Ατομική Διπλωματική Εργασία

Τίτλος Εργασίας:

ΥΠΟΛΟΓΙΣΜΟΣ ΣΕ ΓΡΑΦΟΥΣ ΣΥΝΟΧΗΣ ΜΕ ΤΕΧΝΙΚΕΣ ΙΚΑΝΟΠΟΙΗΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ

Μιχαέλα Χρυσοστόμου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΠΟΛΟΓΙΣΜΟΣ ΣΕ ΓΡΑΦΟΥΣ ΣΥΝΟΧΗΣ ΜΕ ΤΕΧΝΙΚΕΣ ΙΚΑΝΟΠΟΙΗΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ

Μιχαέλα Χρυσοστόμου

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 μου Δρ. Γιάννη Δημόπουλο, για την συνεχή βοήθεια και την στήριξη που μου έχει προσφέρει καθ' όλη την διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.
Επίσης, θα ήθελα να ευχαριστήσω ιδιαιτέρως την οικογένεια μου και τους φίλους μου που με ενθάρρυναν καθημερινά να προχωρήσω σε κάθε δυσκολία που αντιμετώπιζα.

Περίληψη

Σε αυτή τη διπλωματική εργασία μελετάται ο υπολογισμός σε Γράφους Συνοχής με τεχνικές ικανοποίησης περιορισμών. Γίνεται ανάλυση της θεωρίας του Προβλήματος της Συνοχής για να επιτευχθεί η μοντελοποίηση του προβλήματος σε διάφορους επιλυτές για τον υπολογισμό αυτό. Σε πάρα πολλούς τομείς όπως στην ψυχολογία, στην μεταφυσική, στην επιστημολογία, καθώς και στον τομέα της λογικής και της ηθικής υπάρχουν πάρα πολλοί υποστηρικτές της θεωρίας αυτής.

Έχοντας ως δεδομένο ένα γράφο συνοχής όπου κάθε κόμβος αντιπροσωπεύει στοιχεία όπως προτάσεις, ιδέες, χαρακτηριστικά, έννοιες, μέρος μιας εικόνας, στόχους κτλ και το βάρος των ακμών μεταξύ των στοιχείων δείχνουν αν έχουν συνοχή δηλαδή ταιριάζουν μεταξύ τους ή δεν έχουν συνοχή δηλαδή δεν ταιριάζουν μεταξύ τους με διάφορους τρόπους, το Πρόβλημα της Συνοχής ορίζεται ως η διαδικασία για το πώς μπορούμε να δεχτούμε κάποια από τα στοιχεία και να απορρίψουμε άλλα με έναν τρόπο που να μεγιστοποιείται η συνοχή. Η μεγιστοποίηση της Συνοχής επιτυγχάνεται με τη διαίρεση των στοιχείων σε αποδεχτά και μη αποδεχτά σύνολα ώστε να ικανοποιούνται οι περιορισμοί του προβλήματος. Οι περιορισμοί του Προβλήματος της Συνοχής αφορούν την σχέση μεταξύ δύο στοιχείων. Αν δύο στοιχεία ταιριάζουν, υπάρχει ένας θετικός περιορισμός μεταξύ τους. Αν δύο στοιχεία δεν ταιριάζουν υπάρχει ένας αρνητικός περιορισμός μεταξύ τους. Ένας θετικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί με την αποδοχή και των δύο στοιχείων ή απορρίπτοντας και τα δυο στοιχεία. Ένας αρνητικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί μόνο με την αποδοχή του ενός στοιχείου και την απόρριψη του άλλου στοιχείου. Σκοπός της παρούσας διπλωματικής είναι η μοντελοποίηση του προβλήματος μέσω διαφόρων τεχνικών ικανοποίησης περιορισμών, πιο συγκεκριμένα μοντελοποιημένο ως πρόβλημα ικανοποίησης περιορισμών , ώστε να δοθεί ως είσοδο για επίλυση σε επιλυτές προβλημάτων ικανοποίησης περιορισμών με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών, αλλά και με επίλυση του προβλήματος μέσω ψευδοδυαδικών περιορισμών και με χρήση ψευδοδυαδικής βελτιστοποίησης, ώστε να δοθεί ως είσοδο για επίλυση σε επιλυτές ψευδοδυαδικών περιορισμών. Έγινε η μοντελοποίηση του προβλήματος αυτού σε διάφορους επιλυτές όπως ο MiniZinc, Gecode, Gecode με χρήση συνδυαστών αναζήτησης, Minisat+, bsolo_pb10 και

npSolver έτσι ώστε να διαπιστωθεί ποιός είναι ο πιο αποδοτικός στο να βρίσκει την βέλτιστη λύση για το Πρόβλημα της Συνοχής μέσω διαφόρων πειραμάτων που έγιναν.

Περιεχόμενα

Κεφάλαιο **1** **Εισαγωγή…….....1**

1.1 Σκοπός της Διπλωματικής εργασίας……..…..1

1.2 Δομή της Διπλωματικής εργασίας…….…….2

Κεφάλαιο **2** **Πρόβλημα της Συνοχής…….……..5**

2.1 Εισαγωγή…….....5

2.2 Συνοχή ως Ικανοποίηση Περιορισμών…….…….....6

2.3 Τυπικός ορισμός του Προβλήματος Συνοχής…….……7

2.4 Παραδείγματα Προβλημάτων Συνοχής……...8

2.5 Αλγόριθμοι Υπολογισμού της Συνοχής…….....11

Κεφάλαιο **3** **Προβλήματα Ικανοποίησης Περιορισμών και Προτασιακή Ικανοποιησιμότητα και Ψευδοδυαδικοί Περιορισμοί……13**

3.1 Ορισμός Προβλήματος Ικανοποίησης Περιορισμών…….....13

3.2 Κλασικά Παραδείγματα ΠΙΠ……....15

3.3 Λόγοι αναπαράστασης και επίλυσης προβλημάτων ως Πρόβλημα Ικανοποίησης Περιορισμών…..16

3.4 Εφαρμογές ΠΙΠ…….....17

3.5 Προτασιακή Ικανοποιησιμότητα……..17

3.5.1 Αλγόριθμος DPLL…….…....18

3.5.2 Λόγοι επέκτασης προτασιακών επιλυτών με χρήση ψευδοδυαδικών περιορισμών……..……19

3.6 Ορισμός Ψευδοδυαδικών περιορισμών…….…….....20

3.7 Γραμμικοί Ψευδοδυαδικοί περιορισμοί……..……..……20

3.8 Μη γραμμικοί Ψευδοδυαδικοί περιορισμοί…….....21

3.9 Κανονικοποίηση ψευδοδυαδικών περιορισμών…….....22

3.10 Αλγόριθμος ψευδοδυαδικών επιλυτών…….……..…23

3.11 Ψευδοδυαδικοί επιλυτές σε προβλήματα αποφασισιμότητας και σε προβλήματα βελτιστοποίησης…….....23

Κεφάλαιο **4** **Μοντελοποίηση Προβλήματος Συνοχής με βάση τους επιλυτές…….25**

4.1 Επιλυτής MiniZinc……..……..26

4.2 Επιλυτής Gecode……...28

4.2.1 Επιλυτής Gecode με χρήση search combinator……..29

4.3 Επιλυτής Minisat +……...…….32

4.4 Επιλυτής Bsolo_pb10……..……..…33

4.4.1 Αλγόριθμος Κλάδου και Φράγματος για Ακέραιο Προγραμματισμό…….34

4.4.2 Αλγόριθμος Bsolo…….....35

4.4.3 Μέθοδοι βελτιστοποίησης μέσω παραμέτρων στον Bsolo……36

4.4.3.1 Ανεξαρτησία των περιορισμών και Χαλάρωση Γραμμικού Προγραμματισμού…….37

4.4.3.2 Χρήση Στρατηγικής Εκμάθησης……38

4.5 Επιλυτής npSolver ……38

4.6 Μοντελοποίηση Προβλήματος Συνοχής στους Επιλυτές…….....40

4.6.1 Μοντελοποίηση στον επιλυτή MiniZinc……..…..…41

4.6.2 Μοντελοποίηση στους επιλυτές Minisat+ και Bsolo…….49

4.6.3 Μοντελοποίηση στον επιλυτή npSolver…….…....55

4.6.4 Μοντελοποίηση στον επιλυτή Gecode…….....56

4.6.4.1 Μοντελοποίηση με χρήση συνδυαστών αναζήτησης……..56

Κεφάλαιο **5** **Πειραματική Ανάλυση και Συμπεράσματα …….58**

5.1 Πειραματική Ανάλυση.....…….....58

5.1.1 Πειράματα διαφορετικών μοντέλων σε MiniZinc…….....58

5.1.2 Πειράματα διαφορετικών παραμέτρων στον Bsolo……...62

5.1.3 Πειράματα μεταξύ όλων των επιλυτών ανάλογα με πυκνότητα…….……65

5.1.4 Πειράματα μεταξύ όλων των επιλυτών με διαφορετικό τρόπο απεικόνισης της συσχέτισης δύο στοιχείων……...73

Κεφάλαιο **6** **Γενικά Συμπεράσματα και Μελλοντική Εργασία……..……..78**

6.1 Σύνοψη-Συμπεράσματα Πειραμάτων…….……...78

6.2 Μελλοντική Εργασία…….79

Β ι β λ ι ο γ ρ α φ ί α ……..**81**

Π α ρ ά ρ τ η μ α Α…….……..…….……..**Α-1**

Π α ρ ά ρ τ η μ α Β……..**Β-1**

Π α ρ ά ρ τ η μ α Γ……..**Γ-1**

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός της Διπλωματικής εργασίας……1
1.2 Δομή της Διπλωματικής εργασίας……...2

1.1 Σκοπός της Διπλωματικής εργασίας

Βασικός σκοπός της διπλωματικής αυτής είναι η μοντελοποίηση του Προβλήματος της Συνοχής σε γλώσσες Ικανοποίησης Περιορισμών.

Στην καθημερινή μας ζωή για την αντίληψη ενός κειμένου, μιας εικόνας, ενός ατόμου, ή ενός γεγονότος δημιουργούμε μια ερμηνεία που να ταιριάζει με τις διαθέσιμες πληροφορίες, λαμβάνοντας υπόψη τόσο κομμάτια των πληροφοριών που ταιριάζουν όσο και αυτά που δεν ταιριάζουν μεταξύ τους. Για παράδειγμα όταν συναντούμε άγνωστους ανθρώπους μπορούμε μέσω ενός δικού μας συνδυασμού σεναρίων και υποθέσεων για τον άνθρωπο αυτό να φτιάξουμε μια γενική εικόνα για το χαρακτήρα του.

Το Πρόβλημα της Συνοχής λαμβάνοντας υπόψη ένα τέτοιο μεγάλο αριθμό στοιχείων όπου τα στοιχεία αυτά μπορεί να είναι προτάσεις, ιδέες, χαρακτηριστικά, έννοιες, μέρος μιας εικόνας, στόχους κτλ που έχουν συνοχή δηλαδή ταιριάζουν μεταξύ τους ή δεν έχουν συνοχή δηλαδή δεν ταιριάζουν μεταξύ τους με διάφορους τρόπους, ορίζεται ως η διαδικασία για το πώς μπορούμε να δεχτούμε κάποια από τα στοιχεία και να απορρίψουμε άλλα με έναν τρόπο που να μεγιστοποιείται η συνοχή. Η μεγιστοποίηση της συνοχής επιτυγχάνεται με τη διαίρεση των στοιχείων σε αποδεκτά και μη αποδεκτά σύνολα ώστε να ικανοποιούνται οι περιορισμοί του προβλήματος. Οι περιορισμοί αυτοί είναι αν δύο στοιχεία ταιριάζουν, υπάρχει ένας θετικός περιορισμός μεταξύ τους. Αν δύο στοιχεία δεν ταιριάζουν υπάρχει ένας αρνητικός περιορισμός μεταξύ τους. Ένας θετικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί με την αποδοχή και

των δύο στοιχείων ή απορρίπτοντας και τα δυο στοιχεία. Ένας αρνητικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί μόνο με την αποδοχή του ενός στοιχείου και την απόρριψη του άλλου στοιχείου.

Βασική πρόκληση στην διπλωματική εργασία αυτή ήταν η μοντελοποίηση του Προβλήματος Συνοχής μέσω διαφόρων μεθόδων ικανοποίησης περιορισμών σε διάφορους επιλυτές με στόχο την εύρεση της βέλτιστης λύσης σε αποδοτικό χρόνο.

