

1. Τι είναι το μοντέλο
2. Τι ερωτήσεις θέλουμε να ρωτήσουμε για αυτό το μοντέλο

Ένα μοντέλο για προτιμήσεις, είναι μια ταξινόμηση $\geq$ πάνω στο σετ των πιθανών αποτελεσμάτων Ω είναι η απάντηση στην πρώτη ερώτηση.

Για την δεύτερη ερώτηση, η απάντηση διαφέρει από το τι θέλουμε το μοντέλο να κάνει. Ερωτήσεις που μπορούν να γίνουν είναι να βρεθεί η βέλτιστη λύση, να βρεθεί η βέλτιστη λύση ικανοποιώντας συγκεκριμένους περιορισμούς, ταξινόμηση δύο αποτελεσμάτων, σύγκριση δύο αποτελεσμάτων και άλλα.

Όταν έχουμε το μοντέλο και το ερώτημα μας, χρειαζόμαστε ένα αλγόριθμο που θα απαντήσει το ερώτημα δίνοντας το μοντέλο.

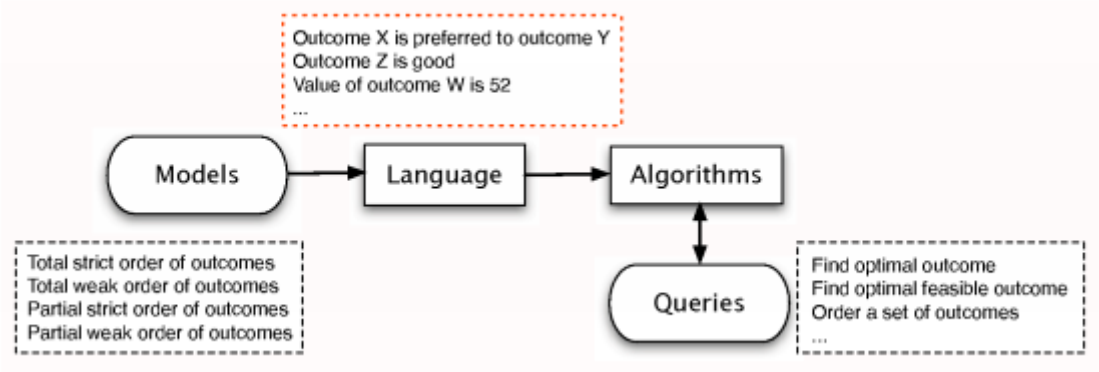
Το σημαντικό εδώ είναι ότι ένα μοντέλο προτιμήσεων πρέπει να προσδιορίζεται για κάθε νέο χρήστη, αποκτώντας τις πληροφορίες που χρειάζεται. Προσδιορίζοντας το μοντέλο είναι ένα από τα μεγαλύτερα ζητήματα στον χειρισμό των προτιμήσεων.

Μια προσέγγιση, είναι να ρωτήσουμε απευθείας τον χρήστη να περιγράψει το μοντέλο. Έτσι, δημιουργείται μια ταξινόμηση. Αυτό όμως δεν είναι πρακτικό αν το Ω είναι μεγάλο. Έτσι, δημιουργούμε μια γλώσσα, που είναι μια συλλογή από πιθανές δηλώσεις που θα κάνουν πιο εύκολη την περιγραφή των μοντέλων.

Για να δημιουργήσουμε ένα μοντέλο προτιμήσεων έχουμε πέντε (5) στοιχεία:

- Το μοντέλο που περιγράφει τη δομή που πραγματικά θέλουμε
- Τα ερωτήματα για το μοντέλο που πρέπει να απαντηθούν
- Ο αλγόριθμος για να υπολογίσει τις απαντήσεις στα ερωτήματα αυτά
- Η γλώσσα που προσδιορίζει τα μοντέλα
- Συνάρτηση ερμηνείας που χαρτογραφεί εκφράσεις σε μοντέλα

Τα στοιχεία αυτά απεικονίζονται στο πιο κάτω μοντέλο, το οποίο δείχνει τις εξαρτήσεις κάθε επιλογής:

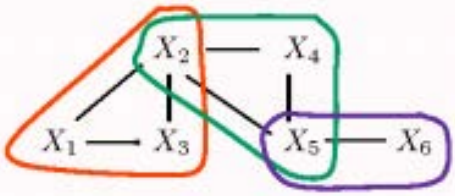


Το τελευταίο στοιχείο για να δημιουργήσουμε το μοντέλο προτιμήσεων είναι η αναπαράσταση της πληροφορίας που μας δίνει ο χρήστης.

Η αναπαράσταση είναι μια δομή που καταγράφει τις πληροφορίες στο μοντέλο, και θα χρησιμοποιηθεί από τους αλγόριθμους για να βρεθούν οι απαντήσεις στα ερωτήματα. Η πρώτη επιλογή για αναπαράσταση, είναι η ίδια η γλώσσα.

Ένα παράδειγμα είναι το GAI (Generalized Additive Independence) δίκτυο. Το δίκτυο αυτό, είναι ένα γράφημα όπου οι κόμβοι αντιστοιχούν στα χαρακτηριστικά στο X.

$$V(X_1,\ldots,X_6) = g_1(X_1,X_2,X_3)+ g_2(X_2,X_4,X_5)+ g_3(X_5,X_6)$$



Αριστερά βλέπουμε την συνάρτηση GAI και δεξιά, το αντίστοιχο GAI δίκτυο. Μια ακμή συνδέει τους κόμβους που αντιστοιχούν σε ένα ζεύγος χαρακτηριστικών X_i, $X_j \in X$, αν X_i και X_j προκύπτουν από κοινού σε κάποιο παράγοντα, $\{ X_i, X_j \} \supseteq Z_l$, όπου $l \in \{1,...,k\}$.
Οι χρήστες, μπορούν να παρέχουν τις δηλώσεις προτιμήσεων με διάφορους τρόπους. Μια απλή κατηγορία δηλώσεων αντιστοιχεί σε σύγκριση ανάμεσα σε εναλλακτικές λύσεις, όπως "Μου αρέσει αυτό το αυτοκίνητο από το άλλο", ή, "Προτιμώ αυτή την πτήση από την άλλη". Παρόμοιες είναι και οι δηλώσεις του τύπου "Θέλω ένα αυτοκίνητο σαν και αυτό, αλλά σε μπλε". Οι δηλώσεις αυτές είναι εύκολο να ερμηνευθούν, όμως μας παρέχουν μια μικρή ποσότητα πληροφοριών για το μοντέλο προτιμήσεων του χρήστη. Μια άλλη κατηγορία, είναι οι γενικευμένες δηλώσεις, όπως "Προτιμώ ένα αυτόματο μίνιβαν παρά ένα με ταχύτητες", ή, "Προτιμώ οποιοδήποτε κόκκινο αυτοκίνητο από οποιοδήποτε μπλε".

Με βάση τα παραπάνω, χρησιμοποιούμε CP-Networks επειδή έχουμε μερική σειρά στα αποτελέσματα και όχι ολική σειρά. Η γλώσσα που χρησιμοποιείται στην προσέγγιση των CP-Networks, είναι η γλώσσα των δηλώσεων των προτιμήσεων στις τιμές των μονών χαρακτηριστικών. Η συνάρτηση διερμηνείας που χρησιμοποιείται από τα CP-Networks αντιστοιχούν στη διερμηνεία ceteris paribus (όλα τα υπόλοιπα παραμένουν ίδια) των δηλώσεων, και στη συνέχεια έχουμε μεταβατικό κλείσιμο της σειράς προτίμησης που προκαλείται από τις δηλώσεις αυτές. [2]

2.3 Ικανοποίηση Περιορισμών

Στην τεχνητή νοημοσύνη, η ικανοποίηση περιορισμών είναι η διαδικασία εύρεσης μιας λύσης σε ένα σετ περιορισμών που επιβάλλουν όρους που οι μεταβλητές πρέπει να ικανοποιούν. Έτσι, μια λύση είναι ένα σετ από τιμές για τις μεταβλητές που ικανοποιούν όλους τους περιορισμούς.

Οι τεχνικές που χρησιμοποιούνται στην ικανοποίηση περιορισμών εξαρτώνται από το είδος των περιορισμών που θα εξεταστούν. Συχνά χρησιμοποιούνται περιορισμοί με πεπερασμένο πεδίο τιμών, στο σημείο που τα προβλήματα ικανοποίησης περιορισμών συνήθως προσδιορίζονται με προβλήματα βασισμένα σε περιορισμούς με πεπερασμένο πεδίο τιμών. Συνήθως τα προβλήματα αυτά λύνονται με αναζήτηση, και συγκεκριμένα με υπαναχωρήσεις ή τοπική αναζήτηση. Μια άλλη μέθοδος είναι η μέθοδος περιορισμού διάδοσης. Συνήθως χρησιμοποιούνται για να αποδείξουν ότι το πρόβλημα είναι μη ικανοποιήσιμο. Επίσης χρησιμοποιούνται σε συνεργασία με την αναζήτηση, ώστε να κάνουν ένα πρόβλημα πιο εύκολο στην επίλυση.

Άλλα είδη περιορισμών είναι σε πραγματικούς ή ρητούς αριθμούς. Η επίλυση των προβλημάτων αυτών γίνεται με εξάλειψη μεταβλητών ή τον simplex αλγόριθμο.

2.4 Προβλήματα Ικανοποίησης Περιορισμών και SAT

Τα προβλήματα ικανοποίησης περιορισμών είναι μαθηματικά προβλήματα που ορίζονται ως ένα σετ από αντικείμενα όπου η κατάσταση τους πρέπει να ικανοποιεί ένα αριθμό από περιορισμούς. Τα προβλήματα αυτά αντιπροσωπεύουν τις οντότητες σε ένα πρόβλημα σαν μια συλλογή από πεπερασμένους περιορισμούς πάνω στις μεταβλητές, τα οποία λύνονται από μεθόδους ικανοποίησης περιορισμών. προβλήματα ικανοποίησης περιορισμών συχνά παρουσιάζουν ψηλή πολυπλοκότητα, απαιτώντας ένα συνδυασμό από ευρηστικά και συνδυαστικές μεθόδους αναζήτησης για να μπορούν να επιλυθούν σε εύλογο χρονικό διάστημα. Τα προβλήματα δυαδικής ικανοποίησης μπορούν να θεωρηθούν μια μορφή προβλημάτων ικανοποίησης περιορισμών.

Επισήμως, ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται ως μια τριπλή <X,D,C>, όπου

$X = \{X_1, \dots, X_n\}$ είναι ένα σετ από μεταβλητές,

$D = \{D_1, \dots, D_n\}$ είναι ένα σετ από πεδία τιμών,

$X = \{C_1, \dots, C_m\}$ είναι ένα σετ από περιορισμούς.

Μια αξιολόγηση είναι συνεπής αν δεν παραβιάζει κανένα από τους περιορισμούς. Μια αξιολόγηση είναι ολοκληρωμένη αν περιέχει όλες τις μεταβλητές. Μια αξιολόγηση είναι

λύση αν είναι συνεπής και ολοκληρωμένη. Μια τέτοια αξιολόγηση επιλύει το πρόβλημα ικανοποίησης περιορισμών.

Ανάμεσα στους δύο τύπους προβλημάτων, δηλαδή τα προβλήματα ικανοποίησης περιορισμών και τα προβλήματα προτασιακής ικανοποίησης υπάρχουν κάποιες διαφορές. Μια από τις πιο άμεσες διαφορές είναι ότι τα προτασιακής ικανοποίησης έχουν δυαδικά πεδία τιμών και μη δυαδικούς περιορισμούς, ενώ τα προβλήματα ικανοποίησης περιορισμών έχουν δυαδικούς περιορισμούς και μη δυαδικά πεδία τιμών.

Μια μέθοδος εξερεύνησης της σχέσης μεταξύ των προβλημάτων ικανοποίησης περιορισμών και των προβλημάτων προτασιακής ικανοποίησης, είναι η μελέτη χαρτογράφησης μεταξύ των προβλημάτων αυτών. Αφού κάθε πρόβλημα είναι NP-πλήρες, μπορούμε να μειώσουμε το ένα στο άλλο σε πολυώνυμο χρόνο.

Δοθέντος μιας προτασιακής φόρμουλας, το πρόβλημα ικανοποίησης είναι να προσδιοριστεί εάν υπάρχει μια ανάθεση με αληθής τιμές στις μεταβλητές που κάνουν όλη την φόρμουλα αληθή.