Στην παρούσα εργασία μελετάται η επίλυση του προβλήματος αυτού ως πρόβλημα ικανοποίησης περιορισμών (CSP) με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών σε διάφορους επιλυτές όπως στον MiniZinc, στον Gecode, στον Gecode με χρήση συνδυαστών αναζήτησης, αλλά και με επίλυση μέσω ψευδοδυαδικών περιορισμών και με χρήση ψευδοδυαδικής βελτιστοποίησης στον Minisat+, στον Bsolo και στον npSolver ώστε να προσδιοριστεί μέσα από διάφορα πειράματα που έγιναν η καλύτερη μέθοδος εύρεσης βέλτιστης λύσης σε αποδοτικό χρόνο.

1.2 Δομή της Διπλωματικής εργασίας

Αρχικά το δεύτερο κεφάλαιο της διπλωματικής είναι βασισμένο στο άρθρο [9] για το Πρόβλημα της Συνοχής ως πρόβλημα ικανοποίησης περιορισμών. Στο κεφάλαιο αυτό παρουσιάζεται η θεωρητική σκοπιά του προβλήματος αυτού, δίνεται ο τυπικός ορισμός του τι πραγματικά ορίζεται ως Πρόβλημα Συνοχής. Φαίνεται σε ποιους τομείς στην καθημερινότητα μας χρησιμοποιείται η θεωρεία του Προβλήματος της Συνοχής. Παρουσιάζονται κάποιοι αλγόριθμοι για την μεγιστοποίηση της συνοχής, καθώς επίσης διατυπώνονται τα τρία είδη μέτρησης της συνοχής.

Όπως αναφέρεται σκοπός της διπλωματικής ήταν η επίλυση του Προβλήματος Συνοχής μέσω διάφορων τεχνικών ικανοποίησης περιορισμών, πιο συγκεκριμένα γίνεται μοντελοποίηση του προβλήματος αυτού ως πρόβλημα ικανοποίησης περιορισμών με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών και μοντελοποίηση μέσω ψευδοδυαδικών περιορισμών. Έτσι στο κεφάλαιο 3 δίνεται με σαφήνεια η έννοια του Προβλήματος Ικανοποίησης Περιορισμών καθώς και η έννοια των ψευδοδυαδικών περιορισμών.

Δίνεται ο ορισμός των προβλημάτων ικανοποίησης περιορισμών, καθώς και τι ορίζεται ως λύση σε ένα πρόβλημα ικανοποίησης περιορισμών. Αναφέρονται οι δύο βασικοί λόγοι αναπαράστασης και επίλυσης προβλημάτων σε προβλήματα ικανοποίησης

περιορισμών καθώς και εφαρμογές των προβλημάτων ικανοποίησης περιορισμών στην καθημερινότητά μας. Επιπλέον στο κεφάλαιο αυτό για τον λόγο ότι ο επιλυτής Minisat+ είναι προτασιακός επιλυτής αλλά έχει την δυνατότητα να χειριστεί ψευδοδυαδικούς περιορισμούς, στην ενότητα αυτή αναφέρεται και η έννοια της προτασιακής ικανοποιησιμότητας, ο αλγόριθμος DPLL που βασίζονται οι SAT επιλυτές, καθώς επίσης και λόγοι που οδήγησαν στην επέκταση των προτασιακών επιλυτών με δομές για υποστήριξη ψευδοδυαδικών περιορισμών.

Επιπρόσθετα στο κεφάλαιο αυτό δίνεται ο ορισμός ενός ψευδοδυαδικού περιορισμού. Δίνονται και επεξηγούνται οι μορφές των δύο ειδών ψευδοδυαδικών περιορισμών, γραμμικών και μη γραμμικών. Επίσης επεξηγείται η λειτουργία του αλγορίθμου επίλυσης των ψευδοδυαδικών περιορισμών. Γίνεται αναφορά για τους ψευδοδυαδικούς επιλυτές σε προβλήματα αποφασισιμότητας και σε προβλήματα βελτιστοποίησης.

Στο κεφάλαιο 4 παρουσιάζονται λίγα λόγια για τον κάθε επιλυτή καθώς επίσης και η μορφή του αρχείου που δέχεται ως είσοδο ο κάθε επιλυτής. Ακολούθως φαίνεται ο τρόπος με τον οποίο έγινε η μοντελοποίηση του Προβλήματος της Συνοχής σε αυτές τις μορφές αρχείων, με βάση τους περιορισμούς του Προβλήματος Συνοχής και το ζητούμενο του προβλήματος, δηλαδή την μεγιστοποίηση της συνοχής, άρα την ικανοποίηση όσων πιο πολλών περιορισμών, ώστε να έχουμε ελαχιστοποίηση του κόστους. Συγκεκριμένα για τον επιλυτή bsolo_pb10 αναλύονται οι παράμετροι με τους οποίους μπορεί να τρέξει ώστε να βρει πιο γρήγορα την βέλτιστη λύση.

Στο κεφάλαιο 5 παρουσιάζεται η πειραματική ανάλυση των αποτελεσμάτων με τα συμπεράσματα που προέκυψαν για κάθε πείραμα. Πιο αναλυτικά αρχικά φαίνονται συγκρίσεις ως προς τους χρόνους εκτέλεσης αναζήτησης και έβρεσης της βέλτιστης λύσης στον επιλυτή MiniZinc για τρία μοντέλα .mzn που υλοποιούν το Προβλήματος Συνοχής με διαφορετικό τρόπο κωδικοποιημένα στην γλώσσα MiniZinc. Φαίνεται μια πειραματική ανάλυση ως προς τον χρόνο αναζήτησης βέλτιστης λύσης στον επιλυτή Bsolo_pb10 , ο οποίος παίρνοντας ως είσοδο το αρχείο σε μορφή ψευδοδυαδικών περιορισμών και ψευδοδυαδικής συνάρτηση βελτιστοποίησης γιο το Πρόβλημα της Συνοχής και με χρήση κάποιων παραμέτρων φαίνεται η διαφορά στην πιο γρήγορη αναζήτηση. Παρουσιάζονται οι χρόνοι εκτέλεσης αναζήτησης βέλτιστης λύσης αλλάζοντας την πυκνότητα των στοιχείων, δηλαδή το ποσοστό σύνδεσης των στοιχείων. Επιπρόσθετα έγινε μια πειραματική μέτρηση ώστε να διαπιστωθεί για το πώς επηρεάζονται οι επιλυτές ως προς το χρόνο εύρεσης βέλτιστης λύσης για το

Πρόβλημα της Συνοχής αλλάζοντας την απεικόνιση του ποσοστού συσχέτισης των στοιχείων. Τα στοιχεία συνδέονται μεταξύ τους (βάρος ακμών) με θετικούς και αρνητικούς ακεραίους από -10 μέχρι 10 ή από -1 μέχρι 1. Έτσι στο τελευταίο πείραμα κρατώντας την πυκνότητα των στοιχείων 100% και αλλάζοντας την απεικόνιση των συσχετίσεων των στοιχείων σε -1 έως 1 , προέκυψαν κάποιες παρατηρήσεις σε σχέση με τους χρόνους επίλυσης των επιλυτών με πυκνότητα στοιχείων 100% και απεικόνιση των συσχετίσεων των στοιχείων με -10 έως 10.

Στο τελευταίο κεφάλαιο διατυπώνονται γενικά συμπεράσματα που προέκυψαν στην παρούσα διπλωματική για το Πρόβλημα της Συνοχής και προτείνονται επίσης μελλοντικές εργασίες ως επέκταση στη διπλωματικής αυτή.

Κεφάλαιο 2

Πρόβλημα Συνοχής

2.1 Εισαγωγή…….....5
2.2 Συνοχή ως Ικανοποίηση Περιορισμών…….....6
2.3 Τυπικός ορισμός του Προβλήματος Συνοχής…….……..7
2.4 Παραδείγματα Προβλημάτων Συνοχής…….....8
2.5 Αλγόριθμοι Υπολογισμού της Συνοχής…….....11

2.1 Εισαγωγή

Σε πάρα πολλούς τομείς υπήρξαν υποστηρικτές του Προβλήματος Συνοχής. Υπήρξαν υποστηρικτές της θεωρίας της συνοχής στη θεωρία της αλήθειας, επίσης υποστηρικτές υπήρξαν και στον τομέα της επιστημολογίας, όπου υποστήριξαν την θεωρεία της συνοχής μέσα από επιστημονική αιτιολογία (epistemic justification). Ακόμα στον τομέα της λογικής καθώς και ορισμένοι επιστήμονες της ηθικής έχουν προσπαθήσει να υπερασπιστούν τις γενικές ηθικές αρχές με βάση την θεωρία της συνοχής, με ηθικές αποφάσεις, όπου η συνοχή αποφασίζεται μέσα από μια διαδικασία για την επίτευξη μιας στοχαστικής ισορροπίας.

Οι ψυχολόγοι έχουν ασχοληθεί έντονα με την έννοια της συνοχής για την κατανόηση των διαδικασιών αντίληψης του κειμένου και αντίληψης του λόγου.

Η έννοια της Συνοχής ορίζεται ως η διαδικασία λήψης μεγάλου αριθμού στοιχείων που ταιριάζουν ή δεν ταιριάζουν μεταξύ τους με διάφορους τρόπους, και πώς μπορούμε να δεχτούμε κάποια από τα στοιχεία αυτά και να απορρίψουμε άλλα με έναν τρόπο ώστε να μεγιστοποιείται η συνοχή. Υπάρχουν πάρα πολλοί αλγόριθμοι οι οποίοι δίνουν διαφορετικές μέθοδοι για την μέτρηση της συνοχής.

Τα Προβλήματα Συνοχής δεν λύνονται σε πολυωνυμικό χρόνο. Είναι δυσεπίλυτα υπολογιστικά προβλήματα με την έννοια ότι στο πλαίσιο μιας ευρέως διαδεδομένης παραδοχής της υπολογιστικής θεωρίας πολυπλοκότητας δεν υπάρχουν αποτελεσματικές

διαδικασίες πολυωνυμικού χρόνου για επίλυσή τους. Υπάρχει απόδειξη ότι το Πρόβλημα της Συνοχής εντάσσεται στα NP-δύσκολα προβλήματα αλλά υπάρχουν ωστόσο αρκετοί αποτελεσματικοί αλγόριθμοι για την μεγιστοποίηση Προβλημάτων της Συνοχής.

2.2 Συνοχή ως Ικανοποίηση Περιορισμών

Για την αντίληψη ενός κειμένου, μιας εικόνας, ενός ατόμου, ή ενός γεγονότος δημιουργούμαι μια ερμηνεία που να ταιριάζει με τις διαθέσιμες πληροφορίες. Η καλύτερη ερμηνεία είναι αυτή που παρέχει την πιο συνεπή αιτιολογία του τι θέλουμε να κατανοήσουμε , λαμβάνοντας υπόψη τόσο κομμάτια των πληροφοριών που ταιριάζουν όσο και αυτά που δεν ταιριάζουν μεταξύ τους.

Για παράδειγμα όταν συναντούμε άγνωστους ανθρώπους μέσω ενός δικού σου συνδυασμού σεναρίων και υποθέσεων για τον άνθρωπο αυτό μπορείς να φτιάξεις μια γενική εικόνα για το χαρακτήρα του.

Η Συνοχή μπορεί να γίνει κατανοητή μέσω της μέγιστης ικανοποίησης των περισσότερων περιορισμών με τέτοιο τρόπο όπως συνοψίζεται πιο κάτω:

Τα στοιχεία αναπαριστούν έννοιες, προτάσεις, μέρος μιας εικόνας, στόχους, δράση κτλ

Τα στοιχεία μπορεί να ταιριάζουν μεταξύ τους ή να μην ταιριάζουν μεταξύ τους. Τα στοιχεία που ορίζονται ως ταιριαστά περιλαμβάνουν επεξήγηση, αφαίρεση, διευκόλυνση, συνεταιρισμό ενώ τα στοιχεία που είναι αταίριαστα περιλαμβάνουν ασυνέπεια, είναι ασυμβίβαστα και έχουν αρνητική συσχέτιση.

Αν δύο στοιχεία ταιριάζουν, υπάρχει ένας θετικός περιορισμός μεταξύ τους. Αν δύο στοιχεία δεν ταιριάζουν υπάρχει ένας αρνητικός περιορισμός μεταξύ τους.

Τα στοιχεία χωρίζονται σε αυτά που γίνονται αποδεκτά και σε αυτά που απορρίπτονται.

Ένας θετικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί με την αποδοχή και των δύο στοιχείων ή απορρίπτοντας και τα δυο στοιχεία.

Ένας αρνητικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί μόνο με την αποδοχή του ενός στοιχείου και την απόρριψη του άλλου στοιχείου.

Το Πρόβλημα της Συνοχής συνίσταται στη διαίρεση ενός συνόλου στοιχείων σε αποδεκτά και μη αποδεκτά σύνολα με ένα τρόπο που να ικανοποιεί τους περισσότερους περιορισμούς. [9]

2.3 Τυπικός ορισμός του Προβλήματος Συνοχής

Τυπικά μπορούμε να ορίσουμε ένα Πρόβλημα Συνοχής ως εξής: Έστω E είναι ένα πεπερασμένο σύνολο από στοιχεία {ei} και C είναι ένα σύνολο από περιορισμούς στο E που απεικονίζονται ως σύνολα {<ei, ej>} ζευγαριών στο E. Το C χωρίζεται σε C+ , που περιλαμβάνει τους θετικούς περιορισμούς και σε C-, που περιλαμβάνει τους αρνητικούς περιορισμούς στο E. Κάθε περιορισμός συνδέεται με ένα αριθμό w που αντιστοιχεί στο βάρος του περιορισμού. Το πρόβλημα είναι να χωρίσεις το E σε δύο σύνολα A (αποδεκτά) και R (μη αποδεκτά) με τρόπο ώστε να τηρείται η μεγιστοποίηση μέσω δυο προϋποθέσεων της συνοχής που πρέπει να ισχύουν και είναι οι εξής:

1. Εάν (ei,ej) είναι στο σύνολο C+, τότε ei ανήκει στο A αν και μόνο αν ej είναι στο A.
2. Εάν (ei,ej) είναι στο σύνολο C-, τότε ei ανήκει στο A αν και μόνο αν ej είναι στο R.

Όπως αναφέρεται και πιο πάνω κάθε περιορισμός συνδέεται με έναν αριθμό w που αντιστοιχεί στο βάρος του περιορισμού. Τώρα έστω W είναι το άθροισμα των βαρών των ικανοποιημένων περιορισμών (θετικών και αρνητικών περιορισμών). Το πρόβλημα της Συνοχής είναι να διαιρέσεις το E στο A και στο R με τρόπο ώστε να μεγιστοποιείται το W. Αν δύο στοιχεία συνδέονται θετικά τότε θέλουμε να γίνουν και τα δύο αποδεκτά ή να απορριφτούν και τα δυο. Άρα να μπουν και τα δύο στοιχεία στο σύνολο A ή και τα δύο στο σύνολο R.
Αν τώρα δύο στοιχεία συνδέονται μεταξύ τους αρνητικά, έχουν αρνητικό περιορισμό μεταξύ τους, τότε θα θέλουμε να μπει το ένα στο σύνολο A και το άλλο στο σύνολο R.[9]

Η συνοχή μεγιστοποιείται αν δεν υπάρχει άλλος διαμερισμός που να έχει μεγαλύτερο συνολικό άθροισμα βαρών από τους ικανοποιημένους περιορισμούς.

Αυτή η αφαιρετική έννοια που περιγράφεται πιο πάνω αφορά άμεσα πρωτογενή φιλοσοφικές και ψυχολογικές συζητήσεις για το Πρόβλημα της Συνοχής. Για να δείξουν ότι ένα πρόβλημα εντάσσεται στην κατηγορία του Προβλήματος της Συνοχής με την πιο πάνω έννοια, είναι αναγκαίο να καθοριστούν τα στοιχεία και οι περιορισμοί,

να διευκρινιστεί η έννοια της αποδοχής και της απόρριψης και ακολούθως οι λύσεις στο πρόβλημα που δίνονται συνεπάγονται την ικανοποίηση των συγκεκριμένων περιορισμών.

2.4 Παραδείγματα Προβλημάτων Συνοχής

Σε πολλούς τομείς στην καθημερινότητά μας πολλά προβλήματα εντάσσονται στην κατηγορία των Προβλημάτων Συνοχής. Κάποια από αυτά αναφέρονται στον πίνακα 2.1 και επεξηγούνται πιο κάτω.[9]

Problem	Elements	Positive constraints	Negative constraints	Accepted as
Truth	Propositions	Entailment, etc	Inconsistency	True
Epistemic justification	Propositions	Entailment, explanation, etc	Inconsistency, competition	Know
Mathematics	Axioms, theorems	Deduction	Inconsistency	Know
Logical justification	Principles, practices	Justify	Inconsistency	Justified
Ethical justification	Principles, judgments	Justify	Inconsistency	Justified
Practical reasoning	Actions,goals,images	Facilitations,connectedness,parts	Incompatibility	Desirable
Perception	Images	connectedness, parts	Inconsistency	seen
Discourse comprehension	Meanings	semantic relatedness	Inconsistency	understood
Cognitive dissonance	beliefs, attitudes	Consistency	Inconsistency	believed
Impression formation	stereotypes, traits	Association	negative association	believed
Democratic deliberation	actions,goals, propositions	facilitation, explanation	incompatible actions and beliefs	joint action

Πίνακας 2.1:Απεικόνιση ειδών Προβλημάτων Συνοχής

Σε πάρα πολλούς τομείς εφαρμόζεται η θεωρεία της Συνοχής.

Στη θεωρεία της συνοχής της αλήθειας (Πίνακας 2.1) τα στοιχεία είναι προτάσεις και αποδεκτές προτάσεις ερμηνεύονται ως αληθής ενώ απορρίπτονται προτάσεις που ερμηνεύονται ως ψευδής.

Η επιστημονική αιτιολόγηση (Epistemic justification) (Πίνακας 2.1) εντάσσεται στην κατηγορία προβλημάτων της συνοχής και ορίζεται όπως στο κεφάλαιο 2.3. Εδώ το Ε είναι ένα σύνολο από προτάσεις και οι θετικοί και αρνητικοί περιορισμοί είναι ένα σύνολο σχέσεων μεταξύ των προτάσεων αυτών. Στους θετικούς περιορισμούς συμπεριλαμβάνεται και η σχέση συνεπαγωγής, αλλά και πιο πολύπλοκες σχέσεις, όπως η εξήγηση (explanation). Οι αρνητικοί περιορισμοί μπορεί να περιλαμβάνουν ασυνέπεια, είτε επειδή δύο υποθέσεις έρχονται σε αντίθεση μεταξύ τους ή επειδή ανταγωνίζονται η μια την άλλη για να εξηγήσουν τα ίδια στοιχεία.

Επιπρόσθετα ο Irvine (1994) υποστήριξε ότι η αιτιολόγηση των μαθηματικών αξιωμάτων (Πίνακας 2) είναι ομοίως ένα πρόβλημα που εντάσσεται στην κατηγορία αυτή. Τα αξιώματα δεν γίνονται δεχτά επειδή είναι αληθής, αλλά επειδή χρησιμεύουν για να παράγουν και να συστηματοποιούν ενδιαφέροντα θεωρήματα τα οποία δικαιολογούνται κατά ένα μεγάλο μέρος επειδή ακολουθούν τα αξιώματα αυτά.

Ακόμα και η Λογική Αιτιολόγηση (Logical justification) (Πίνακας 2.1) μπορεί να θεωρηθεί ως Πρόβλημα Συνοχής. Τα στοιχεία είναι λογικοί κανόνες και αποδεκτά συμπεράσματα. Οι θετικοί περιορισμοί προέρχονται από αιτιολόγηση σχέσεων που διατηρούν μεταξύ τους συγκεκριμένοι κανόνες, ενώ οι αρνητικοί περιορισμοί προέρχονται λόγω του ότι ορισμένοι κανόνες και συμπεράσματα είναι ασύμβατα, αταίριαστα μεταξύ τους.

Επίσης και ο Rawls(1971) υποστήριξε ότι οι ηθικές αρχές (Ethical justification) (Πίνακας 2.1) μπορούν να αναθεωρηθούν και να γίνουν αποδεχτές με βάση το ταίριασμα τους με ηθικές κριτικές. Θεωρίες της συνοχής έχουν επίσης προταθεί και στον τομέα αυτό, κρατώντας το νόμο να είναι το σύνολο των αρχών που κάνει πιο συνεκτική την έννοια των δικαστικών αποφάσεων και των νομοθετικών και κανονιστικών πράξεων.

Ο Millgram και Thagard (1995, 1996) υποστήριξαν ότι η λογική της πρακτικής (Practical Reasoning) (Πίνακας 2.1) περιλαμβάνει επίσης τις αποφάσεις της συνοχής για το πώς να ταιριάζουν μεταξύ τους διάφορες πιθανές δράσεις και στόχοι . Τα

στοιχεία είναι δράσεις και στόχοι , οι θετικοί περιορισμοί είναι σχέσεις διευκόλυνσης (δράση Α διευκολύνει στόχο Β) , και οι αρνητικοί περιορισμοί είναι ασύμβατες σχέσεις, όπως για παράδειγμα «δεν μπορείτε να είστε στην Ελλάδα και στην Κύπρο την ίδια στιγμή». Η απόφαση για το τι πρέπει να γίνει βασίζεται στο συμπέρασμα για το πιο συνεκτικό σχέδιο, όπου η συνοχή περιλαμβάνει την αξιολόγηση των στόχων.

Στην ψυχολογία (Πίνακας 2.1), διάφορες διαδικασίες αντίληψης (Perception), όπως η στερεοσκοπική όραση και η ερμηνεία ασαφή στοιχείων ερμηνεύεται μέσω της θεωρίας της συνοχής και την ικανοποίηση περιορισμών. Εδώ τα στοιχεία είναι υποθέσεις για το τι πρόκειται να δει το άτομο. Οι θετικοί περιορισμοί αφορούν διάφορους τρόπους με τους οποίους οι εικόνες μπορούν να τοποθετηθούν μαζί. Αρνητικοί περιορισμοί αφορούν ασύμβατους τρόπους συνδυασμού εικόνων.

Επιπρόσθετα η αντίληψη μιας λέξης (Word Perception) μπορεί να θεωρηθεί ως ένα πρόβλημα συνοχής στο οποίο γίνονται υποθέσεις σχετικά με το πώς τα γράμματα των λέξεων μπορούν να αξιολογηθούν το ένα εναντίον του άλλου βάση των περιορισμών σχετικά με τις διασυνδέσεις των γραμμάτων.
Ο Kintsch (1988) πιο αναλυτικά αναφέρει το πρόβλημα αυτό ως ένα πρόβλημα με ταυτόχρονη ανάθεση συμπληρωματικών εννοιών σε διαφορετικές λέξεις. Για παράδειγμα, η φράση "η πένα είναι στην τράπεζα» μπορεί να σημαίνει ότι χρησιμοποιείται η πένα για γραφή στο χρηματοπιστωτικό ίδρυμα αυτό, αλλά σε διαφορετικό περιβάλλον αυτό μπορεί να σημαίνει ότι το ζώο βρίσκεται στην πλευρά του ποταμού. Σε αυτό το Πρόβλημα Συνοχής , τα στοιχεία έχουν διαφορετικές σημασίες των λέξεων και οι θετικοί περιορισμοί δίδονται από την έννοια των συνδέσεων ανάμεσα στις λέξεις, όπως «τράπεζα» και «ποταμός».

Θεωρία της συνοχής είχε εφαρμογή σε πρόσφατες εργασίες στην κοινωνική ψυχολογία (Πίνακας 2.1).Υπάρχουν πειραματικά αποτελέσματα που αφορούν τις διαπροσωπικές σχέσεις που ερμηνεύονται μέσω της επεξηγηματικής συνοχής. Τα στοιχεία στο πρόβλημα συνεκτικότητας είναι οι πεποιθήσεις και οι συμπεριφορές. Η μείωση της ασυμφωνίας είναι θέμα ικανοποίησης των διαφόρων θετικών και αρνητικών περιορισμών.

Ένα άλλο είδος Προβλήματος Συνοχής το οποίο έδειξαν ο Kunda και Thagard (1996) είναι ο σχηματισμός της εντύπωση για ένα άτομο (Impression formation) (Πίνακας 2.1), στην οποία οι άνθρωποι κάνουν κριτικές για άλλους ανθρώπους βάση κάποιων πληροφοριών μέσω στερεότυπων, γνωρισμάτων και συμπεριφορών. Τα στοιχεία στο σχηματισμό της εντύπωσης ορίζονται ως διάφορα χαρακτηριστικά για ένα άτομο. Οι θετικοί περιορισμοί προέρχονται από συσχετίσεις μεταξύ των χαρακτηριστικών, ενώ αρνητικοί περιορισμοί προέρχονται από τις κακές συσχετίσεις των χαρακτηριστικών. Για παράδειγμα αν μας πουν η «Η Μαρία είναι "καλογριά" , "κλέφτρα", ο συνδυασμός των δύο αυτών χαρακτηριστικών είναι κακός συσχετισμός χαρακτηριστικών αφού το "καλογριά" παραπέμπει σε κάτι ηθικό , ενώ το "κλέφτρα", παραπέμπει σε κάτι ανήθικο.