Εξετάζουμε την χαρτογράφηση των προβλημάτων ικανοποίησης περιορισμών σε προβλήματα προτασιακής ικανοποίησης. Συνδέουμε μια προτασιακή μεταβλητή, x_{ij}, με κάθε τιμή j μου μπορεί να ανατεθεί στην μεταβλητή X_i του προβλήματος προτασιακής ικανοποίησης. Έχουμε διατάξεις που μας εξασφαλίζουν ότι κάθε μεταβλητή του προβλήματος προτασιακής ικανοποίησης θα πάρει μία τιμή. Προαιρετικά, έχουμε διατάξεις που μας εξασφαλίζουν ότι κάθε μεταβλητή δεν θα πάρει πάνω από μία τιμή. Τέλος, έχουμε δυαδικές διατάξεις που αποκλείουν οποιαδήποτε δεν μας είναι αναγκαία.

Κεφάλαιο 3

CP-Networks και Ψευδο-δυαδική Βελτιστοποίηση

3.1 Εισαγωγή

Η εξαγωγή πληροφοριών για τις προτιμήσεις των χρηστών είναι συνήθως μια επίπονη διαδικασία, για αυτό και έχουν αναπτυχθεί διάφορες τεχνικές για να βοηθήσουν στην απόσπαση των πληροφοριών αυτών.

Τα CP-Nets, είναι μια από τις τεχνικές αυτές, και χρησιμοποιείται για να προσδιοριστούν οι προτιμήσεις των χρηστών, χρησιμοποιώντας ceteris paribus δηλώσεις προτιμήσεων. Δηλαδή, πάντα αλλάζει μία μόνο μεταβλητή, και οι υπόλοιπες μένουν σταθερές.

3.2 Σχέσεις Προτίμησης

Υποθέτουμε ότι ο κόσμος μπορεί να θεωρηθεί σαν ένας αριθμός από καταστάσεις S και σε κάθε κατάσταση s, υπάρχει ένας αριθμός από ενέργειες A_s που μπορούν να εκτελεστούν. Κάθε ενέργεια, όταν εκτελείται σαν κατάσταση, έχει ένα συγκεκριμένο αποτέλεσμα. Το σύνολο αποτελεσμάτων ορίζεται σαν O. Μια κατάταξη προτίμησης είναι μια ολική σειρά $\succeq$ σε ένα σετ αποτελεσμάτων: $o_1 \succeq o_2$ σημαίνει ότι το o_1 προτιμάται εξίσου ή περισσότερο από το o_2. Χρησιμοποιούμε $o_1 \succ o_2$ όταν το o_1 προτιμάται αυστηρά περισσότερο από τον αποφασίζοντα παρά το o_2.

Ο στόχος της λήψης αποφάσεων είναι, γνωρίζοντας μια συγκεκριμένη κατάσταση, να επιλεγεί η ενέργεια που θα έχει το προτιμότερο αποτέλεσμα.

Συχνά, για κάποια κατάσταση s, ένα συγκεκριμένο αποτέλεσμα O, δεν μπορεί να προκύψει από καμία ενέργεια $\alpha \in A_s$. Τα αποτελέσματα που μπορούν να ληφθούν, ονομάζονται εφικτά αποτελέσματα.

Στα CP-Networks, τα ceteris paribus συστατικά, εξασφαλίζουν ότι οι δηλώσεις που κάνει κάποιος είναι σχετικά ασθενής.

Θεωρούμε ότι έχουμε δύο μεταβλητές A και B, οι οποίες είναι κατά προτίμηση ανεξάρτητες, έτσι ώστε οι προτιμήσεις για τις τιμές A και B να αξιολογηθούν ξεχωριστά. Για παράδειγμα, υποθέτουμε ότι $\alpha_1 > \alpha_2$ και $b_1 > b_2$. Σαφώς, το $\alpha_1 b_1$ είναι το πλέον προτιμότερο αποτέλεσμα και το $\alpha_1 > \alpha_2 b_2$ είναι το λιγότερο προτιμότερο αποτέλεσμα. Αν όμως οι περιορισμοί δυνατότητας καθιστούν αδύνατο το $\alpha_1 b_1$, πρέπει να ικανοποιηθούμε με $\alpha_1 b_2$ ή $\alpha_2 b_1$. Όμως, δεν μπορούμε να πούμε πιο από τα δύο προτιμάται περισσότερο χρησιμοποιώντας αυτές τις αξιολογήσεις. Ωστόσο, χρησιμοποιώντας ισχυρές συνθήκες, μπορούμε να κατασκευάσουμε μια εξίσωση, όπου θα ανατίθενται βάρη στις διάφορες μεταβλητές. Μια τέτοια διάσπαση της συνάρτησης προτιμήσεων, μας επιτρέπει να αναγνωρίσουμε τα προτιμότερα αποτελέσματα πολύ εύκολα.

3.3 CP-Networks

Το μοντέλο των CP-Networks είναι παρόμοιο με αυτό του Bayesian Network. Η διαφορά είναι ότι στο CP-Network, οι σχέσεις μεταξύ των κόμβων είναι σχετικά αδύναμες.

Για κάθε μεταβλητή X_i, ζητάμε από τον χρήστη να ορίσει ένα σετ από μεταβλητές οι οποίες θα είναι οι γονείς $Pa(X_i)$, και θα επηρεάζουν την προτίμηση της μεταβλητής αυτής πάνω σε διάφορες τιμές του X_i. Έτσι, δοθέντος μιας συγκεκριμένης τιμής ανάθεσης στο $Pa(X_i)$, ο χρήστης πρέπει να μπορεί να προσδιορίσει την σειρά προτίμησης για τις τιμές του X_i, όταν όλα τα υπόλοιπα παραμένουν ίσα(ceteris paribus). Επίσης, ζητάμε από το χρήστη να προσδιορίσει τις προτιμήσεις του στις τιμές του X_i για όλα τα στιγμιότυπα των μεταβλητών που υπάρχουν στο σετ $Pa(X_i)$. Χρησιμοποιούμε τις πληροφορίες αυτές για να δημιουργήσουμε ένα σχολιασμένο κατευθυνόμενο γράφο, του οποίου οι κόμβοι αντιπροσωπεύουν τις μεταβλητές του προβλήματος και κάθε κόμβος X_i έχει $Pa(X_i)$ σαν άμεσο πρόγονό του. Κάθε κόμβος X_i είναι σχολιασμένος με ένα πίνακα (Conditional Preference Table - CPT), που περιγράφει τις προτιμήσεις του χρήστη πάνω στις τιμές τις μεταβλητής X_i, δοθέντος όλων των συνδυασμών των γονικών τιμών.

Η δομή αυτή, είναι δίκτυα προτίμησης με όρους (Conditional Preference Networks). Ένα CP-Network πάνω στις μεταβλητές $V = \{X_1,...,X_n\}$ είναι ένας κατευθυνόμενος γράφος G πάνω στα $X_1,...,X_n$, του οποίου οι κόμβοι είναι σχολιασμένοι με υπό όρους πίνακες προτιμήσεων $CPT(X_i)$ για κάθε $X_i \in V$.

Μια από τις κύριες ιδιότητες των CP-Networks είναι, δεδομένου ενός CP-Network N, μπορούμε να προσδιορίσουμε το καλύτερο αποτέλεσμα μεταξύ της κατάταξης προτιμήσεων που ικανοποιεί το N.

Διαισθητικά, για να παράγουμε ένα βέλτιστο αποτέλεσμα, πρέπει απλά να σαρώσουμε το δίκτυο, από την αρχή μέχρι το τέλος, δηλαδή, από τους προγόνους προς τους απογόνους, βάζοντας σε κάθε μεταβλητή την προτιμότερη τιμή δεδομένου του στιγμιότυπου των γονέων του. Το δίκτυο αυτό, πράγματι αποφασίζει ένα μοναδικό βέλτιστο αποτέλεσμα.

Η σημασιολογία του ceteris paribus, ενός CP-Network, επιτρέπει να χρησιμοποιήσουμε απευθείας πληροφορίες στο πίνακα της μεταβλητής X, για να τροποποιήσει ή να γυρίσει την τιμή του X στο αποτέλεσμα, ώστε να αποκτήσει άμεσα ένα βελτιωμένο προτιμότερο αποτέλεσμα ή ένα χειρότερο. Μια σειρά από βελτιωμένες αναστροφές από κάποιο αποτέλεσμα σε ένα άλλο, μας παρέχει μια απόδειξη ότι ένα αποτέλεσμα είναι προτιμότερο ή όχι, από κάποιο άλλο, σε όλες τις κατατάξεις που ικανοποιούν το δίκτυο.

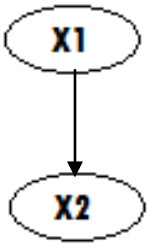
3.4 Άκυκλα CP-Networks

Ένα CP-Network ορίζεται ως το σύνολο των κατατάξεων προτιμήσεων που είναι σύμφωνα με το σετ των περιορισμών προτίμησης που επιβάλλονται από τους πίνακες.

Η σημασιολογία του CP-Network, δεν το αναγκάζει να είναι άκυκλο. Όμως, η ακυκλότητα του δικτύου αυτόματα αναθέτει ένα σημαντικό κομμάτι στο μοντέλο: το μοντέλο γίνεται ικανοποιήσιμο. Ένα κυκλικό μοντέλο υπονοεί ότι όλες οι μεταβλητές που υπάρχουν σε αυτό είναι εξίσου σημαντικές.

3.5 Ψευδο-δυαδική Βελτιστοποίηση

Στο υποκεφάλαιο αυτό, θα εξηγήσουμε πως μετατρέπουμε τα CP-Nets σε ψευδοδυαδική βελτιστοποίηση. Αρχικά, χρησιμοποιούμε το πιο κάτω παράδειγμα, που είναι απλό, για να γίνει πιο κατανοητή η μετατροπή.



Στο πιο πάνω CP-Net, έχουμε δύο μεταβλητές, την X1 και X2. Επειδή είναι κατευθυνόμενος γράφος, η μεταβλητή X1 προηγείται της X2, και η X2 εξαρτάται από την X1. Το πεδίο τιμών των δύο αυτών μεταβλητών είναι το 0 και το 1 αφού είναι Boolean μεταβλητές. Για δική μας ευκολία, αντί 0 και 1 χρησιμοποιούμε 1 και 2. Αντί NOT X1 χρησιμοποιούμε X_{12}.

Το X1 είναι η αρχική μας μεταβλητή, η ρίζα στο γράφο. Δεν εξαρτάται από καμία άλλη μεταβλητή. Η μεταβλητή αυτή χωρίζεται σε δύο, την X_{11} και την X_{12}, αφού έχουμε δυαδικό πεδίο τιμών.

Άρα ο περιορισμός που δημιουργείται είναι της μορφής:

: $X_{11} > X_{12}$, δηλαδή ο χρήστης προτιμά την X_{11} από την X_{12}.

Η μεταβλητή X2 είναι λίγο πιο περίπλοκη αφού είναι εξαρτώμενη μεταβλητή. Με βάση την X_{11} ο χρήστης προτιμά την X_{21}, και με βάση την X_{12} ο χρήστης προτιμά την X_{22}.

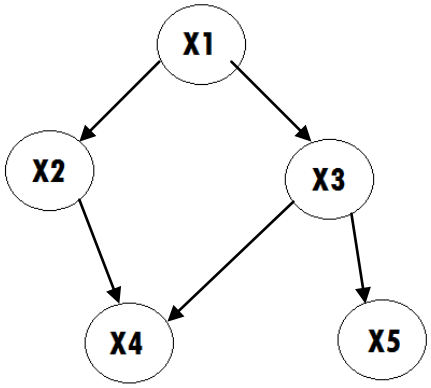
Άρα, δημιουργούνται οι εξής προτιμήσεις με βάση τις μεταβλητές μας:

$X_{11} : X_{21} > X_{22}$

$X_{12} : X_{22} > X_{21}$

Για τις μεταβλητές X_{21} και X_{22} η απόφαση θα παρθεί από τον πατέρα τους. Αν δηλαδή στο πρόβλημα επιλεγεί σαν αρχική μεταβλητή το X_{11} και δεν υπάρχει περιορισμός για το X_{21}, τότε θα επιλεγεί η μεταβλητή X_{21}. Το ίδιο ισχύει και για την μεταβλητή X_{12} και X_{21}.

Ένα πιο περίπλοκο παράδειγμα είναι το πιο κάτω:



Στον πιο πάνω γράφο, παρατηρούμε ότι οι μεταβλητές X1 και X3 έχουν δύο παιδιά, και η μεταβλητή X4 έχει δύο πατέρες. Ότι ισχύει στο παραπάνω παράδειγμα ισχύει και σε αυτό, με εξαίρεση την X4 που έχει δύο πατέρες.