Τέλος, σημαντικά πολιτικά και οικονομικά προβλήματα (Democratic deliberation) (Πίνακας 2.1) μπορούν επίσης να εξηγηθούν και πάλι από την άποψη της παράλληλης ικανοποίησης περιορισμών.
Ο Arrow (1963) έδειξε ότι οι σύνηθες παραδοχές που χρησιμοποιούνται στα οικονομικά μοντέλα κοινωνικής πρόνοιας είναι από κοινού ασυνεπής. Ο Mackie υποστηρίζει ότι η διαβουλευτική δημοκρατία δεν πρέπει να θεωρείται μέρος της εξιδανίκευσης της πλήρους συναίνεσης, αλλά με όρους μιας ομαδικής διαδικασίας να ικανοποιούνται όσο πιο πολλά θετικοί και αρνητικοί περιορισμοί.

2.5 Αλγόριθμοι Υπολογισμού Συνοχής

Αν η Συνοχή μπορεί πράγματι να χαρακτηριστεί σε γενικές γραμμές ως η ικανοποίηση των πολλαπλών θετικών και αρνητικών περιορισμών, μπορούμε να αντιμετωπίσουμε με ακρίβεια το ζήτημα του πως μπορεί η συνοχή να υπολογιστεί. Αρκεί να γίνει επιλογή των αποδεχτών και μη αποδεκτών στοιχείων με τρόπο ώστε να μεγιστοποιείται η συνοχή με βάση την τήρηση των δύο προϋποθέσεων ικανοποίησης περιορισμών που αναφέρθηκαν στο υποκεφάλαιο 2.3.

Υπάρχουν πέντε αλγόριθμοι για τη μεγιστοποίηση της συνοχής.[9]
Ο πρώτος αλγόριθμος είναι ο εξαντλητικός αλγόριθμος (exhaustive algorithm), που εκτελεί εξαντλητική αναζήτηση και εξετάζει όλες τις πιθανές λύσεις. Δημιουργεί όλους τους πιθανούς τρόπους διαίρεσης στοιχείου σε αποδεχτά και μη αποδεχτά σύνολα.

Ακολούθως αξιολογεί κάθε ένα από αυτά ως προς το βαθμό που επιτυγχάνεται η συνοχή και διαλέγει αυτό με την ψηλότερη τιμή του W.

Ο δεύτερος αλγόριθμος είναι στοιχειώδης αλγόριθμος (incremental algorithm) που εξετάζει τα στοιχεία με αυθαίρετη σειρά. Παίρνει μια αυθαίρετη διάταξη των στοιχείων e1,e2…en στο σύνολο E. Ορίζει δυο σύνολα A και R, αρχικά είναι κενά και θα περιλαμβάνουν τα αποδεχτά και τα μη αποδεκτά στοιχεία. Ακολούθως για κάθε στοιχείο ei αν η προσθήκη του στο σύνολο A αυξάνει το συνολικό βάρος των ικανοποιημένων περιορισμών περισσότερο από την προσθήκη του ei στο R τότε το στοιχείο ei μπαίνει στο σύνολο A αλλιώς αν δεν ισχύει η συνθήκη, δηλαδή η προσθήκη του στοιχείου ei στο σύνολο A μειώνει το συνολικό βάρος των ικανοποιημένων περιορισμών τότε το ei μπαίνει στο σύνολο R.

Ο τρίτος αλγόριθμος ο οποίος είναι αλγόριθμος σύνδεσης (connectionist algorithm) ο οποίος χρησιμοποιεί ένα τεχνητό νευρωνικό δίκτυο για την αξιολόγηση της συνοχής.

Ο τέταρτος αλγόριθμος είναι άπληστος αλγόριθμος (greedy algorithm) ο οποίος χρησιμοποιεί τοπικά βέλτιστες επιλογές για να προσεγγίσει όσο το δυνατό βέλτιστη λύση.

Και ο τελευταίος αλγόριθμος ο οποίος εγγυάται ότι θα ικανοποιήσει ένα υψηλό ποσοστό περιορισμών (semidefinite programming-SDP).

Στην πράξη ο πρώτος και ο δεύτερος αλγόριθμος δεν χρησιμοποιούνται πολύ σε σχέση με τους υπόλοιπους που η χρήση τους είναι σημαντική για τον υπολογισμό της συνοχής. Στο παρελθόν η θεωρεία της συνοχής είχε αρκετές ασάφειες σε σύγκριση με τις πιο αυστηρές θεωρίες λογικής και λογισμού πιθανοτήτων. Το τελευταίο διάστημα όμως οι θεωρίες της συνοχής είναι πιο πλήρεις χαρακτηρίζοντας την συνοχή μέσω της ικανοποίησης περιορισμών χρησιμοποιώντας τους πιο πάνω αλγορίθμους.

Κεφάλαιο 3

Προβλήματα Ικανοποίησης Περιορισμών και Προτασιακή Ικανοποιησιμότητα και Ψευδοδυαδικοί Περιορισμοί

3.1 Ορισμός Προβλήματος Ικανοποίησης Περιορισμών…….....13
3.2 Κλασικά Παραδείγματα ΠΙΠ……...15
3.3 Λόγοι αναπαράστασης και επίλυσης προβλημάτων ως Προβλήματα Ικανοποίησης Περιορισμών…….....16
3.4 Εφαρμογές ΠΙΠ…….....17
3.5 Προτασιακή Ικανοποιησιμότητα……..……17
3.5.1 Αλγόριθμος DPLL…….……...18
3.5.2 Λόγοι επέκτασης προτασιακών επιλυτών με χρήση ψευδοδυαδικών περιορισμών……..…….19
3.6 Ορισμός Ψευδοδυαδικών περιορισμών…….…….....20
3.7 Γραμμικοί Ψευδοδυαδικοί περιορισμοί……..……..……..20
3.8 Μη γραμμικοί Ψευδοδυαδικοί περιορισμοί…….....21
3.9 Κανονικοποίηση ψευδοδυαδικών περιορισμών…….....22
3.10 Αλγόριθμος ψευδοδυαδικών επιλυτών…….……23
3.11 Ψευδοδυαδικοί επιλυτές σε προβλήματα αποφασισιμότητας και σε προβλήματα βελτιστοποίησης…….....23

3.1 Ορισμός Προβλήματος Ικανοποίησης Περιορισμών

Ένας από τους τομείς της Τεχνητής Νοημοσύνης ασχολείται με τα προβλήματα ικανοποίησης περιορισμών. Τα προβλήματα ικανοποίησης περιορισμών είναι προβλήματα αναζήτησης τα οποία χαρακτηρίζονται για την απλή δομή τους και την απλή τυπική τους αναπαράσταση. Είναι ένα είδος δηλωτικού προγραμματισμού όπου μοντελοποιούμε το πρόβλημα με τις μεταβλητές και τους περιορισμούς και ακολούθως

ο επιλυτής αναλαμβάνει να επιλύσει το πρόβλημα αναθέτοντας στις μεταβλητές τιμές ώστε να ικανοποιούνται οι περιορισμοί.

Καλούμαστε να απεικονίσουμε τους περιορισμούς οι οποίοι περιέχουν ένα υποσύνολο μεταβλητών και προσδιορίζουν τις επιτρεπόμενες τιμές για τις μεταβλητές αυτές. Κάθε μεταβλητή έχει το δικό της πεδίο ορισμού και οι τιμές οι οποίες ανατίθενται στις μεταβλητές είναι μέσω του πεδίου ορισμού της κάθε μεταβλητής, το οποίο μπορεί να οριστεί είτε με ακέραιες είτε με πραγματικές τιμές.

Η διαδικασία μοντελοποίησης του προβλήματος είναι βασική διαδικασία η οποία χρησιμοποιείται για επίλυση πολλών συνδυαστικών προβλημάτων.

Ο τυπικός ορισμός ενός προβλήματος ικανοποίησης περιορισμών ορίζεται ως εξής:

- ένα σύνολο μεταβλητών X = { X 1 , X2 , … , Xn }
- Για κάθε μεταβλητή Xi ένα περιορισμένο σύνολο Di των πιθανών τιμών της συγκεκριμένης μεταβλητής.
- ένα σύνολο περιορισμών C = { C1 , C2 , … , Cn }. Κάθε περιορισμός δρα σε ένα σύνολο K ⊆ X και καθορίζει τους επιτρεπτούς συνδυασμούς τιμών που μπορούν να πάρουν οι μεταβλητές που ανήκουν στο K.

Ανάλογα με των αριθμό των μεταβλητών που περιλαμβάνονται σε ένα περιορισμό, μπορεί να χαρακτηριστεί ο περιορισμός αυτός ως μοναδιαίος, δυαδικός ή περιορισμός ανώτερης τάξης.[17]

- Μοναδιαίοι περιορισμοί αφορούν μόνο μία μεταβλητή σε ένα περιορισμό όπως φαίνεται και πιο κάτω:

$$SA \neq green$$

- Δυαδικοί περιορισμοί αφορούν ζευγάρια μεταβλητών σε ένα περιορισμό:

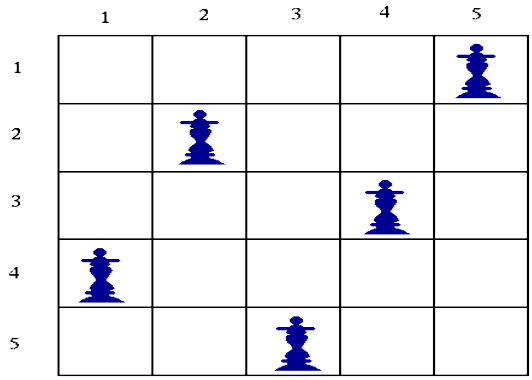
$$SA \neq WA$$

- Περιορισμοί ανώτερης τάξης οι οποίοι αφορούν τρεις ή περισσότερες μεταβλητές σε ένα περιορισμό.

Λύση σε ένα πρόβλημα ικανοποίησης περιορισμών είναι μια ανάθεση τιμών στις μεταβλητές η οποία ικανοποιεί όλους τους περιορισμούς ονομάζεται συνεπής, αλλιώς ονομάζεται ασυνεπής. Λύση σε ένα πρόβλημα ικανοποίησης περιορισμών είναι μια συνεπής ανάθεση τιμών σε όλες τις μεταβλητές. Υπάρχουν δύο κατηγορίες αναζήτησης λύσης. Η μια κατηγορία είναι σε αυτούς τους επιλυτές που διασχίζουν το χώρο με μερική ανάθεση τιμών και σε αυτούς που εξερευνούν το χώρο με πλήρης ανάθεση τιμών σε όλες τις μεταβλητές.

3.2 Κλασικά Παραδείγματα Προβλημάτων Ικανοποίησης Περιορισμών (ΠΙΠ)

Ένα από τα πιο κλασικά προβλήματα ικανοποίησης περιορισμών είναι το πρόβλημα των N Βασιλισσών.[12,20]



Σχήμα 3.1: Απεικόνιση 5-Βασιλισσών

Το πρόβλημα αυτό απαιτεί να τοποθετηθούν N Βασίλισσες σε μια σκακιέρα NXN χωρίς να απειλούν η μια την άλλη. Με άλλα λόγια κάθε βασίλισσα πρέπει να τοποθετηθεί με τέτοιο τρόπο στην σκακιέρα ώστε σε κάθε στήλη, γραμμή και διαγώνιο να υπάρχει μόνο μια βασίλισσα. Η τελική αναπαράσταση φαίνεται ξεκάθαρα στο Σχήμα 3.1.

Το πρόβλημα αυτό μέσω επίλυσης ως πρόβλημα ικανοποίησης περιορισμών απαιτεί να ορίσουμε τις μεταβλητές και το πεδίο ορισμού τους και τους περιορισμούς για το πρόβλημα. Οι μεταβλητές είναι όσες και οι βασίλισσες. Το πεδίο ορισμού των μεταβλητών είναι ίδιο για όλες και είναι οι γραμμές{1,2..N} στις οποίες θα τοποθετηθούν στην σκακιέρα. Οι περιορισμοί που ισχύουν για το πρόβλημα αυτό είναι:

-Όλες οι βασίλισσες πρέπει να είναι σε διαφορετική γραμμή:
Για κάθε i,j Qj≠Qi

-Ισχύουν επίσης οι περιορισμοί:
Qj ≠ Qj+n + n , για κάθε n>1 n+j≤N, όπου ο περιορισμός αυτός καλύπτει την περίπτωση μια βασίλισσα σε κάθε διαγώνιο
Και ο περιορισμός:
Qj≠ Qj-n - n , για κάθε n>1 n+j≤N, όπου ο περιορισμός αυτός καλύπτει την περίπτωση μια βασίλισσα σε κάθε στήλη

Ένα άλλο παράδειγμα ενός προβλήματος ικανοποίησης περιορισμών είναι ο χρωματισμός χάρτη, όπου οι μεταβλητές ορίζονται ως τα αρχικά της κάθε χώρας {WA,NT,SA,Q,NSW,V,T}, το πεδίο ορισμού είναι ίδιο για όλες τις μεταβλητές και αποτελείται από τρία χρώματα {red, green, blue} και οι περιορισμοί είναι WA ≠ NT, ομοίως για τα υπόλοιπα ζεύγη μεταβλητών.[13]

Τυπικός Ορισμός Χρωματισμού Χάρτη
Το παραπάνω πρόβλημα ικανοποίησης περιορισμών είναι συνεπές. Μια λύση του είναι:
{WA = red; NT = blue; SA = green; Q = red; NSW = blue; V = red; T = blue}

3.3 Λόγοι αναπαράστασης και επίλυσης προβλημάτων ως Προβλήματα Ικανοποίησης Περιορισμών

Υπάρχουν δύο τρόποι για τους οποίους επιλέγουμε αναπαράσταση και επίλυση προβλημάτων ως προβλήματα ικανοποίησης περιορισμών. Ο πρώτος λόγος είναι ότι η αναπαράσταση ως προβλήματα ικανοποίησης περιορισμών είναι συχνά πολύ πιο κοντά στο αρχικό πρόβλημα. Οι μεταβλητές του προβλήματος αντιστοιχούν άμεσα στο πρόβλημα και οι περιορισμοί μπορούν να εκφραστούν χωρίς να χρειάζεται να μεταφραστούν μέσω γραμμικών ανισοτήτων.
Ο δεύτερος λόγος αναπαράστασης ως προβλήματα ικανοποίησης περιορισμών είναι πως αν οι αλγόριθμοι προβλημάτων ικανοποίησης περιορισμών είναι απλοί, μπορούν να βρουν λύση μερικές φορές πιο γρήγορα από ότι αν χρησιμοποιούνταν μέθοδοι ακέραιου προγραμματισμού.