Οι μεταβλητές X1 και X3 δεν επηρεάζονται κάπου, αφού η X1 δεν εξαρτάται από καμία μεταβλητή μιας και είναι η ρίζα του γράφου και η X3 εξαρτάται μόνο από την X1.

Η μεταβλητή X4, έχει πατέρες τις μεταβλητές X2 Κι X3. Άρα η μεταβλητή X4 εξαρτάται από τις δύο αυτές μεταβλητές. Άρα στον σχηματισμό των προτιμήσεων, οι μεταβλητή αυτή πρέπει να εξαρτάται και από τις δύο αυτές μεταβλητές. Δηλαδή, θα σχηματίσουμε 4 διαφορετικές προτιμήσεις ανάλογα με την κάθε περίπτωση που μπορεί να προκύψει.

X_{21} ^ X_{31}: $X_{41} > X_{42}$

X_{21} ^ X_{32}: $X_{42} > X_{41}$

X_{22} ^ X_{31}: $X_{42} > X_{41}$

X_{22} ^ X_{32}: $X_{41} > X_{42}$

Για την πρώτη προτίμηση, με βάση το X_{21} και X_{31}, ο χρήστης προτιμά την μεταβλητή X_{42} >από την μεταβλητή X_{41}.

3.6 Παραδείγματα CP-Nets

Παράδειγμα 1:

Έχουμε ένα απλό CP-Network που εκφράζει τις προτιμήσεις πάνω στο δείπνο. Το δίκτυο αυτό περιλαμβάνει δύο μεταβλητές, S και W, που αντιπροσωπεύουν την σούπα και το κρασί αντίστοιχα.

Ο χρήστης προτιμά αυστηρά ψαρόσουπα (S_f) παρά χορτόσουπα (S_v), ενώ, το είδος του κρασιού εξαρτάται από το είδος της σούπας που θα του σερβιριστεί. Προτιμά κόκκινο κρασί (W_r), αν θα σερβιριστεί χορτόσουπα και λευκό κρασί (W_w), αν του σερβιριστεί ψαρόσουπα.

$$S_f \succ S_v$$

S_f	$W_w \succ W_r$
S_v	$W_r \succ W_w$

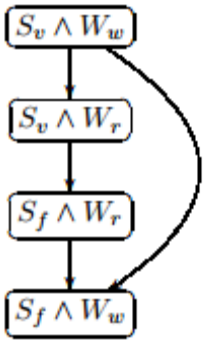
Στον πιο κάτω γράφος προτιμήσεων, ένα τόξο από το o_i στο o_j, υποδεικνύει ότι μια προτίμηση από το o_j στο o_i, καθορίζεται απευθείας από τους πιο πάνω πίνακες στο CP-Network.

Όπως φαίνεται στον γράφο αυτό το $S_f \wedge W_w$, είναι η καλύτερη προτίμηση, δηλαδή αυτό που επιθυμεί ο χρήστης και είναι ο τελευταίος κόμβος, ενώ η χειρότερη επιλογή, που είναι και ο πρώτος κόμβος στο γράφο, είναι να σερβιριστεί χορτόσουπα με λευκό κρασί.

Δεδομένου ενός CP-Network, μπορούμε να έχουμε πρόσβαση σε κάθε αποτέλεσμα όσο αφορά τους όρους προτιμήσεων που παραβιάζει.

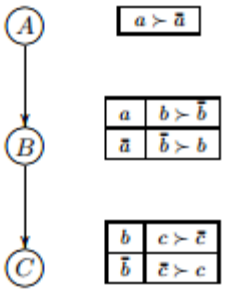
Στο παράδειγμά μας, το αποτέλεσμα $S_f \wedge W_w$ δεν παραβιάζει κανένα περιορισμό προτίμησης, το $S_f \wedge W_r$ παραβιάζει τον περιορισμό για το κρασί, το $S_v \wedge W_w$ παραβιάζει τον περιορισμό προτίμησης για την σούπα και το $S_v \wedge W_w$ παραβιάζει και τους δύο περιορισμούς. Η σημασιολογία του ceteris paribus εμμέσως διασφαλίζει ότι παραβιάζοντας την προτίμηση για την σούπα, είναι χειρότερο από το να παραβιάσεις την προτίμηση για το

κρασί. $S_f \succ W_r \succ S_v \succ W_w$. Αυτό συμβαίνει επειδή οι προτιμήσεις των γονέων έχουν μεγαλύτερη προτεραιότητα από τις προτιμήσεις των παιδιών.



Παράδειγμα 2:

Έχουμε το πιο κάτω CP-Network με τους πίνακες προτιμήσεων:



Οι πιο κάτω κατατάξεις, είναι οι μόνες που ικανοποιούν αυτό το δίκτυο:

$$abc \succ ab\bar{c} \succ a\bar{b}\bar{c} \succ \overbrace{a\bar{b}c \succ \bar{a}\bar{b}\bar{c}} \succ \bar{a}\bar{b}c \succ \bar{a}bc \succ \bar{a}b\bar{c}$$
$$abc \succ ab\bar{c} \succ a\bar{b}\bar{c} \succ \underbrace{\bar{a}\bar{b}\bar{c} \succ a\bar{b}c} \succ \bar{a}\bar{b}c \succ \bar{a}bc \succ \bar{a}b\bar{c}$$

Τα δύο αποτελέσματα που δεν είναι εντελώς διατεταγμένα, είναι $\bar{a}\bar{b}\bar{c}$ και $a\bar{b}c$. Αν αφαιρέσουμε κάποιο από αυτά τα δύο, από κάθε μία από τις πιο πάνω αλυσίδες αποτελεσμάτων, μπορούμε να κινηθούμε από το ένα αποτέλεσμα στο επόμενο στην αλυσίδα,

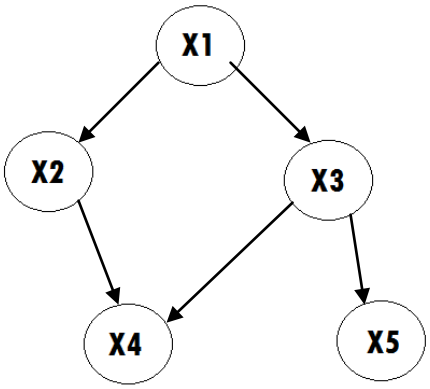
αντιστρέφοντας την τιμή ακριβώς μίας τιμής ανάλογα με τις προτιμήσεις που βρίσκονται στους πίνακες προτιμήσεων.

Για παράδειγμα, για να κινηθούμε από το πρώτο αποτέλεσμα σε αυτές τις ακολουθίες (abc) στο επόμενο (ab$\bar{c}$), χρησιμοποιούμε την προτίμηση c $>$ $\bar{c}$ δοθέντος b, για να αποδείξουμε ότι το δεύτερο αποτέλεσμα δεν είναι προτιμότερο από το πρώτο. Αντιστρέφουμε το C, σε μια λιγότερο προτιμότερη τιμή, δοθέντος του στιγμιότυπου b από τους γονείς B. Αντιστρόφως, μετακινούμαστε προς τα πίσω μέσα στην ακολουθία, γυρίζοντας το $\bar{c}$ στο δεύτερο αποτέλεσμα σε c, αποκτώντας έτσι ένα προτιμότερο πρώτο αποτέλεσμα.

Όπως αναφέρθηκε πιο πάνω, είναι λιγότερο προτιμότερο να παραβιάσουμε τους περιορισμούς προτίμησης για τις μεταβλητές που είναι πατέρες, παρά να παραβιάσουμε τους περιορισμούς προτίμησης για οποιοδήποτε από τα παιδιά. Αυτό είναι πολύ σημαντικό για το ceteris paribus.

Θεωρούμε δύο αποτελέσματα $\bar{a}\bar{b}\bar{c}$ και a$\bar{b}$c, τα οποία είναι μη διατεταγμένα από το πιο πάνω CP-Network. Το αποτέλεσμα $\bar{a}\bar{b}\bar{c}$ παραβιάζει την προτίμηση πάνω στην τιμή του A, ενώ το a$\bar{b}$c παραβιάζει τις προτιμήσεις στις τιμές B και C. Το A είναι πρόγονος των B και C, και το CP-Network δεν μας προσδιορίζει ποιο από αυτά τα αποτελέσματα είναι προτιμότερο. Αν και διαισθητικά η προτίμηση στο A έχει μεγαλύτερη προτεραιότητα από αυτή των B και C, δύο ή περισσότερες παραβιάσεις μικρότερης προτεραιότητας προτιμήσεων, δεν προτιμούνται από το να παραβιαστεί μία μόνο μεγαλύτερης προτεραιότητας προτίμηση.

Παράδειγμα 3:



Οι προτιμήσεις του χρήστη για το συγκεκριμένο παράδειγμα με δύο πατέρες είναι:

:X11 > X12

X11: X21 >X22

X12: X22 >X21

X11: X31 >X32

X12: X32 >X31

X21 ^ X31 : X41 > X42

X21 ^ X32 : X42 > X41

X22 ^ X31 : X42 > X41

X22 ^ X32 : X41 > X42

X31: X51 >X52

X32: X52 >X51

Παρατηρούμε ότι στις προτιμήσεις του χρήστη, η μεταβλητή X4 εξαρτάται από δύο μεταβλητές, την X2 και X3, που είναι και οι πατέρες της. Άρα, οι αποφάσεις θα παρθούν βασιζόμενες και στις δύο αυτές μεταβλητές, για να επιλεγεί η βέλτιστη και εφικτή λύση του συγκεκριμένου παραδείγματος.

Κεφάλαιο 4

Minisat+

4.1 Το εργαλείο Minisat

Το Minisat είναι ένας μινιμαλιστικός, ανοικτού κώδικα, προβλημάτων ικανοποίησης επιλυτής.

Ένας SAT (satisfiability problem) επιλυτής, μπορεί να προσδιορίσει αν είναι δυνατόν να βρεθούν αναθέσεις σε δυαδικές μεταβλητές που μπορούν να κάνουν μια έκφραση αληθή, αν η έκφραση αυτή είναι γραμμένη με AND, OR, NOT, παρενθέσεις, και Boolean μεταβλητές. Αν είναι ικανοποιήσιμο, οι περισσότεροι επιλυτές, μπορούν να δείξουν επίσης ένα σετ από αναθέσεις που κάνουν την έκφραση αληθή.

Το Minisat είναι ένα εργαλείο το οποίο είναι εύκολο να τροποποιηθεί. Μπορεί ο κάθε χρήστης να το αλλάξει όπως αυτός επιθυμεί για να του βγάζει όπως αυτός θέλει τα αποτελέσματα. Είναι πολύ αποδοτικό και είναι σχεδιασμένο για ένταξη.

Όπως πολλοί SAT επιλυτές, το Minisat απαιτεί η είσοδος του να είναι σε συζευκτική κανονική μορφή. Η μορφή αυτή αποτελείται από τα εξής στοιχεία:

- Όρος: ένας όρος είναι είτε μια δυαδική μεταβλητή (π.χ. X4) ή μια αρνητική δυαδική μεταβλητή (π.χ. NOT X4, γραμμένο ως –X4).
- Διάταξη: μια διάταξη είναι ένα σετ από ένα ή περισσότερους όρους, συνδεδεμένα με OR (γραμμένο ως |).
- Έκφραση: μια έκφραση είναι ένα σετ από μία ή περισσότερες διατάξεις, και κάθε μία είναι συνδεδεμένη με AND (γραμμένο ως &).

Ένα παράδειγμα είναι:

(x1 | -x5 | x4) &
(-x1 | x5 | x3 | x4) &
(-x3 | x4).

4.2 Το εργαλείο Minisat+

Το εργαλείο αυτό, υλοποιήθηκε σε ένα διαγωνισμό το 2005 και είναι βασισμένο στο Minisat, το οποίο είναι ένας επιλυτής κατασκευασμένος για ερευνητές και προγραμματιστές. Το Minisat+, υποστηρίζει ψευδο-δυαδικούς περιορισμούς, αλλά και κυκλικούς, βασισμένους στην είσοδο του επιλυτή(solver) αυτού.

Το Minisat+, έχει ειδικό τύπο αρχείου για να μπορεί να βγάλει σωστά αποτελέσματα.
Αρχικά στο αρχείο υπάρχει η αντικειμενική συνάρτηση, την οποία μεγιστοποιούμε ή ελαχιστοποιούμε ανάλογα με το τι αποτελέσματα θέλουμε. Η αντικειμενική συνάρτηση είναι της μορφής:

min: + C_1*X_{11} + C_2*X_{21} + C_3*X_{31} + C_4*X_{41}+ C_5*X_{51} + C_6*X_{61};

Όπου C είναι το βάρος κάθε μεταβλητής, το οποίο είναι και διαφορετικό για κάθε μεταβλητή ξεχωριστά και εξαρτάται από το επίπεδο που βρίσκεται στο γράφο και πόσα παιδιά έχει σαν πατέρας.
Όπου X είναι η μεταβλητή του CP-Net. Στην αντικειμενική συνάρτηση μπαίνουν μόνο οι μεταβλητές που βρίσκονται στις προτιμήσεις του χρήστη.

Στη συνέχεια του αρχείου, έχουμε τους περιορισμούς της κάθε μεταβλητής, δηλαδή ότι δεν μπορεί να είναι με τον εαυτό της. Τέλος, έχουμε τους περιορισμούς ανάλογα με τις προτιμήσεις του χρήστη, που στη συγκεκριμένη περίπτωση είναι τυχαίοι, αφού δημιουργούμε τυχαίο γράφο. Οι περιορισμοί αυτοί, έχουν την ίδια μορφή, η οποία είναι:

+ C_1*X_{11} + C_2*X_{12} >=+1;

όπου C είναι το βάρος κάθε μεταβλητής, το οποίο είναι και διαφορετικό για κάθε μεταβλητή ξεχωριστά και εξαρτάται από το επίπεδο που βρίσκεται στο γράφο και πόσα παιδιά έχει σαν πατέρας.
Όπου X είναι η μεταβλητή που βρίσκεται στο CP-Net.

Κεφάλαιο 5

Υλοποίηση και Πειραματική Αξιολόγηση

5.1 Random Uniform CSP Generator

Για την δημιουργία περιορισμών ικανοποιησιμότητας, χρησιμοποιήθηκε το εργαλείο Random Uniform CSP Generator, το οποίο είναι γραμμένο σε γλώσσα C.

Πολλοί ερευνητές που ασχολούνται με τα προβλήματα ικανοποίησης περιορισμών χρησιμοποιούν τυχαίες περιπτώσεις για να αξιολογήσουν τους αλγόριθμους ικανοποίησης περιορισμών. Παρόλο που γενικά έχει συμφωνηθεί ότι η τελική αξιολόγηση ενός αλγόριθμου για προβλήματα ικανοποίησης περιορισμών πρέπει να γίνεται σε προβλήματα που υπάρχουν στον «πραγματικό» κόσμο, υπάρχει παγκόσμια ομοφωνία για την αξία του «εργαστηριακού» πειράματος σε τυχαία προβλήματα.

Τα τυχαία προβλήματα προσφέρουν διάφορα πλεονεκτήματα για την αξιολόγηση της απόδοσης των αλγορίθμων για προβλήματα ικανοποίησης περιορισμών. Κάποια από αυτά είναι:

- Μπορούν να παραχθούν μεγάλες ποσότητες, έτσι ώστε να μπορούν να αναφερθούν στατιστικά μέσα και διακυμάνσεις.
- Είναι εύκολο να αλλαχθούν συστηματικά οι παράμετροι του Random Uniform CSP Generator, για να παρατηρήσουμε πως η απόδοση ενός αλγόριθμου έχει σχέση με τον αριθμό των μεταβλητών για παράδειγμα.

- Είναι εύκολο να βρεθούν παράμετροι που να παράγουν προβλήματα όπου το 50% να μπορούν να λυθούν. Γενικά, τέτοια προβλήματα είναι δύσκολα και τείνει να τονίζει τις διαφορές στην απόδοση του αλγορίθμου.

Αν και υπάρχουν διάφορες εκδοχές του προγράμματος αυτού, οι περισσότεροι ερευνητές τα τελευταία χρόνια χρησιμοποιούν ένα απλό μοντέλο, το οποίο δέχεται τέσσερις (4) παραμέτρους:

1. Τον αριθμό των μεταβλητών του προβλήματος.
2. Τον αριθμό των τιμών στο πεδίο ορισμού κάθε μεταβλητής. Κάθε μεταβλητή έχει ένα πεδίο τιμών με το ίδιο μέγεθος.
3. Τον αριθμό των περιορισμών. Όλοι οι περιορισμοί είναι δυαδικοί. Οι περιορισμοί επιλέγονται τυχαία από μια ομοιόμορφη κατανομή. Ο αριθμός αυτός μπορεί να είναι ορισμένος είτε σαν ακέραιος, είτε σαν κλάσμα ανάμεσα στο 0 και το 1. Για παράδειγμα, αν ένα πρόβλημα έχει 20 μεταβλητές, τότε ο μέγιστος αριθμός περιορισμών είναι 20*19/2=190. Ένα συγκεκριμένο πρόβλημα που μπορεί να οριστεί με ένα αριθμό περιορισμού ισούται με 95 ή 0.5. Η C συνάρτηση, παίρνει την παράμετρο αυτή σαν ακέραιο, έτσι δεν χρειάζεται να γίνει στρογγυλοποίηση ή περικοπή.
4. Η στενότητα κάθε περιορισμού. Όλοι οι περιορισμοί έχουν την ίδια στενότητα. Η στενότητα αναφέρεται στον αριθμό των ζευγών τιμών, τα οποία δεν αναγνωρίζονται από τον περιορισμό. Τα συγκεκριμένα ζεύγη εκλέγονται τυχαία από μια ομοιόμορφη κατανομή. Η στενότητα μπορεί να οριστεί είτε σαν ακέραιος, είτε σαν κλάσμα ανάμεσα στο 0 και το 1. Για παράδειγμα, αν ένα πρόβλημα έχει μεταβλητές με μέγεθος πεδίου ορισμού 5, τότε ο μέγιστος αριθμός των ζευγών τιμών που δεν αναγνωρίζονται από τον περιορισμό είναι 5*5=25. Ένα συγκεκριμένο πρόβλημα μπορεί να οριστεί με στενότητα 5 ή 0.2. Η C συνάρτηση, παίρνει την παράμετρο αυτή σαν ακέραιο, έτσι δεν χρειάζεται να γίνει στρογγυλοποίηση ή περικοπή.

Ο κώδικας στη C παράγει τυχαία παραδείγματα με βάση το πιο πάνω μοντέλο.

Ο κώδικας με λίγα λόγια δουλεύει ως εξής:

Ο κώδικας στη C περιέχει ένα ρητά ορισμένο ψευδο-τυχαίο αριθμό γεννήτριας. Στηριζόμενη στη συνάρτηση rand() ή random(), είμαστε σίγουροι ότι δεν θα υπάρξουν διπλότυπες περιπτώσεις. Χρησιμοποιείται η ran2, αφού πιστεύουν ότι βγάζει τέλειους τυχαίους αριθμούς.

Ένα παράδειγμα εισόδου του προγράμματος αυτού είναι:

Urbcsp 100 10 10 10 100 100

100: αριθμός μεταβλητών

10: μέγεθος πεδίου τιμών κάθε μεταβλητής

10: αριθμός περιορισμών

10: αριθμός ασυμβίβαστων ζευγών τιμών σε κάθε περιορισμό

100: ένας αρνητικός αριθμός σημαίνει να αρχίσει μια νέα ακολουθία από ψευδο-τυχαίους αριθμούς. Ένας θετικός αριθμός σημαίνει συνέχισε στην ίδια ακολουθία.

100: πόσες φορές θα τρέξει

Ένα παράδειγμα εξόδου του προγράμματος αυτού είναι:

```
Instance 99
 17  19: (6 6) (2 0) (3 8) (5 7) (9 6) (2 7) (5 6) (8 2) (9 9) (4 8)
 57  94: (0 3) (0 1) (8 0) (2 2) (7 6) (9 1) (8 4) (3 0) (9 2) (8 8)
 10  28: (5 0) (4 6) (9 2) (8 2) (1 2) (3 5) (4 8) (1 1) (3 3) (4 0)
  1  90: (1 4) (4 5) (5 3) (7 8) (7 2) (7 1) (0 0) (0 4) (0 5) (1 9)
 55  64: (2 0) (5 9) (0 8) (0 2) (9 0) (5 1) (5 4) (2 7) (1 6) (5 0)
  9  32: (0 5) (1 1) (6 3) (1 8) (2 4) (5 6) (3 5) (2 8) (9 9) (5 3)
  3  12: (0 7) (3 6) (8 8) (0 8) (6 1) (1 4) (2 0) (3 2) (4 1) (3 0)
 52  69: (5 5) (7 8) (8 2) (1 8) (9 7) (9 2) (9 3) (3 1) (9 9) (4 8)
 11  59: (9 0) (0 1) (8 7) (5 8) (7 4) (2 2) (2 1) (8 4) (9 8) (6 9)
 14  44: (9 0) (2 4) (3 3) (5 0) (2 7) (1 4) (3 9) (9 6) (6 8) (7 0)
```

Για να δημιουργήσουμε περιορισμούς Bessiere, χρησιμοποιούμε την έξοδο του Random Uniform CSP Generator. Παίρνουμε κάθε instance ξεχωριστά και κατασκευάζουμε τους περιορισμούς με τον πιο κάτω τρόπο:

Στο πιο πάνω παράδειγμα, έχουμε 100 μεταβλητές, 10 περιορισμούς, ένα πεδίο τιμών από 0 μέχρι 9 και 10 ασυμβίβαστα ζευγάρια τιμών σε κάθε περιορισμό.

Πριν από το «:», έχουμε το id των μεταβλητών, και μετά τα ασυμβίβαστα ζευγάρια τιμών για κάθε περιορισμό.

Παίρνουμε κάθε μεταβλητή με τα ζευγάρια και δημιουργούμε τον περιορισμό των μεταβλητών X_{17} και X_{19} ως εξής:

$X_{176} + X_{196} + X_{172} + X_{190} + X_{173} + X_{198} + X_{175} + X_{197} + X_{179} + X_{196} + X_{172} + X_{197} + X_{175} + X_{196} + X_{178} + X_{199} + X_{179} + X_{194} + X_{178}$

Αυτό γίνεται για όλες τις γραμμές του συγκεκριμένου Instance 99, και για όλα τα Instances του προβλήματος αυτού.

5.2 Μέρος 1

Αρχικά, στην πρώτη φάση της υλοποίησης, υλοποιήθηκε το κομμάτι στο οποίο δημιουργείται το αρχείο με την αντικειμενική συνάρτηση, το οποίο θα μπει σαν είσοδο στο εργαλείο Minisat+. Δημιουργούμε το CP-Net, το οποίο είναι τυχαίο, τις προτιμήσεις του χρήστη και τους περιορισμούς πάνω στις προτιμήσεις αυτές.

Για να είμαστε πάντα σίγουροι ότι το πρόγραμμα δουλεύει με οποιοδήποτε αριθμό και αν δώσει ο χρήστης για την δημιουργία του γράφου, βάζουμε κάποια όρια στην υλοποίηση. Ο χρήστης μπορεί να δώσει όσο μεγάλο αριθμό θέλει για να δημιουργηθεί το τυχαίο CP-Net. Όμως, δεν θα γίνεται πάντα αυτό. Αν ο χρήστης δώσει μέχρι 999 μεταβλητές, τότε θα δημιουργηθεί ένας γράφος με 999 μεταβλητές. Αν όμως ο χρήστης δώσει πάνω από 1000 μεταβλητές, τότε θα δημιουργηθεί ένας πιο μικρός γράφος. Πιο συγκεκριμένα, αν ο χρήστης δώσει από 1000 μέχρι 1999 μεταβλητές, τότε ο αριθμός αυτός θα διαιρεθεί με το 10. Αν ο χρήστης δώσει από 2000 μέχρι 9999 μεταβλητές, τότε ο αριθμός αυτός θα διαιρεθεί με το 100. Και αν ο χρήστης δώσει από 10000 μεταβλητές, τότε ο αριθμός αυτός θα διαιρεθεί με το 1000. Έτσι, θα έχουμε πάντα ένα πιο μικρό γράφο, άρα και ένα πιο μικρό πρόβλημα για την εύρεση της βέλτιστης λύσης.

Στη φάση αυτή, μπορεί και ο χρήστης να δώσει ένα αρχείο με προτιμήσεις και ένα με περιορισμούς, και να δημιουργηθεί το αρχείο το οποίο θα μπει στο Minisat+.