3.4 Εφαρμογές ΠΠ

Υπάρχουν πάρα πολλά παραδείγματα προβλημάτων ικανοποίησης περιορισμών. Κάποια από τα πολλά προβλήματα αυτά είναι το παράδειγμα του χρονοπρογραμματισμού εργασιών, οπού στο πρόβλημα αυτό λαμβάνοντας υπόψη ένα σύνολο από πόρους και δραστηριότητες οι οποίες απαιτούν τους πόρους αυτούς για να υλοποιηθούν, καθώς επίσης λαμβάνοντας υπόψη και ένα σύνολο από περιορισμούς ανάμεσα στις ενέργειες του προβλήματος αυτού, θα πρέπει να βρεθεί μια λύση στο πρόβλημα του χρονοπρογραμματισμού. Η λύση αυτή θα πρέπει να περιλαμβάνει των προγραμματισμό των διαφόρων ενεργειών με βάση τους διαθέσιμους πόρους με τέτοιο τρόπο ώστε να ικανοποιούνται οι περιορισμοί. Παρόμοιο πρόβλημα είναι η δημιουργία ωρολογίου προγράμματος (time tabling). Για παράδειγμα για την δημιουργία του ωρολογίου προγράμματος του Πανεπιστημίου Κύπρου θα πρέπει να βρεθεί μια λύση στο πρόβλημα αυτό ώστε να συντονιστούν όλα τα έτη από κάθε κλάδο με βάση τα μαθήματά τους, τους καθηγητές και τις διαθέσιμες αίθουσες διδασκαλίας, ώστε να ικανοποιούνται όλοι οι περιορισμοί για το πρόβλημα αυτό. Για παράδειγμα σε μια αίθουσα δεν μπορεί την ίδια ώρα να γίνονται δύο μαθήματα. Επίσης ένας καθηγητής δεν μπορεί να διδάσκει ταυτόχρονα σε δύο τμήματα κτλ. Τα προβλήματα ικανοποίησης περιορισμών χρησιμοποιούνται σε πάρα πολλούς τομείς όπως στην ικανοποιησιμότητα στην προτασιακή λογική, στην χρονική συλλογιστική, στην χωρική συλλογιστική. Επίσης αντιμετωπίζεται ως πρόβλημα ικανοποίησης περιορισμών και ο σχεδιασμός κυκλωμάτων ή ακόμα και η κατανομή προσωπικού σε επιχειρήσεις .[17]

3.5 Προτασιακή Ικανοποιησιμότητα

Το πρόβλημα Ικανοποιησιμότητας στην προτασιακή λογική είναι γνωστό ως SAT και αποτελεί ένα θέμα μελέτης εδώ και πάρα πολλά χρόνια. Το πρόβλημα της ικανοποιησιμότητας είναι ένα από τα βασικά προβλήματα στην θεωρεία υπολογισμού αλλά και στην μαθηματική λογική. Η προτασιακή ικανοποιησιμότητα μοντελοποιεί ένα ευρύ σύνολο προβλημάτων στο οποίο πρέπει να οριστούν οι μεταβλητές απόφασης έτσι ώστε να ικανοποιούν ένα δεδομένο σύνολο περιορισμών. Πιο ειδικά δεδομένου ενός σύνολο από δυαδικές μεταβλητές και ενός συνόλου από περιορισμούς που είναι σε κανονική διαζευκτική μορφή (Conjunctive Normal Form), στόχος είναι η εύρεση μιας

ανάθεσης τιμών που ικανοποιεί όλους τους περιορισμούς άρα το πρόβλημα είναι ικανοποιήσιμο, είτε η μη εύρεση μιας τέτοιας ανάθεσης τιμών η οποία να ικανοποιεί όλους τους περιορισμούς και άρα το πρόβλημα είναι μη ικανοποιήσιμο.[14]

Η μορφή των περιορισμών ακολουθούν ένα λογικό τύπο σε συζευκτική κανονική μορφή (CNF):
Ένας περιορισμός είναι ένα σύνολο ορών συζεύξεων από διαζεύξεις. Είναι η σύζευξη , το λογικό AND που συμβολίζεται με το σύμβολο «/\» μιας συλλογής όρων . Κάθε όρος αποτελείται από την διάζευξη, το λογικό OR, που συμβολίζεται με το σύμβολο «\/» διαφόρων στοιχείων που ονομάζονται λεκτικά (literals).
Κάθε στοιχείο είναι μια λογική μεταβλητή για παράδειγμα x ή η άρνηση της $\overline{x}$. Ένα παράδειγμα μορφής περιορισμού είναι:

$$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}),$$

Οι τιμές που παίρνει μια μεταβλητή είναι 0 που είναι η αντίστοιχη αποτίμηση ως ψευδής (faulse) ή 1 που είναι η αντίστοιχη αποτίμηση σε αληθής (true).
Ένας όρος (clause) ο οποίος είναι αληθής είναι και ικανοποιήσιμος, αν έστω και ένα από τα λεκτικά της (literals) πάρει την τιμή αληθής άρα την τιμή 1. Ένας όρος είναι μη ικανοποιήσιμο αν όλα τα λεκτικά που αποτελούν τον όρο είναι ψευδής.
Ένας όρος ορίζεται ως ικανοποιήσιμος όταν όλοι οι διαζευκτικές όροι που την αποτελούν προκύψουν αληθής και άρα ικανοποιήσιμος.

3.5.1 Αλγόριθμος DPLL (Davis-Putnam-Logemann-Loveland)

Ένας από τους πιο διαδεδομένους αλγόριθμος επίλυσης του προβλήματος της ικανοποιησιμότητας είναι ο DPLL (Davis-Putnam-Logemann-Loveland) και οι περισσότεροι SAT επιλυτές στηρίζονται στον αλγόριθμο αυτό. Ο αλγόριθμος αυτός χρησιμοποιεί την διαδικασία αναζήτησης με οπισθοδρόμηση.
Με το που θα γίνει μια ανάθεση τιμής σε μια μεταβλητή τότε γίνεται έλεγχος κατά πόσο επηρεάζει η ανάθεση αυτή τις υπόλοιπες μεταβλητές. Η μεταβλητή αντιπροσωπεύει ένα λεκτικό στον όρο (clause) το οποίο ανήκει και δεν έχει πάρει τιμή. Μοναδιαίο (unit) είναι όταν σε ένα όρο υπάρχει μόνο ένα λεκτικό (literal) ή όταν όλα τα λεκτικά σε ένα όρο εκτός από ένα είναι ψευδής. [18]

Εάν η ανάθεση της τιμής αυτής στην μεταβλητή οδηγήσει σε σύγκρουση με μια άλλη μεταβλητή τότε γίνεται οπισθοδρόμηση προς τα πίσω αφαιρώντας από την μεταβλητή την τιμή που της ανατέθηκε και συνεχίζει με την ανάθεση νέας τιμής στην μεταβλητή αυτή. Αν πάλι η ανάθεση αυτή δεν οδηγήσει σε κάποιου είδους σύγκρουσης συνεχίζεται η διαδικασία με την δημιουργία μιας νέας ανάθεσης τιμών σε μια άλλη μεταβλητή που ακόμα δεν έχει πάρει τιμή.

3.5.2 Λόγοι επέκτασης προτασιακών επιλυτών με χρήση ψευδοδυαδικών περιορισμών.

Ειδικές δομές έχουν υλοποιηθεί στα εργαλεία επίλυσης προβλημάτων (SAT) για να διαχειρίζονται ψευδοδυαδικούς περιορισμούς. Ο λόγος για τον οποίο δημιουργήθηκαν οι δομές αυτές ήταν για την καλύτερη αναπαράσταση των περιορισμών των προβλημάτων. Η απεικόνιση με ψευδοδυαδικούς περιορισμούς σε σχέση με την απεικόνιση στα προτασιακά προβλήματα, μπορούν να απεικονίσουν αρκετά προβλήματα βελτιστοποίησης με μια πιο φυσική έκφραση σε σύγκριση με τους διαζευκτικούς περιορισμούς στα προτασιακά προβλήματα. Η διαζευκτική μορφή περιορισμών στα προτασιακά (SAT) προβλήματα δεν μπορούσε να αναπαραστήσει σε ικανοποιητικό βαθμό περιορισμούς «counting constraints» δηλαδή αριθμητικούς περιορισμούς, σε αντίθεση με τα προβλήματα με μορφή ψευδοδυαδικών περιορισμών που έχουν την δυνατότητα αναπαράστασης τέτοιων περιορισμών σε ικανοποιητικό βαθμό αφού μπορούν να αναπαραστήσουν περιορισμούς σε μορφή εξισώσεων (γραμμική και μη γραμμική μορφή περιορισμών). Η απεικόνιση των ψευδοδυαδικών περιορισμών δεν διαφέρει κατά πολύ με την απεικόνιση των προτασιακών (SAT) προβλημάτων, οι ψευδοδυαδικοί περιορισμοί γενικεύουν το SAT θεωρώντας γραμμικές ανισότητες για τις δυαδικές μεταβλητές αντί για απλές προτάσεις (clauses) για τις δυαδικές μεταβλητές.[2]

3.6 Ορισμός Ψευδοδυαδικών περιορισμών

Η απεικόνιση ενός προβλήματος με ψευδοδυαδικούς περιορισμούς σε σχέση με την απεικόνιση προβλημάτων με διαζευκτικούς περιορισμούς, είναι πιο φυσική, απλή και πιο εκφραστική. Ένας ψευδοδυαδικός περιορισμός ορίζεται ως ένα πεπερασμένο σύνολο από δυαδικές μεταβλητές οι οποίες μπορούν να πάρουν μόνο δύο τιμές. Την τιμή 1 και η μεταβλητή αποτιμάται σε αληθής (true) ή την τιμή 0 που η μεταβλητή αποτιμάται σε ψευδής (false). Κάθε μια από αυτές τις δυαδικές μεταβλητές πολλαπλασιάζεται με ένα συντελεστή ο οποίος είναι ακέραιος αριθμός.

Ψευδοδυαδικά προβλήματα είναι γνωστά και ως προβλήματα ακέραιου γραμμικού 0-1 προγραμματισμού. Ένα τέτοιο μοντέλο γραμμικού προγραμματισμού περιλαμβάνει μεταβλητές απόφασης, την αντικειμενική συνάρτηση και τους περιορισμούς. Οι μεταβλητές απόφασης είναι οι μεταβλητές που ο επιλυτής θα ορίσει τιμές με τρόπο ώστε να βρεθεί η βέλτιστη λύση στο πρόβλημά μας. Η αντικειμενική συνάρτηση εκφράζει με ένα μαθηματικό τρόπο τον στόχο του προβλήματος μας, όπως για παράδειγμα την ελαχιστοποίηση κόστους ή την μεγιστοποίηση κέρδους ανάλογα με το πρόβλημα. Οι περιορισμοί είναι εξισώσεις ή ανισώσεις. Οι Ψευδοδυαδικοί περιορισμοί σε προβλήματα ακέραιου γραμμικού προγραμματισμού ακολουθούν μια συγκεκριμένη μορφή. Γίνεται κανονικοποίηση κάθε περιορισμού στην μορφή αυτή.

Γενικά σε ένα πρόβλημα Ακέραιου Προγραμματισμού οι περιορισμοί μπορούν να εκφράζονται γραμμικά ή μη γραμμικά.[5,16]

3.7 Γραμμικοί ψευδοδυαδικοί περιορισμοί

Η μορφή ενός ψευδοδυαδικού περιορισμού έχει την πιο κάτω μορφή:

$$\sum_{i=0}^{n} \mathrm{k_i}\, x_i \{=, <, >, \leq, \geq\}\ \mathrm{p}$$

όπου το k_i και p είναι ακέραιες τιμές, x_i είναι λεκτικό (literal) παίρνει την τιμή της μεταβλητής ή το συμπλήρωμα της. Σε κάθε περιορισμό μπορεί να ισχύει μόνο ένας από τους σχεσιακούς μαθηματικούς τελεστές $=, <, >, \leq, \geq$.

Οι σχεσιακοί αυτοί τελεστές έχουν τη γνωστή σε όλους μαθηματική τους έννοια και ορίζουν τη σχέση μεταξύ του δεξιού και του αριστερού μέλους του περιορισμού.
Ο περιορισμός θεωρείται ότι είναι ικανοποιήσιμος όταν ο δεξιά και ο αριστερά όρος πάρουν τιμές ακεραίων τέτοιες ώστε να ικανοποιούν τον σχεσιακό μαθηματικό τελεστή.[5,15]

Πιο κάτω φαίνονται παραδείγματα γραμμικών ψευδοδυαδικών περιορισμών

1. $5X_1 + 3X_2 + 2(\neg X_3) \geq 8$
2. $3(\neg X_1) + 2X_2 < 0$

3.8 Μη γραμμικοί Ψευδοδυαδικοί περιορισμοί

Ένας μη γραμμικός ψευδοδυαδικός περιορισμός έχει την μορφή:

$$\sum_{i=1}^{n} \kappa_i \left(\prod_j x_{i,j} \right) \{=, <, >, \leq, \geq\}\, p$$

Όπου το κi και το p παίρνουν ακέραια τιμή , x_i και $_j$ είναι λεκτικά (literals) παίρνουν την τιμή της μεταβλητής ή το συμπλήρωμά της. Σε κάθε περιορισμό μπορεί να ισχύει μόνο ένας από τους σχεσιακούς μαθηματικούς τελεστές $=, <, >, \leq, \geq$.
Οι σχεσιακοί αυτοί τελεστές έχουν τη γνωστή σε όλους μαθηματική τους έννοια και ορίζει τη σχέση μεταξύ του δεξιού και του αριστερού μέλους του περιορισμού.
Ο περιορισμός θεωρείται ότι είναι ικανοποιήσιμος όταν ο δεξιά και ο αριστερά όρος πάρουν ακέραιες τιμές τέτοιες ώστε να ικανοποιούν τον σχεσιακό μαθηματικό τελεστή.[5]

Πιο κάτω φαίνονται παραδείγματα μη γραμμικών ψευδοδυαδικών περιορισμών:

1. $X_1X_6 + 5X_2 + X_3 < 3$
2. $2(\neg X_1)X_2 + X_3 \geq -1$

3.9 Κανονικοποίηση ψευδοδυαδικών περιορισμών

Κάθε όρος που είναι γραμμένος σε συζευκτική κανονική μορφή (CNF) μπορεί εύκολα να μετατραπεί σε ψευδοδυαδικό περιορισμό. Μια πρόταση η οποία είναι ικανοποιήσιμη εάν είναι και σε κανονικοποιημένη μορφή, θα βοηθήσει σε μεγάλο βαθμό τον επιλυτή στο να εντοπίσει τη λύση. [8,16]
Πιο κάτω φαίνεται μια γενική κανονικοποιημένη μορφή για τους ψευδοδυαδικούς περιορισμούς.

$$\sum_{i=0}^{n} k_i \, x_i \geq p$$

Το k_i και το p παίρνουν ακέραια τιμή και το x_i είναι λεκτικό (literal) παίρνουν την τιμή της μεταβλητής ή το συμπλήρωμα της.
Πιο κάτω περιγράφονται βήματα για να μεταφραστεί οποιοσδήποτε περιορισμός με οποιοδήποτε τελεστή στην πιο πάνω κανονικοποιημένη μορφή των ψευδοδυαδικών περιορισμών.

<u>Ως προς τους τελεστές:</u>

Αν ο τελεστής του περιορισμού είναι ο = τότε δημιουργούνται δύο ίδιοι περιορισμοί, ο ένας με τον τελεστή ≥ και ο άλλος με τον τελεστή ≤ .
Αν ο τελεστής του περιορισμού είναι ο >, τότε ο τελεστής γίνεται ≥ και στο δεξί μέρος προστίθεται 1.
Αν ο τελεστής του περιορισμού είναι ο >, τότε ο τελεστής γίνεται ≤ και στο δεξί μέρος αφαιρείται 1.
Αν ο τελεστής του περιορισμού είναι ο ≤ τότε πολλαπλασιάζονται και τα δύο μέλη με 1 και αλλάζει ο τελεστής σε ≥.

Ως προς τους όρους:

Η αναπαράσταση μιας κανονικοποιημένης μορφής ενός ψευδοδυαδικού περιορισμού περιλαμβάνει μονό θετικούς συντελεστές που αυτό διευκολύνει σε μεγάλο βαθμό την αποτελεσματικότητα των αλγορίθμων επίλυσης προβλημάτων προτασιακής λογικής. Έτσι αλλάζουμε τον περιορισμό ώστε τα ki και p να μην είναι αρνητικά. Για κάθε μεταβλητή Xi το αντικαθιστούμε με 1- ┐Xi και κάθε μεταβλητή της μορφής ┐Xi με αρνητικό τελεστή, όπου Xi αντικαθιστάται από 1-Xi . Επίσης όλοι οι ακέραιοι μεταφέρονται στην δεξιά μεριά.[8]

3.10 Αλγόριθμος ψευδοδυαδικών επιλυτών

Ένας ψευδοδυαδικός επιλυτής χρησιμοποιεί έναν αλγόριθμο για να αναθέτει τιμές στις μεταβλητές με τέτοιο τρόπο ώστε να ικανοποιούνται οι περιορισμοί βρίσκοντας μια βέλτιστη λύση στο πρόβλημα. Ο αλγόριθμος αυτός αναθέτει στις διάφορες μεταβλητές τιμές 1 και 0. Αρχίζει με την ανάθεση μιας τιμής σε μια μεταβλητή γίνεται έλεγχος αν η τιμή που δόθηκε σε αυτή την μεταβλητή επηρεάζει τις υπόλοιπες μεταβλητές. Αν η τιμή που δόθηκε επηρεάζει τις υπόλοιπες μεταβλητές τότε γίνεται οπισθοδρόμηση (backtracking) δηλαδή επιστρέφει πίσω και αφαιρεί από την μεταβλητή την τιμή που τις ανατέθηκε και γίνεται ανάθεση καινούριας τιμής στην μεταβλητή. Αλλιώς αν η τιμή που δόθηκε δεν επηρεάζει τις υπόλοιπες μεταβλητές τότε συνεχίζει ανάθεση τιμών σε καινούρια μεταβλητή που δεν έχει πάρει τιμή ακόμα. Ο αλγόριθμος σταματά όταν βρει μια ανάθεση τιμών στις μεταβλητές που να ικανοποιεί όλους τους περιορισμούς.

3.11 Ψευδοδυαδικοί επιλυτές σε προβλήματα αποφασισιμότητας και προβλήματα βελτιστοποίησης

Υπάρχουν δύο είδη προβλημάτων ικανοποίησης ψευδοδυαδικών περιορισμών. Η μια κατηγορία αναφέρεται ως ψευδοδυαδικά προβλήματα επίλυσης (Pseudo-Boolean Solving) η οποία επιλύει προβλήματα αποφασισιμότητας. Ως απάντηση στα προβλήματα αυτά επιστρέφεται κατά πόσο είναι ικανοποιήσιμο. Πιο συγκεκριμένα αν υπάρχει έστω μια ανάθεση τιμών στις μεταβλητές τέτοια ώστε να ικανοποιούνται οι

περιορισμοί, ο επιλυτής θα μας επιστρέψει ικανοποιήσιμο (SATISFY), αλλιώς θα επιστρέψει μη ικανοποιήσιμο (UNSATISFY). Στα προβλήματα αυτά δεν μας εγγυάται ο επιλυτής ότι θα επιστρέψει τη λύση στο πρόβλημα, δηλαδή την ανάθεση τιμών στις μεταβλητές. Η μόνη απάντηση που δίνεται συνήθως είναι αν μπορεί να επιλυθεί ή όχι το πρόβλημα.

Η δεύτερη κατηγορία επιλύει προβλήματα βελτιστοποίησης (Pseudo-Boolean Optimization). Ως απάντηση στα προβλήματα αυτά επιστρέφονται δύο αποτελέσματα. Το ένα αποτέλεσμα είναι κατά πόσο είναι ικανοποιήσιμο ή μη ικανοποιήσιμο ένα πρόβλημα. Το δεύτερο αποτέλεσμα είναι αν είναι ικανοποιήσιμο το πρόβλημα επιστρέφεται η βέλτιστη λύση για τη συγκεκριμένη φόρμουλα περιορισμών, δηλαδή επιστρέφεται ο καλύτερος συνδυασμός αναθέσεων τιμών στις μεταβλητές που να ικανοποιούν τους περιορισμούς και τις συναρτήσεις βελτιστοποίησης και να δίνουν την βέλτιστη λύση για το πρόβλημα.

Οι διάφοροι επιλυτές ψευδοδυαδικών περιορισμών υποστηρίζουν αρκετές συναρτήσεις βελτιστοποίησης. Κύριες συναρτήσεις βελτιστοποιήσεις που υποστηρίζουν είναι η συνάρτηση min και η συνάρτηση max.

Η βασική διαφορά των δύο κατηγοριών είναι ότι στην πρώτη κατηγορία δεν σου εγγυάται ο επιλυτής ότι θα σου επιστρέψει την λύση στο πρόβλημα ενώ σε πρόβλημα βελτιστοποίησης θα πάρεις ως απάντηση αν έχει λύση και τη βέλτιστη πιθανή λύση.

Στην παρούσα διπλωματική γίνεται χρήση της συναρτήσεις βελτιστοποίησης min σε συνδυασμό με πολλούς ψευδοδυαδικούς και SAT επιλυτές, όπως στο minisat+,bsolo_pb10, npSolver.

Κεφάλαιο 4

Μοντελοποίηση Προβλήματος Συνοχής με βάση τους επιλυτές

4.1 Επιλυτής MiniZinc……..……..26

4.2 Επιλυτής Gecode……...28

4.2.1 Επιλυτής Gecode με χρήση search combinator..……29

4.3 Επιλυτής Minisat +……...……32

4.4 Επιλυτής Bsolo_pb10……..……..…33

4.4.1 Αλγόριθμος Κλάδου και Φράγματος για Ακέραιο Προγραμματισμό……34

4.4.2 Αλγόριθμος Bsolo…….....35

4.4.3 Μέθοδοι βελτιστοποίησης μέσω παραμέτρων στον Bsolo……36

4.4.3.1 Ανεξαρτησία των περιορισμών και Χαλάρωση Γραμμικού Προγραμματισμού…….37

4.4.3.2 Χρήση Στρατηγικής Εκμάθηση……..38

4.5 Επιλυτής npSolver ……38

4.6 Μοντελοποίηση Προβλήματος Συνοχής στους Επιλυτές…….....40

4.6.1 Μοντελοποίηση στον επιλυτή MiniZinc……..…..…41

4.6.2 Μοντελοποίηση στους Minisat+ και Bsolo……..…….49

4.6.3 Μοντελοποίηση στον επιλυτή npSolver…….…....55

4.6.4 Μοντελοποίηση στον επιλυτή Gecode…….....56

4.6.4.1 Με χρήση search combinators…….56

4.1 Επιλυτής MiniZinc

Αρχικά το Πρόβλημα της Συνοχής μοντελοποιήθηκε ως πρόβλημα ικανοποίησης περιορισμών με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών στη γλώσσα MiniZinc στον επιλυτή MiniZinc.
Η γλώσσα MiniZinc είναι μεσαίου επιπέδου γλώσσα η οποία σχεδιάστηκε για την επίλυση προβλημάτων ικανοποίησης περιορισμών. Χρησιμοποιείται για προβλήματα αποφασισιμότητας και προβλήματα βελτιστοποίησης για ακέραιους και πραγματικούς αριθμούς. Είναι ένα υποσύνολο της γλώσσας Zinc και χρησιμοποιεί την FlatZinc για την επίλυση των προβλημάτων. Αυτό επιτυγχάνεται με την μετατροπή του μοντέλου της MiniZinc σε μοντέλο FlatZinc.
Ως είσοδο στον επιλυτή θα δοθεί ένα πρόγραμμα σε MiniZinc το οποίο αποτελείται από ένα ή δύο αρχεία. Υπάρχει δηλαδή το αρχείο με την επέκταση ".mzn" το οποίο περιέχει το πρόβλημα μοντελοποιημένο σε μορφή προβλήματος ικανοποίησης περιορισμών και μπορεί να υπάρχει και ένα δεύτερο αρχείο δεδομένων ".dzn" το οποίο να περιέχει τα δεδομένα του προγράμματος.