5.2.1 Εντολές Εκτέλεσης του Προγράμματος

Για την εκτέλεση του προγράμματος αυτού είναι η εντολή:
./diplwmatiki

Όπου «diplwmatiki» είναι το όνομα του αρχείου με τον κώδικα που υλοποιήσαμε για να κατασκευαστεί το CP-Net, το οποίο είναι τυχαίο, οι προτιμήσεις του χρήστη και οι περιορισμοί πάνω στις προτιμήσεις αυτές.

./user
Όπου «user» είναι το όνομα του αρχείου το οποίο θα διαβάζει τα αρχεία προτιμήσεων και περιορισμών από τον χρήστη και θα δημιουργεί CP-Net από το αρχείο των προτιμήσεων.

Για την εκτέλεση του εργαλείου Minisat+ είναι η εντολή:
./minisat+ objfunction.txt > output.txt

Όπου «objfunction.txt» είναι το αρχείο το οποίο δημιουργείται από το πρόγραμμα που υλοποιήσαμε, και περιέχει την αντικειμενική συνάρτηση και τους περιορισμούς του χρήστη. Το αρχείο «output.txt» περιέχει την έξοδο του εργαλείου Minisat+.

5.2.2 Δομές δεδομένων

Στο κομμάτι αυτό δημιουργήθηκαν οι πιο κάτω τύποι δεδομένων:

Τύπος Δεδομένων 1:
int varoi[variables];

Ο πίνακας αυτός έχει δημιουργηθεί, για να φυλαχτούν μέσα οι μεταβλητές που θα αποτελούν το CP-Net.

Τύπος Δεδομένων 2:
int graph[variables][variables];

Ο δυσδιάστατος πίνακας graph, αντιπροσωπεύει το CP-Net, που δημιουργείται στην μορφή άκυκλου γράφου. Επέλεξα να κατασκευάσω τον γράφο χρησιμοποιώντας ένα δυσδιάστατο πίνακα.

Τύπος Δεδομένων 3:
char *metavlites[variables];

Στον πίνακα αυτό, θα μπουν οι μεταβλητές που θα δημιουργηθούν για το CP-Net. Κάθε θέση του πίνακα θα είναι και μία μεταβλητή, και θα υπάρχουν όσες μεταβλητές επιλέξει ο χρήστης.

Τύπος Δεδομένων 4:

int varoi[variables];

Στον μονοδιάστατο αυτό πίνακα αυτό θα μπουν τα βάρη των μεταβλητών.

5.2.3 Τρόπος Λειτουργίας του Προγράμματος

Αρχικά, στο πρόγραμμα ο χρήστης δίνει την είσοδο του προγράμματος, δηλαδή τον αριθμό των μεταβλητών που θα αποτελείται το CP-Net. Αφού δώσει την είσοδο αυτή, τότε δημιουργείται το τυχαίο CP-Net, οι προτιμήσεις του χρήστη πάνω στο συγκεκριμένο CP-Net, και οι περιορισμοί πάνω στις προτιμήσεις αυτές.

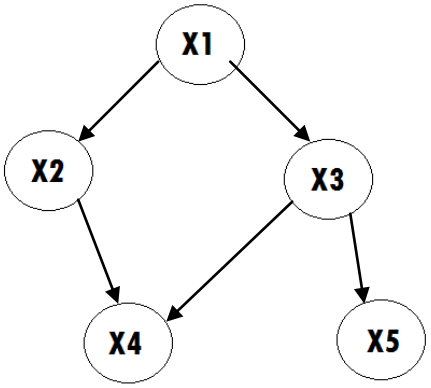
Όταν δημιουργηθεί το CP-Net, βρίσκουμε τα βάρη κάθε μεταβλητής που θα μπει στην αντικειμενική συνάρτηση. Όσο πιο ψηλά βρίσκεται η μεταβλητή πάνω στο CP-Net, τόσο μεγαλύτερο είναι το βάρος της. Τα βάρη βρίσκονται με τον εξής τρόπο:

Πρώτον, μετράμε τα επίπεδα του γράφου. Αρχίζουμε από το τελευταίο επίπεδο, δηλαδή τα φύλλα του γράφου και βάζουμε σαν βάρος ένα (1). Άρα, αρχίζουμε από κάτω προς τα πάνω και βάζουμε τα βάρη ανάλογα με το επίπεδο στο οποίο βρίσκονται.

Επίπεδο 3: Αρχικό Βάρος 3

Επίπεδο 2: Αρχικό Βάρος 2

Επίπεδο 1: Αρχικό Βάρος 1

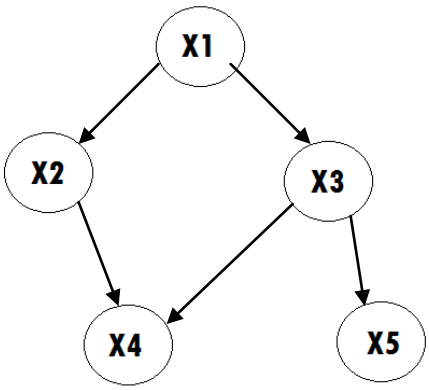


Στη συνέχεια, προσθέτουμε στο προηγούμενο επίπεδο τα παιδιά του επιπέδου που βρισκόμαστε. Δηλαδή, με βάση το πιο πάνω παράδειγμα, τα φύλλα, δηλαδή οι μεταβλητές X4 και X5, θα παραμείνουν με βάρος 1. Άρα οι μεταβλητές αυτές στην αντικειμενική συνάρτηση θα έχουν βάρος 1. Οι μεταβλητές X2 και X3 βρίσκονται στο δεύτερο επίπεδο με αρχικό βάρος 2 και τώρα θα προστεθεί και το επίπεδο 1. Άρα οι μεταβλητές X2 και X3 τώρα θα έχουν βάρος 4. Τέλος, η μεταβλητή X1, που είναι και η ρίζα του γράφου, έχει αρχικό βάρος 3, και στο βάρος αυτό θα προστεθούν και τα βάρη των υπόλοιπων μεταβλητών. Δηλαδή, το βάρος 1 των μεταβλητών X4 και X5 και το βάρος 4 των μεταβλητών X2 και X3. Άρα η μεταβλητή X1 θα έχει τελικό βάρος 13.

Επίπεδο 1: 3+4+4+1+1 = 13 (τελικό βάρος)

Επίπεδο 2: 2+1+1=4 (τελικό βάρος)

Επίπεδο 1: 1 (παραμένει το ίδιο)



Τέλος, δημιουργείται το αρχείο «objfunction.txt», το οποίο περιέχει την αντικειμενική συνάρτηση με τα βάρη και την κάθε μεταβλητή, και στη συνέχεια έχουμε νέους περιορισμούς που δημιουργούνται από τα δύο πιο πάνω αρχεία (προτιμήσεις και περιορισμοί χρήστη).

Στο δεύτερο πρόγραμμα που έχουμε δημιουργήσει, το πρόγραμμα ζητά από τον χρήστη να δώσει το αρχείο των προτιμήσεων, το αρχείο των περιορισμών και να δώσει τον αριθμό από τον οποίο αποτελείται το CP-Net. Με το αρχείο των προτιμήσεων, δημιουργούμε τον γράφο που αποτελείται το CP-Net. Από τον γράφο, βρίσκουμε τα βάρη που θα έχει κάθε μεταβλητή, και δημιουργούμε την αντικειμενική συνάρτηση και τους περιορισμούς του αρχείου «objfunction.txt», το οποίο θα δοθεί στη συνέχεια στο εργαλείο Minisat+.

Μετά από αυτή τη διαδικασία, χρησιμοποιούμε το εργαλείο Minisat+. Το εργαλείο αυτό, διαβάζει το αρχείο «objfunction.txt», και βρίσκει τις εφικτές λύσεις για το συγκεκριμένο CP-Net. Στο τέλος, βρίσκει και την βέλτιστη λύση του προβλήματος αυτού, καθώς και τον χρόνο που χρειάζεται για να υπολογίσει κάθε λύση ξεχωριστά.

Στη συνέχεια, παίρνουμε την έξοδο του Minisat+, δηλαδή το αρχείο «output.txt», το διαβάζουμε και βρίσκουμε την βέλτιστη λύση του προβλήματος. Όταν βρούμε τις μεταβλητές που αποτελούν την λύση αυτή, δημιουργούμε ένα περιορισμό με τις μεταβλητές αυτές, ώστε να ξανατρέξουμε το εργαλείο Minisat+ και να βρούμε την δεύτερη βέλτιστη λύση του προβλήματος. Ο περιορισμός αυτός μπαίνει στο αρχείο «objfunction.txt».

Με την δημιουργία ενός bash αρχείου, αυτό γίνεται αυτόματα όσες φορές θέλει ο χρήστης. Επίσης, έχουμε ένα αρχείο «bestobj.txt», το οποίο περιέχει όλες τις βέλτιστες λύσεις του προβλήματος.

5.3 Μέρος 2

Ο σκοπός του κομματιού αυτού ήταν να χρησιμοποιηθεί το εργαλείο Minisat+. Η έξοδος του εργαλείου αυτού, θα μας δώσει όλες τις εφικτές λύσεις του προβλήματος, αλλά και την βέλτιστη. Στη συνέχεια, η βέλτιστη λύση θα γίνει περιορισμός, ώστε να βρεθεί η επόμενη βέλτιστη λύση. Μέσα από διάφορες τεχνικές που θα περιγραφούν παρακάτω, ο στόχος αυτός έχει επιτευχθεί.

Αρχικά, παίρνουμε το αρχείο «objfunction.txt», και το δίνουμε σαν είσοδο στο Minisat+. Η έξοδος του Minisat+, βρίσκεται στο αρχείο «output.txt». Διαβάζουμε το αρχείο αυτό, και βρίσκουμε την γραμμή στην οποία βρίσκεται η βέλτιστη λύση. Στη συνέχεια, κάνουμε την βέλτιστη λύση περιορισμό, και την γράφουμε στο αρχείο «objfunction.txt». Ξανατρέχουμε το Minisat+ για να πάρουμε την επόμενη βέλτιστη λύση. Αυτό γίνεται όσες φορές θέλει ο χρήστης. Για μεγαλύτερη ευκολία, έχει δημιουργηθεί ένα bash αρχείο, το οποίο ζητά από την χρήστη πόσες φορές θέλει να βρεθούν βέλτιστες λύσεις και γίνεται αυτόματα, χωρίς να πρέπει ο χρήστης να τρέχει συνέχεια τα αρχεία.

5.3.1 Τρόπος Λειτουργίας

Αρχικά, δίνουμε στο Minisat+, το αρχείο «objfunction.txt», στο οποίο βρίσκονται οι αντικειμενική συνάρτηση του προβλήματος, μαζί με τους περιορισμούς, ώστε να μας δώσει σωστή λύση. Αντί όμως να μας τυπώσει την έξοδο του εργαλείου αυτού στην οθόνη, ανακατευθύνουμε την έξοδο στο αρχείο «output.txt», ώστε να μπορούμε να την διαβάσουμε για να βρούμε την βέλτιστη λύση.

Για να βρούμε τις μεταβλητές που αποτελούν την βέλτιστη λύση, διαβάζουμε το αρχείο «output.txt», και βρίσκουμε την γραμμή που αρχίζει με «v».

Ένα παράδειγμα με την γραμμή αυτή είναι:

v -x151 x121 -x141 -x131 x181 -x161 -x171 -x191 -x1101 -x111 x112 x122 x132 x142 x152 x162 x172 -x182 x192 x1102

Στην γραμμή αυτή, υπάρχουν και θετικές και αρνητικές μεταβλητές. Εμείς όμως θέλουμε μόνο τις θετικές. Διαβάζουμε την γραμμή αυτή και βάζουμε σε πίνακα μόνο τις θετικές μεταβλητές. Με τις μεταβλητές αυτές δημιουργούμε ένα καινούριο περιορισμό στο αρχείο«objfunction.txt» , ο οποίος είναι διαφορετικός από τους αρχικούς.

Ένα παράδειγμα περιορισμού με τις βέλτιστες μεταβλητές είναι:

+1*x112 + 1*x122 + 1*x132 + 1*x142 + 1*x152 + 1*x162 + 1*x172 + 1*x182 + 1*x192 <= +8;

Τύποι Δεδομένων:

char *table[200];

Στον πίνακα αυτό, μπαίνουν οι θετικές μεταβλητές της βέλτιστης λύσης του προβλήματος, με τις οποίες θα δημιουργηθεί αργότερα ο νέος περιορισμός.