Ένα μοντέλο MiniZinc αποτελείται από την δήλωση των μεταβλητών, τους ορισμούς των περιορισμών του προβλήματος καθώς επίσης αν είναι πρόβλημα βελτιστοποίησης απαιτείται η δήλωση της αντικειμενικής συνάρτησης βελτιστοποίησης .

Βασική Δομή ενός μοντέλου MiniZinc:
Υπάρχουν αρκετά είδη αντικειμένων που μπορούν να ενσωματωθούν σε ένα μοντέλο MiniZinc. Κάποια από αυτά είναι τα εξής: [7]

1) Μπορεί να γίνεται χρήση περιεχομένου από άλλο αρχείο σε αυτό το μοντέλο. Αυτό επιτυγχάνεται με την πιο κάτω μορφή:

include ⟨*filename*⟩;

Όπου το *filename είναι ένα string.*

2) Δηλώσεις νέων μεταβλητών. Αυτές οι μεταβλητές ονομάζονται καθολικές μεταβλητές (global variables) και μπορούν να δηλωθούν οπουδήποτε στο μοντέλο. Υπάρχουν δύο ειδών μεταβλητές. Αυτές που ορίζονται με σταθερή τιμή στο μοντέλο ή στο αρχείο δεδομένων και ονομάζονται παράμετροι (parameters). Και αυτές που παίρνουν τιμές όταν το μοντέλο επιλυθεί και ονομάζονται μεταβλητές απόφασεις (decision variables) . Λέμε πως οι parameters παίρνουν σταθερές τιμές (fixed) ενώ οι μεταβλητές απόφασης (decision variables) παίρνουν αόριστες τιμές (unfixed) . Η δήλωση αυτή γίνεται με την πιο κάτω μορφή:

⟨*type inst expr*⟩: ⟨*variable*⟩ [= ⟨*expression*⟩];

Όπου *inst* για τις παραμέτρους δηλώνεται με τη λέξη par ενώ για τις μεταβλητές απόφασεις (decision variables) χρησιμοποιείται η λέξη var. Αν δεν υπάρχει ρητή δήλωση τότε το μοντέλο αντιλαμβάνεται ότι η μεταβλητή αυτή είναι par δηλαδή παράμετρος άρα σταθερή η τιμή της. Όσο αφορά το *type* μπορεί να πάρει ένα από τους βασικούς τύπους int ή float range, πίνακα ή σύνολο μπορεί να πάρει ένα από τους βασικούς τύπους float, int, string, bool, ann αν μιλάμε για παραμέτρους, και float, int, bool αν μιλάμε για μεταβλητές απόφασης.
Παράδειγμα: int nc=3;
var 1..5=wa;
Η nc είναι μια μεταβλητή parameter ενώ η wa είναι μια μεταβλητή απόφασης.

3) Ανάθεση τιμών σε μεταβλητή γίνεται με την πιο κάτω μορφή:

⟨*variable*⟩ = ⟨*expression*⟩;
Παράδειγμα : nv=5;
Ανατίθεται στην μεταβλητή nv η τιμή 5.

4) Σημαντικό μέρος του μοντέλου είναι οι δηλώσεις των περιορισμών του προβλήματος. Ο τρόπος δήλωσης ενός περιορισμού επιτυγχάνεται με τον πιο κάτω τρόπο:
constraint ⟨*Boolean expression*⟩;
Υπάρχουν πάρα πολύ σύνθετοι περιορισμοί [7]

5) Επίσης απαραίτητο σημείο στο μοντέλο είναι το αντικείμενο επίλυσης, πιο είδος λύσης να ψάχνει ο επιλυτής. Υπάρχουν τρία είδη επίλυσης

- Solve satisfy;
- solve maximize ⟨arithmetic expression⟩;
- solve minimize ⟨arithmetic expression⟩;

Είναι απαραίτητο να υπάρχει ένα από τα πιο πάνω στο μοντέλο για να είναι ορθό το μοντέλο.

6) Ένα άλλο στοιχείο στο μοντέλο που διευκολύνει την παρουσίαση της λύσης του μοντέλου είναι ο τρόπος με τον οποίο θα παρουσιάζεται η λύση στην οθόνη μας. Αυτό επιτυγχάνεται με την πιο κάτω μορφή:
output [⟨*string expression*⟩, ・ ・ ・ , ⟨*string expression*⟩];

Ο τρόπος με τον οποίο καλούμε τον επιλυτή MinZinc στο να αναζητήσει λύση είναι:
$ minizinc όνομα_μοντέλου.mzn όνομα_αρχείου_δεδομένων.dzn

4.2 Ο επιλυτής Gecode

Ο επιλυτής Gecode είναι μια βιβλιοθήκη λογισμικού για την επίλυση προβλημάτων ικανοποίησης περιορισμών. Είναι προγραμματισμένος σε C++ και διανέμεται ως ελεύθερο λογισμικό. Ο Gecode έχει συνδέσεις σε πολλές γλώσσες προγραμματισμού όπως η Prolog , Python και Ruby, και μια διεπαφή για τη γλώσσα μοντελοποίησης AMPL. Η πρώτη έκδοση του Gecode έγινε το Δεκέμβριο του 2005. Από τότε ο Gecode έχει γίνει γνωστός για συστήματα προγραμματισμού με ικανοποίηση περιορισμούς . Οι λόγοι για τους οποίους έγινε δημοφιλής είναι ότι τρέχει γρήγορα, είναι επεκτάσιμος, δωρεάν και ανοιχτού κώδικα καθώς και για το λόγο ότι είναι γραμμένος σε μια δημοφιλή και διαδεδομένη γλώσσα. Εκτός του ότι είναι πολύ χρήσιμος από μόνος του, η επεκτασιμότητα και η αδειοδότηση καθιστά τον Gecode ιδιαίτερα κατάλληλο για χρήση σε πολλά άλλα έργα.

Η μοντελοποίηση στο Gecode στην παρούσα διπλωματική γίνεται μέσω δύο τρόπων. Ο πρώτος τρόπος είναι το μοντέλο το οποίο έτρεχε στον επιλυτή MiniZinc, το ίδιο μοντέλο αναλλοίωτο τρέχει και στον Gecode, ενώ ο δεύτερος τρόπος είναι με ενσωμάτωση συνδυαστών αναζήτησης δηλαδή ενσωμάτωση συνδυασμών ευρετικών στον τρόπο αναζήτησης της λύσης στο μοντέλο.[10]

4.2.1 Επιλυτής Gecode με χρήση συνδυαστών αναζήτησης

Ο επιλυτής Gecode έχει ακόμα μια ιδιαιτερότητα. Έχει επεκταθεί ο διερμηνέας Gecode Flatzinc ώστε να υποστηρίζει συνδυαστές αναζήτησης (search combinators), δηλαδή μια νέα ισχυρή γλώσσα μοντελοποίησης για αναζήτηση μέσω ευρετικών. Έχει την ευελιξία στο να ενσωματώνει ευρετικά στον τρόπο αναζήτησης της λύσης. Αυτό γίνεται με τη χρήση διαφόρων συνδυαστών αναζήτησης. Η μόνη αλλαγή που γίνεται στο μοντέλο είναι στο αντικείμενο solve. Στο solve ενσωματώνονται συνδυαστές αναζήτησης λύσης. Υπάρχουν πολλοί υλοποιημένοι συνδυαστές στον επιλυτή, δίνεται όμως η δυνατότητα να δημιουργήσεις και τους δικού σου.
Η ευρετική αναζήτηση κάνει συχνά τη διαφορά στο να επιλύει αποτελεσματικά και με επιτυχία ένα πρόβλημα ικανοποίησης περιορισμών. Ένα ευρετικό κάνει τον αλγόριθμο αναζήτησης αποδοτικό για διάφορους λόγους όπως για παράδειγμα η ενσωμάτωση της γνώσης, η τυχαιοποίηση, η αποφυγή μεγάλων ουρών. Επιπρόσθετα η χρήση συνδυαστών αναζήτησης είναι πιο παραγωγική από την προσφυγή σε μια γενικής χρήσης γλώσσα.
Το να προσθέτεις όμως ένα περισσότερο συνδυαστή αναζήτησης είναι εύκολο, όμως όπως θα δούμε και στα πειράματα που έγιναν μπορεί να επηρεαστεί ο χρόνος αναζήτησης βέλτιστης λύσης ως προς το χειρότερο.

Στον επιλυτή αυτό μπορείς να ορίσεις τους δικούς σου συνδυαστές αναζήτησης έχοντας υπόψη σου τα εξής:

Οι συνδυαστές ορίζονται με την λέξη «annotation» ακολουθούμενο με το όνομά του και τις παραμέτρους που παίρνουν (αν παίρνουν παραμέτρους). Στο μοντέλο απαιτείται εκτός από τη δήλωση του πρωτότυπου του συνδυαστή, να φαίνεται και η υλοποίηση του. Κάποια παραδείγματα συνδυαστών είναι τα ακόλουθα:

- annotation prune;

Ο συνδυαστής αυτός είναι αχρείαστος από μόνος του, όμως σε συνδυασμό με άλλους συνδυαστές (annotations) είναι πολύ χρήσιμος. Ο συνδυαστής αυτός κόβει το δέντρο αναζήτησης κάτω από τον τρέχοντα κόμβο.

- annotation ifthenelse(var bool: b, ann:s1, ann:s2);

Όπου ann μπορούν να δοθούν ως παράμετροι τα διάφορα annotation που υποστηρίζονται από τον επιλυτή Gecode, είτε διάφορα annotation που δηλώνονται και υλοποιούνται από τον χρήστη.
Ο συνδυαστής ifthenelse εστιάζει σε μια μεταβλητή έκφρασης b. Έως ότου η μεταβλητή b είναι αληθής στον τρέχοντα κόμβο εκτελείται ο συνδυαστής s1. Αν η b γίνει ψευδής, τότε ο συνδυαστής s2 εκτελείται σε όλο το υπόλοιπο υποδέντρο κάτω από τον τρέχοντα κόμβο.

- annotation portfolio(array[int] of ann: s);

Ο συνδυαστής portfolio παίρνει ένα σύνολο από συνδυαστές μπορούν να δοθούν ως παράμετροι τα διάφορα annotation που υποστηρίζονται από τον επιλυτή Gecode, είτε διάφορα annotation που δηλώνονται και υλοποιούνται από τον χρήστη και εκτελεί τον πρώτο συνδυαστή s1 στο δέντρο. Αν ο συνδυαστής s1 εξερεύνησε όλο το δέντρο χωρίς να καλεστεί ο συνδυαστής prune ο οποίος κόβει το δέντρο αναζήτησης κάτω από τον τρέχοντα κόμβο τότε η αναζήτηση τελειώνει αλλιώς τώρα ο συνδυαστής συνεχίζει με την εκτέλεση των συνδυαστών [s2…sn], δηλαδή τώρα εκτελείται ο συνδυαστής s2 στο δέντρο. Αν ο συνδυαστής s2 εξερεύνησε όλο το δέντρο χωρίς να καλεστεί ο συνδυαστής prune ο οποίος κόβει το δέντρο αναζήτησης κάτω από τον τρέχοντα κόμβο τότε η αναζήτηση τελειώνει αλλιώς τώρα ο συνδυαστής συνεχίζει με την εκτέλεση του επόμενου συνδυαστή στο δέντρο αναζήτησης δηλαδή του s3. Η διαδικασία συνεχίζεται αν δεν διακοπεί μέχρι και τον συνδυαστή sn.

Επιπρόσθετα τα δυο βασικά ευρετικά αναζήτησης επιλογής των δυνατών τιμών σε μια μεταβλητή απόφασης που υποστηρίζονται είναι τα εξής :
annotation int_search(array[int] of var int: x, ann:varsel,ann:valsel);
annotation bool_search(array[int] of var bool: x, ann:varsel,ann:valsel);

Για ακέραιες μεταβλητές όπου ann μπορούν να καλεστούν ως παράμετροι πολλά annotations. Μερικά από αυτά είναι:

- annotation min_dom_size;

Αναθέτει πρώτα τιμές στις μεταβλητές αυτές με το μικρότερο σε μέγεθος πεδίο ορισμού.

- annotation max_dom_size;

Αναθέτει πρώτα τιμές στις μεταβλητές αυτές με το μεγαλύτερο σε μέγεθος πεδίο ορισμού.

Για Boolean μεταβλητές όπου ann μπορούν να καλεστούν ως παράμετροι τα πιο κάτω annotations:
input_order,max_weighted_degree, assign_lb, assign_ub .