5.4 Μέρος 3

Στο μέρος αυτό, έχουμε την ένωση των πιο πάνω προγραμμάτων, δηλαδή του Μέρος 1 και του Μέρος 2, σε ένα bash αρχείο.

Τα αρχεία bash έχουν την δυνατότητα να τρέξουν ένα ολόκληρο αρχείο από εντολές, γνωστό και ως «Bash script». Τα αρχεία αυτά μπορεί να περιέχουν εντολές, βρόχους, και οτιδήποτε υπάρχει σε ένα αρχείο .c.

Η υλοποίηση αυτή είχε ως στόχο να δίνει ο χρήστης πόσες φορές θέλει να βρεθούν οι βέλτιστες λύσεις του συγκεκριμένου προβλήματος. Σε ένα καινούριο αρχείο, εμφανίζονται όσες βέλτιστες λύσεις επιλέξει ο χρήστης να εμφανιστούν, αν υπάρχουν.

5.4.1 Τρόπος Λειτουργίας

Το πρόγραμμα αυτό έχει δημιουργηθεί ώστε να μπορέσουμε να τρέξουμε 2 αρχεία .c και το Minisat+ ταυτόχρονα.

Αρχικά, τυπώνεται στην οθόνη το μήνυμα που ζητά από τον χρήστη να πληκτρολογήσει πόσες φορές θέλει να βρεθούν βέλτιστες λύσεις.

Στη συνέχεια, υπάρχει ένα for, το οποίο γίνεται όσες φορές θελήσει ο χρήστης, ώστε να τυπωθεί η βέλτιστη λύση στο αρχείο «bestobj.txt». Μέσα στο for αυτό, υπάρχουν οι εντολές με τα αρχεία και με ποια σειρά θα εκτελεστούν.

5.4.2 Εγχειρίδιο χρήσης του Προγράμματος

Το πρόγραμμα που έχει περιγραφεί πιο πάνω, δέχεται σαν είσοδο τον αριθμό των βέλτιστων λύσεων που επιθυμεί ο χρήστης να του εμφανιστούν στο αρχείο «bestobj.txt».

Κώδικας Προγράμματος:

```
#!/bin/bash
        echo Poses fores thelete na vrethei i veltisti lisi?
          read count

for i in 1 .. count-1
do
          ./afterminisat
          ./file1
           ./minisat+ objfunction.txt > output.txt
```

done

Τα αρχεία bash, ξεχωρίζουν από την αρχή και το τέλος τους. Δηλαδή, ένα αρχείο bash, πάντα αρχίζει με την εντολή #!/bin/bash.
Στη συνέχεια, τυπώνεται στην οθόνη το μήνυμα «Poses fores thelete na vrethei i veltisti lisi?» με την εντολή echo. Το echo είναι η αντίστοιχη εντολή του printf. Με την εντολή read διαβάζουμε τον αριθμό αυτό, και μπαίνει στην μεταβλητή count.
Με ένα for, από το 1 μέχρι τον αριθμό που δίνει ο χρήστης, εκτελούνται οι 3 εντολές με συγκεκριμένη σειρά, ώστε να έχουμε τα αποτελέσματα που θέλουμε.
Τέλος, πρέπει να γραφτεί η εντολή done, με την οποία δηλώνουμε το τέλος του προγράμματος.

5.4.3 Εντολές Εκτέλεσης του Προγράμματος

./hello.sh

Όπου «hello.sh» είναι το αρχείο που περιέχει όλες τις εντολές με τα προγράμματα που υλοποιήθηκαν, ώστε ο χρήστης να δίνει απλά τον αριθμό με τις βέλτιστες λύσεις που θέλει να του εμφανιστούν στο αρχείο, και να τρέχουν αυτόματα.

5.5 Πειραματική Αξιολόγηση

Μεταβλητές	*seconds*	*decisions*
10	0	44
20	0.0001998	318
50	0.0073984	1361
100	0.021996	3966
200	0.0311946	4296
500	0.2139638	15424

Πίνακας 1

Για την εξαγωγή των παραπάνω αποτελεσμάτων, τρέχουμε το πρόγραμμα που δημιουργήσαμε 5 φορές για κάθε αριθμό μεταβλητών, έτσι ώστε να βγάλουμε ένα μέσο όρο για κάθε διαφορετικό εύρος μεταβλητών. Από την έξοδο του εργαλείου Minisat+, παίρνουμε τον χρόνο που χρειάζεται για να υπολογίσει τις εφικτές και βέλτιστη λύση, και τις αποφάσεις που πρέπει να πάρει ανά δευτερόλεπτο για τις εφικτές και βέλτιστη λύση.

Μεταβλητές	*seconds*	*decisions*
1000	0.0263956	5918
1500	0.0097978	4632
2000	0.1189812	60138
2500	0.1459774	15144
5000	1.689132	74640

Πίνακας 2

Για την εξαγωγή των παραπάνω αποτελεσμάτων, τρέχουμε το πρόγραμμα που δημιουργήσαμε 5 φορές για κάθε αριθμό μεταβλητών, έτσι ώστε να βγάλουμε ένα μέσο όρο για κάθε διαφορετικό εύρος μεταβλητών. Από την έξοδο του εργαλείου Minisat+, παίρνουμε τον χρόνο που χρειάζεται για να υπολογίσει τις εφικτές και βέλτιστη λύση, και τις αποφάσεις που πρέπει να πάρει ανά δευτερόλεπτο για τις εφικτές και βέλτιστη λύση.

Ο γράφος που δημιουργείται στις συγκεκριμένες περιπτώσεις, δεν είναι γράφος με τόσες μεταβλητές. Στο πρόγραμμα που έχουμε δημιουργήσει, έχουμε βάλει ένα όριο 998 μεταβλητών για τον γράφο. Αν είναι πάνω από το όριο αυτό, τότε διαιρούμε με το 10, στρογγυλοποιούμε και παίρνουμε τον αριθμό αυτό για να κατασκευάσουμε τον γράφο. Για παράδειγμα, στην πρώτη γραμμή του Πίνακα 2, έχουμε ένα γράφο με 1000 μεταβλητές. Ο χρήστης δίνει 1000 μεταβλητές. Όμως το πρόγραμμα δημιουργεί ένα γράφο με 100

μεταβλητές, έτσι ώστε να μπορεί να δημιουργήσει τον γράφο και να μπορέσει να βγάλει βέλτιστη λύση του προβλήματος το εργαλείο Minisat+.

5.6 Συμπεράσματα Πειραματικής Αξιολόγησης

Από τον πίνακα 1, βλέπουμε από τους μέσους όρους, ότι όσο αυξάνονται οι μεταβλητές, τόσο αυξάνεται και ο χρόνος που χρειάζεται για να βρεθεί η βέλτιστη λύση αλλά και οι εφικτές. Επίσης, ανάλογα με τις μεταβλητές αυξάνονται και οι αποφάσεις που παίρνει το Minisat+ ανά δευτερόλεπτο για να βρεθεί η βέλτιστη λύση αλλά και οι εφικτές.

Από τον πίνακα 2 μπορούμε να εξάγουμε σχεδόν τα ίδια συμπεράσματα. Όπως βλέπουμε από τα αποτελέσματα των μέσων όρων, ο χρόνος για να υπολογιστεί η βέλτιστη λύση αλλά και οι εφικτές λύσεις, και οι αποφάσεις που παίρνει ανά δευτερόλεπτο για τις λύσεις αυτές, είναι ανάλογα του χρόνου. Αυτό δεν ισχύει για τις 1500 μεταβλητές. Ένας λόγος που μπορεί να συμβαίνει αυτό είναι λόγω της πολυπλοκότητας του γράφου. Μπορεί για τα περισσότερα παραδείγματα, ο γράφος να μην ήταν πολύπλοκός, άρα χρειάζεται λιγότερο χρόνο να υπολογίσει τις βέλτιστες λύσεις αλλά και τις εφικτές για κάθε παράδειγμα ξεχωριστά.

Κεφάλαιο 6

Συμπεράσματα

6.1 Συμπεράσματα Υλοποίησης

Κατά τη διάρκεια της πρώτης φάσης της παρούσας διπλωματικής εργασίας έγινε εξοικείωση με το εργαλείο Minisat+. Για τον λόγο αυτό έχουν παρθεί μερικές μετρικές σχετικά με τα αποτελέσματα του εργαλείου αυτού, έχοντας σαν είσοδο το αρχείο του τυχαίου CP-Net και του αρχείου , για να εξαχθούν κάποια συγκεκριμένα συμπεράσματα .

Σημασία έχει η πολυπλοκότητα του CP-Net, δηλαδή, πόσα παιδιά έχει κάθε πατέρας, και πόσους πατέρες έχει κάθε παιδί. Αν για παράδειγμα σε ένα CP-Net υπάρχουν 5 μεταβλητές, και οι 4 είναι όλες παιδιά της ρίζας, και σε ένα άλλο CP-Net, έχω 5 μεταβλητές αλλά το φύλλο έχει δύο πατέρες, τότε στο δεύτερο παράδειγμα θα χρειαστεί περισσότερο χρόνο για να υπολογίσει τις εφικτές λύσεις.

6.2 Μελλοντικές Επεκτάσεις

Θα μπορούσαν να υπάρχουν πολλές μελλοντικές επεκτάσεις καθώς και βελτιστοποιήσεις στην παρούσα διπλωματική εργασία. Οι βελτιστοποιήσεις οι οποίες θα μπορούσαν να γίνουν είναι η χρήση άλλων δομών για να επιτευχθούν καλύτερα αποτελέσματα τόσο σε χρόνο όσο και σε μη σπατάλη της αχρησιμοποίητης μνήμης.

Επίσης, θα μπορούσαν να υλοποιηθούν δύο αλγόριθμοι, έτσι ώστε να βρίσκει πάντα τις βέλτιστες λύσεις.

Επιπρόσθετα, η υλοποίηση του προγράμματος αυτού θα μπορούσε να γίνει σε άλλη γλώσσα προγραμματισμού και ποιο συγκεκριμένα σε μια αντικειμενοστραφή γλώσσα προγραμματισμού, όπως για παράδειγμα η JAVA. Αυτές οι γλώσσες παρέχουν περισσότερες δυνατότητες και διευκολύνσεις στον προγραμματιστή γιατί έχουν πολλές βιβλιοθήκες και πολλές συναρτήσεις υλοποιημένες. Με αυτό τον τρόπο ο προγραμματιστής δεν χρειάζεται να γράφει συναρτήσεις μιας και οι συναρτήσεις αυτές ήδη είναι υλοποιημένες.

Βιβλιογραφία

[2] Ronen I. Brafman and Carmel Domshlak, “Preference Handling – An Introductory Tutorial”

[3] Toby Walsh, “SAT v CSP”, 2000

[4] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, David Poole, “CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements”, 2004

Παράρτημα Α

Κύριες Λειτουργίες αρχείου «diplwmatiki.c»:

```
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>


int main(){

 int i;
 int j;
 int depth;
 int parents=3;
 int variables=0;
 int random;
 int random2;
 int randomKomvon=0; //posoi "1" tha mpun  ston pinaka
 int randomRiza;
 int countMetavlites=0;
int random3;


 printf("Please give number of variables: ");
 scanf("%d",&variables);

if(variables>999)
   variables=variables/10;
if(variables>1999)
   variables=variables/100;
if(variables>9999)
   variables=variables/1000;


 int varoi[variables];
 for(i=0;i<variables;i++)
    varoi[i]=0;


 int graph[variables][variables];

 //arxikopiume ton pinaka me 0
 for(i=0;i<variables;i++)
    for(j=0;j<variables;j++)
    graph[i][j]=0;

 char *metavlites[variables];
 int tableMetavlites[variables];
 int tableMetavlites2[variables];
 char *voithitikes[variables];
 char *voithitikes2[variables];

 //desmevi xoro gia tis metavlites ston pinaka
 for(i=0;i<variables;i++){
 metavlites[i]=(char*) malloc(sizeof(char));
 voithitikes[i]=(char*) malloc(sizeof(char));
 voithitikes2[i]=(char*) malloc(sizeof(char));
}

 //apothikevo tis thesis ton metavliton
 for(i=0;i<variables;i++){
 tableMetavlites[i]=i;
 tableMetavlites2[i]=i;
}

   printf("\n");

   //Dimiourgounte oi metavlites kai mpennun ston metavlites[]

   char temp[5];
   temp[0]='x';
   for(i=1;i<=variables;i++){
   if(i>9){
```

```
        sprintf(temp, "x%d", i);

        }
    else{
    temp[1]=(char)(((int)'0')+i);
    temp[2]='\0';
    }
    strcpy(metavlites[i-1],temp);
}

srand(time(NULL));

//Dimiourgounte tis nees voithitikes metavlites k mpainnoun sto voithitikes[]

    char temp1[6];
    temp[0]='x';
    temp[1]='1';
    for(i=1;i<=variables;i++){
    if(i>=1){

        sprintf(temp1, "x1%d", i);

        }
    strcpy(voithitikes[i-1],temp1);
    strcpy(voithitikes2[i-1],temp1);
}

srand(time(NULL));
printf("\n\n ");
for(i=0;i<variables;i++)
printf("%s ",metavlites[i]);
printf("\n ");
for(i=0;i<variables;i++)
printf("%s ",voithitikes[i]);
printf("\n ");
for(i=0;i<variables;i++)
printf("%s ",voithitikes2[i]);
printf("\n\n ");


//pios komvos tha ine i riza
randomRiza=rand() % variables;

//vazo -1 stin riza
tableMetavlites[randomRiza]=-1;

printf("Riza=%d\n",randomRiza);
printf("variables=%d\n",variables);

countMetavlites=variables-1;

    srand(time(NULL));

    //posoi komvoi tha mpun stn riza
    randomKomvon=rand() % 3 + 1;
      int tempRandom=0;

    int table[randomKomvon];
    for(i=0;i<randomKomvon;i++)
    table[i]=-1;

    //mpainoun oi komvoi stin riza
    for(i=0;i<randomKomvon;i++){
    random= rand() % (variables);

    while(table[j]==random)
    random= rand() % (variables);

    while(graph[randomRiza][random]==1 || randomRiza==random){ //elegxos gia na min dixni i riza ston eafto tis + na min mpenni to 1 ,
dio fores stn idia thesi
    random= rand() % (variables);

}
      tempRandom=random;
    graph[randomRiza][random]=1;  //vazi ena ston pinaka ts rizas
    table[i]=random;              //filao tis thesis ton pedion ts rizas
    tableMetavlites[random]=-1;
    countMetavlites--;

}
```

```
printf("\n%d\n",tempRandom);
 char *  tempv=voithitikes[tempRandom];
 //strcpy(tempv, voithitikes[tempRandom]);
  printf("\ntempv %s\n",tempv);
for(i=0;i<randomKomvon;i++)
printf("\ntable=%d",table[i]);

 srand(time(NULL));



while(countMetavlites!=0){

//poses metavlites tha mpun sto epipedo afto
random = rand() % (countMetavlites)+1;

for(i=0;i<random;i++){

//pia metavliti apo tis 3apolites tha mpun mesa
random2 = rand() % (variables);
random3=rand()%randomKomvon; //pia apo tis metavlites tou table tha exi pedi

while(tableMetavlites[random2]==-1 || graph[table[random3]][random2]==1 || randomRiza==random2)
random2 = rand() % (variables);


graph[table[random3]][random2]=1;


}
countMetavlites=countMetavlites-random;
}

//2 pateres

int tableIpopsifioi[variables];

//arxikopioiisi pinaka

for(i=0;i<variables;i++)
tableIpopsifioi[i]=-1;

int count=0;

for(i=0;i<variables;i++){
++count;
 if(graph[randomRiza][i]!=0)
 tableIpopsifioi[variables]=i;

                    }
 for(i=0;i<count;i++){
 for(j=0;j<variables;j++){
 if(graph[tableIpopsifioi[i]][j]==1){

                            if(graph[tableIpopsifioi[i+1]][j]==1)
                             graph[tableIpopsifioi[i+1]][j]=1;
                }


                        }
}

//ARXEIO PREFERENCES
FILE *new=fopen("preferences.txt","w");
fputs(":",new);
fputs(metavlites[randomRiza], new);
fputs("1",new);
fputs(">",new);
fputs(metavlites[randomRiza], new);
fputs("2\n",new);
tableMetavlites[randomRiza]=-1;

for(i=0;i<variables;i++){
                    if(graph[randomRiza][i]==1){
                    tableMetavlites[i]=-1;
                    fputs(metavlites[randomRiza], new);
                    fputs("1",new);
                    fputs(":",new);
                    fputs(metavlites[i], new);
```

```
                fputs("1",new);
                fputs(">",new);
                fputs(metavlites[i], new);
                fputs("2\n",new);

                fputs(metavlites[randomRiza], new);
                fputs("2",new);
                fputs(":",new);
                fputs(metavlites[i], new);
                fputs("2",new);
                fputs(">",new);
                fputs(metavlites[i], new);
                fputs("1\n",new);
                }
                }



i=0;
j=0;
int tablePateres[2];


//vari
int epipeda=1;

//vriskoume ta epipeda
for(i=0;i<variables;i++)
for(j=0;j<variables;j++){
                if(graph[i][j]==1){
                epipeda++;
                j=variables;
                        }

                }


//vazoume to varos tis rizas
varoi[randomRiza]=1;
printf("\nepipeda=%d",epipeda);

//vazoume ta varoi ton pedion ts rizas
for(i=0;i<randomKomvon;i++)
varoi[table[i]]=1;

int k=0;
int epipedo3=0; //posa paidia vriskontai sto epipedo 3
int epipedo2=0; // varos g epipedo 2

int table5[variables];
for(i=0;i<variables;i++)
table5[i]=-1;
int d=0;

//vazoume ta vari sto grafo an exoume 3 epipeda
if(epipeda==3){

for(i=0;i<randomKomvon;i++){
epipeda--;
for(j=0;j<variables;j++){
if(graph[table[i]][j]==1){
varoi[j]=1;
epipedo3++;
table5[d]=j;
d++;
printf("\nd====%d",d);
}
}
}

printf("\nepipedo3==== %d \n",epipedo3);

for(k=0;k<randomKomvon;k++){
varoi[table[k]]= 1 + epipedo3;
printf("\nvaroi paidiwn====%d",varoi[table[k]]);
epipedo2 = epipedo2+ varoi[table[k]];
}
```

```
printf("\nepipedo2====%d",epipedo2);

for(k=0;k<randomKomvon;k++){
varoi[randomRiza]=1+epipedo2+epipedo3;
printf("\nvaros rizas====%d",varoi[randomRiza]);

}

}// telos if epipeda 3


int table2[variables];
for(i=0;i<variables;i++)
table2[i]=-1;

int a=0;
int epipedo4=0; //varos epipedou 4

//vazoume ta vari ston grafo an exoume 4 epipeda
if(epipeda==4){
for(i=0;i<randomKomvon;i++){
epipeda--;
for(j=0;j<variables;j++){
  if(graph[table[i]][j]==1){
    varoi[j]=1;
    epipedo4++;
    table2[a]=j;
    a++;
  }
 }
}




printf("\nepipedo4====%d",epipedo4);

for(i=0;i<a;i++)
printf("\ntable2=%d",table2[i]);

for(k=0;k<a;k++)
for(i=0;i<variables;i++){
if((graph[table2[k]][i]==1) && varoi[table2[i]]==0){
varoi[table2[k]]=1;
}
}

for(k=0;k<randomKomvon;k++){
varoi[table[k]]= 1 + epipedo4;
printf("\nvaroi paidiwn====%d",varoi[table[k]]);
epipedo2 = epipedo2+ varoi[table[k]];
}

printf("\nepipedo2====%d",epipedo2);

for(k=0;k<randomKomvon;k++){
varoi[randomRiza]=1+epipedo2+epipedo4;
printf("\nvaros rizas====%d",varoi[randomRiza]);

}

} // telos if epipeda 4


int table3[variables];
for(i=0;i<variables;i++)
table3[i]=-1;

int b=0;
int epipeda5=0;

if(epipeda==5){

for(i=0;i<randomKomvon;i++){
epipeda--;
for(j=0;j<variables;j++){
if(graph[table[i]][j]==1){
varoi[j]=1;
```

```
epipedo4++;
table2[a]=j;
a++;
}
}
}
printf("\nepipedo4====%d",epipedo4);

for(i=0;i<a;i++)
printf("\ntable2=%d",table2[i]);

for(k=0;k<a;k++)
for(i=0;i<variables;i++){
if((graph[table2[k]][i]==1) && varoi[table2[i]]==0){
varoi[table2[k]]=1;
table3[b]=i;
b++;
}
}

for(k=0;k<b;k++)
for(i=0;i<variables;i++){
if((graph[table3[k]][i]==1) && varoi[table3[i]]==0){
varoi[table3[k]]++;
}
}

for(k=0;k<randomKomvon;k++)
for(i=0;i<variables;i++){
if(graph[table[k]][i]==1){
varoi[table[k]]++;
}
}

for(k=0;k<randomKomvon;k++){
varoi[randomRiza]=varoi[randomRiza]+varoi[table[k]];
}

}// telos if epipeda 5



for(i=0;i<variables;i++)
printf("\nvaroi=%d",varoi[i]);



int table4[2];

for(i=0;i<variables;i++)
printf("\nmetavlites=%s",metavlites[i]);

//tipoma sto arxio PREFERENCES

int c=0;

for(i=0;i<variables;i++){
                for(j=0;j<variables;j++){
                if(graph[j][i]==1){
                b=j;
                ++c;
                if(c==1)
                table4[0]=j;
                else
                table4[1]=j;

                }

                }

                if(c==1 && (i!=randomRiza && b!=randomRiza)){

                fputs(metavlites[b], new);
                fputs("1",new);
                fputs(":",new);
                fputs(metavlites[i], new);
                fputs("1",new);
                fputs(">",new);
                fputs(metavlites[i], new);
                fputs("2\n",new);
```

```
                 fputs(metavlites[b], new);
                fputs("2",new);
                fputs(":",new);
                fputs(metavlites[i], new);
                fputs("2",new);
                fputs(">",new);
                fputs(metavlites[i], new);
                fputs("1\n",new);
                        }

                else if(c==2 && (i!=randomRiza && b!=randomRiza)){


                    fputs(metavlites[table4[0]],new);
                    fputs("1",new);
                    fputs("^",new);
                    fputs(metavlites[table4[1]],new);
                    fputs("1",new);
                    fputs(":",new);
                    fputs(metavlites[i], new);
                    fputs("1",new);
                    fputs(">",new);
                    fputs(metavlites[i], new);
                    fputs("2\n",new);

                    fputs(metavlites[table4[0]],new);
                    fputs("1",new);
                    fputs("^",new);
                    fputs(metavlites[table4[1]],new);
                    fputs("2",new);
                    fputs(":",new);
                    fputs(metavlites[i], new);
                    fputs("2",new);
                    fputs(">",new);
                    fputs(metavlites[i], new);
                    fputs("1\n",new);

                    fputs(metavlites[table4[0]],new);
                    fputs("2",new);
                    fputs("^",new);
                    fputs(metavlites[table4[1]],new);
                    fputs("1",new);
                    fputs(":",new);
                    fputs(metavlites[i], new);
                    fputs("2",new);
                    fputs(">",new);
                    fputs(metavlites[i], new);
                    fputs("1\n",new);

                    fputs(metavlites[table4[0]],new);
                    fputs("2",new);
                    fputs("^",new);
                    fputs(metavlites[table4[1]],new);
                    fputs("2",new);
                    fputs(":",new);
                    fputs(metavlites[i], new);
                    fputs("1",new);
                    fputs(">",new);
                    fputs(metavlites[i], new);
                    fputs("2\n",new);

                    }
                       c=0;
                       b=0;
                       }
fclose(new);


//ARXEIO CONSTRAINTS

FILE *cons=fopen("constraints.txt","w");

//eksiswsi
```

```
fputs("min: +",cons);
fprintf(cons, "%d",varoi[randomRiza]);
fputs("*",cons);
fputs(voithitikes[randomRiza],cons);
fputs("1",cons);

for(i=0;i<randomKomvon;i++){
fputs(" + ",cons);
fprintf(cons, "%d",varoi[table[i]]);
fputs("*",cons);
fputs(voithitikes[table[i]],cons);
fputs("1",cons);
}

if(table5[0]!=-1){
printf("\nd=%d",d);
for(i=0;i<d;i++){
fputs(" + ",cons);
fprintf(cons, "%d",varoi[table5[i]]);
fputs("*",cons);
fputs(voithitikes[table5[i]],cons);
fputs("1",cons);

		}
}


if(table2[0]!=-1){
//printf("\na=%d",a);
for(i=0;i<a;i++){
fputs(" + ",cons);
fprintf(cons, "%d",varoi[table2[i]]);
fputs("*",cons);
fputs(voithitikes[table2[i]],cons);
fputs("1",cons);

		}
}

if(table3[0]!=-1){

for(i=0;i<b;i++){
fputs(" + ",cons);
fprintf(cons, "%d",varoi[table3[i]]);
fputs("*",cons);
fputs(voithitikes[table3[i]],cons);
fputs("1",cons);

		}
}

fputs(" ;\n",cons);

//constraints

for(i=0;i<variables;i++){
fputs("\n+1*",cons);
fputs(voithitikes[i],cons);
fputs("1",cons);
fputs(" + 1*",cons);
fputs(voithitikes[i],cons);
fputs("2",cons);
fputs(" >= +1;",cons);
}

//violations

fprintf(cons, "\n+1");
fputs("*",cons);
fputs(voithitikes[randomRiza],cons);
fputs("2",cons);
fputs(" >= +1;",cons);

for(i=0;i<randomKomvon;i++){
fputs("\n+1",cons);
fputs("*",cons);
fputs(voithitikes[randomRiza],cons);
fputs("1 + ",cons);
fputs("1*",cons);
fputs(voithitikes[table[i]],cons);
```

```
fputs("2",cons);
fputs(" >= +1;",cons);

fputs("\n+1",cons);
fputs("*",cons);
fputs(voithitikes[randomRiza],cons);
fputs("2 + ",cons);
fputs("1*",cons);
fputs(voithitikes[table[i]],cons);
fputs("1",cons);
fputs(" >= +1;",cons);
}


for(i=0;i<randomKomvon;i++){

    for(j=0;j<variables;j++)
    if(graph[table[i]][j]==1){
    fputs("\n+",cons);
    fputs("1*",cons);
    fputs(voithitikes[table[i]],cons);
    fputs("1",cons);
    fputs(" + ",cons);
    fputs("1*",cons);
    fputs(voithitikes[j],cons);
    fputs("2",cons);
    fputs(" >= +1;",cons);

    fputs("\n+1",cons);
    fputs("*",cons);
    fputs(voithitikes[table[i]],cons);
    fputs("2",cons);
    fputs(" + ",cons);
    fputs("1*",cons);
    fputs(voithitikes[j],cons);
    fputs("1",cons);
    fputs(" >= +1;",cons);

}

}

//}
//if 4 epipeda
if(table2[0]!=-1){
printf("epipeda4");
for(i=0;i<a;i++){

    for(j=0;j<variables;j++)
    if(graph[table2[i]][j]==1){
    fputs("\n",cons);
    fputs("1*",cons);
    fputs(voithitikes[table2[i]],cons);
    fputs("1",cons);
    fputs(" + ",cons);
    fputs("1*",cons);
    fputs(voithitikes[j],cons);
    fputs("2",cons);
    fputs(" >= +1;",cons);

    fputs("\n",cons);
    fputs("1*",cons);
    fputs(voithitikes[table2[i]],cons);
    fputs("2",cons);
    fputs(" + ",cons);
    fputs("1*",cons);
    fputs(voithitikes[j],cons);
    fputs("1",cons);
    fputs(" >= +1;",cons);

}

}

}


fclose(cons);
```

```
// vriskw poies metavlites leipoun apo ton grafo
int metav[2]={0,0};
int m=0,qw=0,as=0;

int ar=0;

for(i=0;i<variables;i++){
m=0;
for(j=0;j<variables;j++) {
if(graph[j][i]==0)
++m;
}
if(m==variables){
metav[ar]=i;
++ar;
}
}



if(metav[0]==randomRiza)
metav[0]=-1;
else if(metav[1]==randomRiza)
metav[1]=-1;

char *temp_m = (char *) malloc(10);
int qaz=0;

for(i=0;i<ar;i++){
if(metav[i]!=-1){
strcpy(temp_m,voithitikes[metav[i]]);
qaz=1;
}
}


 FILE *obj=fopen("objfunction.txt","w");
FILE *con=fopen("constraints.txt","r");
char line[1000];
fgets(line,1000,con);
  printf("\n%s\n",line);

for(i=0;i<variables;i++)
printf("%s ",voithitikes[i]);


  char * pch=  (char *) malloc(30);
  char *pchtable[(variables+1)];
 char *array[(variables+1)];
  int zx=0,cv=0;
int op=0;

  for(i=0;i<100;i++){
    pchtable[i]=(char *) malloc(30);
    array[i]=(char *) malloc(30);
 pchtable[i]=NULL;
    array[i]=NULL;

  }

  pch = strtok (line," +;");


                // mpainnoun ston pinaka pchtable oi metavlites tis antikimenikis sinartisis
                  while (pch != NULL)
                  {
                                pchtable[zx]= pch;
                                zx++;
                                pch = strtok (NULL, " +;");

                  }

                  --zx;
                  printf ("\n\n");
                for(i=0;i<zx;i++){
                                printf(" pchtable= %s  \n",pchtable[i]);
                                //printf("array = %s \n",array[i]);
                }
                int same=0;
                int result=-10;
```

```
		for(i=0;i<(zx-2);i++){
			for(j=i+1;j<(zx-1);j++){
				printf(" i=%d, j=%d\n ",i,j);
				result=strcmp(pchtable[i],pchtable[j]);
				//printf("\n result=%d \n",result);

				if(result==0){
					//printf("\n yaaaaaaaaaay \n");
					pchtable[i] = '\0';
					//same=1;
					break;
				}
			}
			//if(same==1)
			//	break;
		}
		for(i=0;i<zx;i++){
			printf(" pchtable= %s  \n",pchtable[i]);
			//printf("array = %s \n",array[i]);
		}
  printf("zx=%d \n",zx);
  printf("variables=%d\n ",variables);



fputs(pchtable[0],obj);
//fputs(" +",obj);
for(j=1;j<zx;j++){

if(pchtable[j]=='\0')
		continue;
fputs(" + ",obj);
fputs(pchtable[j],obj);
		}

if(qaz==1){
  fputs(" + 1*",obj);
  fputs(temp_m,obj);

}
  fputs("1;",obj);
fputs("\n",obj);

 while(fgets(line, 80, con) != NULL)
   {
		fputs(line,obj);
   }
  if(qaz==1){
  fputs("\n",obj);
  fputs("+1*",obj);
  fputs(tempv,obj);
  fputs("1 + 1*",obj);
   fputs(temp_m,obj);
   fputs("2 >= +1;",obj);

   fputs("\n",obj);
  fputs("+1*",obj);
  fputs(tempv,obj);
  fputs("2 + 1*",obj);
   fputs(temp_m,obj);
   fputs("1 >= +1;",obj);
}

 fputs("\n",obj);


fclose(con);
fclose(obj);



for(i=0;i<variables;i++){
    printf("\n");
    for(j=0;j<variables;j++)
    printf("%d",graph[i][j]);
}

 printf("\n\n");
    return 0;
```

```
}
```

Παράρτημα Β

Κύριες Λειτουργίες αρχείου «afterminisat.c»:

```
int main(){

FILE *out=fopen("output.txt","r");
char myString [1000];
char line[1000];
    int upper = 0;
    int lower = 0;
    char c;
    char a,b;

    while(fgets( myString, 1000, out) != NULL){
                int jj=-1;
      while(++jj < strlen(myString)) {
        if ((c = myString[jj]) != ' ') break;
      }
      if (c=='v') {
                a='v';
                strcpy(line,myString);
        //printf("%s \n",line); // print *U* to show character recognized as uppercase
      }

if (c=='s') {
 b='s';
                //strcpy(line,myString);
        //printf("UNSATISFIABLE \n"); // print *U* to show character recognized as uppercase
      }


    }

 if (b=='s' && a!='v') {
                strcpy(line,myString);
        printf("UNSATISFIABLE \n"); // print *U* to show character recognized as uppercase
      }


   const char s[2] = " ";
   char *token;
int count=0;
 char *table[200];

   /* get the first token */
   token = strtok(line, " ");

   /* walk through other tokens */
   while( token != NULL )
   {
     //printf( " %s\n", token );
table[count] = malloc(strlen(token) + 1);

       c = token[0];

      if (c!='-') {
                strcpy(table[count], token);
count++;
      }

//strcpy(table[count], token);
      token = strtok(NULL, s);

   }

//count--;
 int param;
int i;
/*for(i=0;i<count;i++)
```

```
                printf("table = %s ", table[i]);
    printf("\n");
*/
fclose (out);


FILE *obj = fopen("objfunction.txt","a");
FILE *best = fopen("bestobj.txt","a");

 if (obj == NULL) {
        printf("I couldn't open file for appending.\n");
        exit(0);
      }

param =count-1;
fputs("+1*",obj);
fputs("+1*",best);

fputs(table[1],obj);
fputs(table[1],best);

for(i=2;i<count;i++){
fputs(" + 1*",obj);
fputs(" + 1*",best);

fputs(table[i],obj);
fputs(table[i],best);
//printf("\ni = %d     ",i);
//printf("\ncount = %d ",count);
if(i==count-1){
fputs("\n",best);
                fputs(" <= +",obj);
//fputs(param,obj);
fprintf(obj, "%d;", param-1);
}



}

      /* close the file */
      fclose(obj);
                fclose(best);



return 0;
}
```

Παράρτημα Γ

```
int main(){

 int i;
 int j;
int q;
 int variables=0;
 FILE *preferences;
char fileName[15];


printf("Please give the file with the cp-network:  ");
gets(fileName);

preferences = fopen(fileName, "r");

printf("Please give number of variables: ");
 scanf("%d",&variables);

 int varoi[variables];
 for(i=0;i<variables;i++)
    varoi[i]=0;


 int graph[variables+1][variables+1];

 //arxikopiume ton pinaka me 0
 for(i=0;i<variables+1;i++)
    for(j=0;j<variables+1;j++)
    graph[i][j]=0;


int count=0;
int count1=0;
int c;
if ( preferences != NULL )
  {
     char line [ 20 ];
 fgets ( line, sizeof line, preferences );
count=1;
     while ( fgets ( line, sizeof line, preferences ) != NULL )
     {
                        printf("%d\n",strlen(line));
            count++;
            if(strlen(line)==12){
                                if(count%2==0){
                                        j= line[1]-'0';
                                        i= line[5]-'0';

                                        graph[j][i]=1;
                                }

            }

            else if(strlen(line)>=16){
                                if(count%2==0){
                                        j= line[1]-'0';
                                        i= line[9]-'0';
                                        q= line[5]-'0';

                                        graph[j][i]=1;
                                        graph[q][i]=1;

                                }

            }

    }
}
else
{
```

```
perror("Error while opening file. \n");
exit(-1);
}
 int table[variables+1];
for(i=0;i<variables+1;i++)
		table[i]=0;



int epipeda=1;

//vriskoume ta epipeda
for(i=1;i<variables+1;i++){
  count=0;
		for(j=1;j<variables+1;j++){
		    if(graph[i][j]==1){
					count++;
		     }

	}
		if(count<2 && count>0){
				epipeda++;
		}
}
count=0;
int riza=0;
for(i=1;i<variables+1;i++){
for(j=1;j<variables+1;j++){
		if(graph[j][i]==0)
			count++;

}
if(count==variables){
table[i]=epipeda;

riza=i;
break;
}
}

int next[20];
for(i=0;i<20;i++)
		next[i]=-1;

count=0;
count1=0;
int count2=0;
do{
		count1=0;
		i=1;
		while(i<variables+1){
			if(graph[riza][i]==1){
				table[i]=table[riza]-1;
				next[count2]=i;
				count2++;
			}
			else{
				count1++;
			}
		i++;
		}

		printf("count1=%d \n",count1);
		if(count1==variables)
			table[riza]=1;

		if(next[count]!=-1){
			riza=next[count];
			count++;
		}

}while(next[count]!=-1);


for(i=1;i<variables+1;i++)
printf("%d\n",table[i]);

printf("\n");
```

```
int a;
count=0;

 int table1[variables+1];
for(i=0;i<variables+1;i++)
        table1[i]=0;

int b;
for(i=1;i<epipeda+1;i++){
        count=0;
        for(j=1;j<variables+1;j++){
                if(table[j]==i && i==1){
                        table1[j]=1;
                        count++;
                }
                else if(table[j]==i && i!=1 && table1[j]!=1){
                        table[j]=table[j]+a;
                        b=table[j];
                        count++;
                        table1[j]=1;
                }
        }

        if(i==1)
                a=i*count;
        else
                a=b*count;



}


for(i=1;i<variables+1;i++){
    for(j=1;j<variables+1;j++)
    printf("%d",graph[i][j]);
printf("\n");
}
printf("\n");


for(i=1;i<variables+1;i++)
printf("%d\n",table[i]);

printf("\n");


return 0;
}
```