Το μοντέλο ακολουθεί την μορφή αρχείου σε γλώσσα MiniZinc με την διαφορά ότι στο σημείο καλέσματος του επιλυτή στο να βρει την βέλτιστη λύση καλούνται οι συνδυαστές αναζήτησης:

Solve:: print(x,bab(x1,int_search(y, random_order,assign_lb))) **satisfy**;

Αν το μοντέλο χρησιμοποιεί συνδυαστές αναζήτησης τότε ο επιλυτής καλείται αλλιώς από ότι θα καλεστεί αν απλά θα τρέξει ένα μοντέλο το οποίο έτρεχε στον MiniZinc επιλυτή.

Ο Gecode χωρίς χρήση συνδυαστών αναζήτησης καλείται με τον εξής τρόπο:
$ mzn2fzn αρχείο.mzn αρχείο_δεδομένων.dzn
$ fzn-gecode αρχείο.fzn

Ο Gecode με χρήση συνδυαστών αναζήτησης καλείται με τον εξής τρόπο:
$ scmzn2mzn -I . -I $ (μονοπάτι που είναι εγκατεστημένος ο MiniZinc)/lib/minizinc/std αρχείο.mzn αρχείο_δεδομένων.dzn
$ mzn2fzn --ignore-stdlib αρχείο.pr.mzn
$ fz_sc αρχείο.pr.fzn

4.3 Ο επιλυτής Minisat+

Ο Minisat+ εντάσσεται στα εργαλεία επίλυσης προβλημάτων ικανοποίησης προτασιακής λογικής (SAT problems). Τα εργαλεία αυτά επεκτάθηκαν με δομές για χειρισμό και άλλων ειδών περιορισμών όπως ψευδοδυαδικούς περιορισμούς. [8]
Στην εκπόνηση της διπλωματικής αυτής θα μελετηθεί η επίλυση ψευδοδυαδικών προβλημάτων μέσω του επιλυτή αυτού. Για την επίτευξη χρειάζεται να γίνει μεταφράσει του προβλήματος σε γραμμικούς ψευδοδυαδικούς περιορισμούς που περιγράφτηκαν στο κεφάλαιο 3. Το εργαλείο επίλυσης αυτό υποστηρίζει επίσης συναρτήσεις βελτιστοποίησης που δίνει σαν αποτέλεσμα την βέλτιστη λύση στο πρόβλημα. Για το εργαλείο αυτό οι πιο γνωστές συναρτήσεις βελτιστοποίησης είναι η συνάρτηση min η οποία προσπαθεί να ελαχιστοποιήσει την τιμή της συνάρτησης ικανοποιώντας όλους τους περιορισμούς και η συνάρτηση max η οποία προσπαθεί να μεγιστοποιήσει την τιμή της συνάρτησης ικανοποιώντας παράλληλα όλους τους περιορισμούς.

<u>Μορφή αρχείου:</u>

Ο επιλυτής Minisat+ απαιτεί μια συγκεκριμένη μορφή αρχείου ως είσοδο. Ένα παράδειγμα της μορφής απεικονίζεται στο Σχήμα 4.2.

Στην πρώτη γραμμή βρίσκεται η αντικειμενική συνάρτηση βελτιστοποίησης του προβλήματος και απεικονίζεται ως εξής : min:2*xx01+1*xx02+4*xx03;

Ακολούθως περιλαμβάνει τους περιορισμούς σε μορφή που δέχεται ο επιλυτής.
-1*x01-1*x12+1*xx01>=-1;
-1*x02-1*x11+1*xx01>=-1;
-1*x01-1*x22+1*xx02>=-1;

Κάθε μεταβλητή αποτελείται από το πρόσημο της ακολουθούμενο από το 1* και το όνομα της μεταβλητής. Στο δεξιό μέρος της ανίσωσης βρίσκεται ένας ακέραιος αριθμός τον οποίο προκύπτει από την μετατροπή περιορισμού από προτασιακής μορφής σε γραμμική ψευδοδυαδική μορφή.
Ο τρόπος με τον οποίο καλείται ο επιλυτής Minisat+ είναι ο εξής:

$ minisat+ όνομα_αρχείου

4.4 Ο επιλυτής Bsolo_pb10

Ο επιλυτής bsolo είναι εργαλείο το οποίο χρησιμοποιείται για την επίλυση ψευδοδυαδικών προβλημάτων. Ο αλγόριθμος ο οποίος χρησιμοποιεί ο bsolo περιλαμβάνει βασικά χαρακτηριστικά από τον αλγόριθμο κλάδου και φράγματος (Branch and Bound). Πιο συγκεκριμένα χρησιμοποιείται το χαμηλότερο κάτω φράγμα. Περιλαμβάνει στοιχεία και από τους αλγορίθμους επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής όπως τις τεχνικές κλαδέματος αναζήτησης συμπεριλαμβανομένου και της μη χρονολογικής οπισθοδρόμησης στο δέντρο αναζήτησης καθώς επίσης γίνεται χρήση των μηχανισμών μάθησης που είναι βασισμένοι σε συγκρούσεις. [3,4]

Μορφή αρχείου:
Ο επιλυτής bsolo απαιτεί μια συγκεκριμένη μορφή αρχείου ως είσοδο. Ένα παράδειγμα της μορφής απεικονίζεται στο Σχήμα 4.3

Στην πρώτη γραμμή βρίσκεται η αντικειμενική συνάρτηση βελτιστοποίησης του προβλήματος και απεικονίζεται ως εξής : min:2 xx01+1 xx02+4 xx03;

Ακολούθως περιλαμβάνει τους περιορισμούς σε μορφή που δέχεται ο επιλυτής.
-1 x01-1 x12+1 xx01>=-1;
-1 x02-1 x11+1 xx01>=-1;
-1 x01-1 x22+1 xx02>=-1;

Κάθε μεταβλητή αποτελείται από το πρόσημο της ακολουθούμενο από το 1, ακολουθεί κενό που συμβολίζει το πολλαπλασιασμό και το όνομα της μεταβλητής Στο δεξιό μέρος της ανίσωσης βρίσκεται ένας ακέραιος αριθμός τον οποίο προκύπτει από την μετατροπή περιορισμού από προτασιακής μορφής σε γραμμική ψευδοδυαδική μορφή.
Ο τρόπος καλέσματος του επιλυτή bsolo_pb10 είναι:
$ bsolo_pb10 <επιλογές παραμέτρων> όνομα_αρχείου
Οι επιλογές παραμέτρων είναι προαιρετικές.

4.4.1 Αλγόριθμος Κλάδου και Φράγματος για Ακέραιο Προγραμματισμό

Οι τεχνικές του αλγορίθμου κλάδου και φράγματος έχουν εφαρμογή σε διάφορα προβλήματα Επιχειρησιακής Έρευνας αλλά και σε πολλά προβλήματα ακέραιου προγραμματισμού. Η βασική ιδέα του αλγόριθμου είναι η τεχνική του, «διαίρει και βασίλευε». Η τεχνική αυτή στόχο έχει να απλοποιήσει το πρόβλημα σε μικρότερα υποπροβλήματα. Πιο γενικά η μέθοδος αυτή υπολογίζει αρχικά ένα άνω όριο που αντιστοιχεί στην τιμή της αντικειμενικής συνάρτησης, δηλαδή την τιμή της συνάρτησης του κόστους που προκύπτει επιλύοντας το αρχικό πρόβλημα . Η τιμή αυτή ενημερώνεται όταν κατά την αναζήτηση βρεθεί μικρότερη τιμή της συνάρτησης του κόστους για τα προβλήματα. Γίνεται αναζήτηση σε κάθε κόμβο στο δέντρο αναζήτησης, ώστε να εντοπιστεί το κατώτερο όριο για το κόστος της συνάρτησης με βάση τις μέχρι τώρα αναθέσεις τιμών. Αν σε κάποιο σημείο βρούμε μια λύση όπου η τιμή προκύπτει μεγαλύτερη ή ίση από την τιμή του άνω ορίου τότε κατά τη διάρκεια της αναζήτησης καλύτερης λύσης μπορούμε να αγνοηθεί (κλαδευτεί) ο συγκεκριμένος κόμβος καθώς δεν μπορεί να βελτιωθεί με την τρέχουσα ανάθεση τιμών.

Στην παρούσα διπλωματική εργασία έγινε απεικόνιση του προβλήματος σε μορφή 0/1 ακέραιου προγραμματισμού και είναι σημαντικό να γίνει κατανοητός ο αλγόριθμος αυτός αφού χρησιμοποιείται από τον επιλυτή BSOLO.

4.4.2 Αλγόριθμος Bsolo

Ως πρώτο βήμα γίνεται ο υπολογισμός και η αρχικοποίηση του ανώτατου ορίου με την μέγιστη δυνατή τιμή.

Σε δεύτερο βήμα γίνεται έλεγχος αν παράγεται κάποια σύγκρουση με την τρέχουσα κατάσταση, εφαρμόζοντας μοναδιαία μετάδοση (unit propagation), μέσω της συνάρτησης consistent_state. Αν εντοπιστεί κάποια σύγκρουση τότε καλούνται κάποιοι από τους μηχανισμούς μάθησης που είναι βασισμένοι στις συγκρούσεις. Καλείται η διαδικασία ανάλυσης συγκρούσεων , η διαδικασία μάθησης των προτάσεων και αν χρειάζεται καλείται και η διαδικασία της οπισθοδρόμησης.

Στο τρίτο βήμα αν εντοπιστεί μια λύση στο πρόβλημα, τότε το άνω όριο ενημερώνεται με την νέα τιμή, επίσης με την διαδικασία της οπισθοδρόμησης αν εντοπιστεί καλύτερη λύση με χαμηλότερο κόστος τότε και πάλι ενημερώνεται η τιμή του άνω ορίου.

Στο τέταρτο και σημαντικό βήμα του αλγορίθμου γίνεται μια εκτίμηση του κατώτατου ορίου μέσω της τρέχουσας ανάθεσης των τιμών στις μεταβλητές. Αν η τιμή του κάτω ορίου προκύπτει μεγαλύτερη ή ίση από την τιμή του τρέχον άνω ορίου, τότε παρατηρείται ένα δεύτερο είδος σύγκρουσης και καλείται η διαδικασία ανάλυσης της σύγκρουσης ώστε να εντοπίσει σε πιο σημείο στο δέντρο αναζήτησης θα πρέπει να γίνει η διαδικασία της οπισθοδρόμησης και εκτελείται ξανά το δεύτερο βήμα.

Όπως αναφέρεται και πιο πάνω υπάρχουν δύο ειδών συγκρούσεις. Μέσα από την διαδικασία ανάλυσης των συγκρούσεων υπάρχει η δυνατότητα μάθησης ενός καινούριου ψευδοδυαδικού περιορισμού. Στα προβλήματα ψευδοδυαδικών περιορισμών, όπως και το πρόβλημα που θέτεται στην παρούσα διπλωματική, η διαδικασία αυτή είναι εξαιρετικά αποτελεσματική στα προβλήματα αυτά αφού επιτρέπει τη μη χρονολογική οπισθοδρόμηση. Επίσης η μάθηση των προτάσεων εξασφαλίζει ότι στη πορεία κατά την διαδικασία αναζήτησης μπορεί να αποφευχθεί η επίσκεψη σε αχρείαστα μέρη στο δέντρο αναζήτησης. Ο ολοκληρωμένος αλγόριθμος φαίνεται στο σχήμα 4.1 [3]

```
int bsolo(φ) {
  ub = ∑ c_j + 1;
  while (TRUE) {
    decide();
    if (!consistent_state())
      return ub;
    while (Estimate_LB() ⩾ ub) {
      Issue_LB_based_conflict();
      if (!consistent_state())
        return ub;
    }
  }
}
int consistent_state() {
  while (Deduce() == CONFLICT)
    if (Diagnose() == CONFLICT)
      return FALSE;
  if (Solution_found())
    Update_ub();
  return TRUE;
}
```

Σχήμα 4.1: Αλγόριθμος bsolo

4.4.3 Μέθοδοι Βελτιστοποίησης μέσω παραμέτρων στο Bsolo

Ο επιλυτής bsolo καλείται με κάποιες παραμέτρους (options) οι οποίες τον κάνουν να βρίσκει την βέλτιστη λύση πολύ πιο γρήγορα.

Usage : bsolo<options> file

Επιλογές (options):

-bn– χρήση κατώτατου ορίου (lower bound usage) (το n μπορέι να πάρει τις τιμές 0,1,2,3) (προεπιλογή n=1)

0 - Δεν χρησιμοποιείται κατώτατο όριο

1 – Μικτή εκδοχή χρήσης κατώτερου ορίου

2 – Δυναμική χρήση του LPR (linear programming relaxation) κάτω φράγματος

3 - Δυναμική χρήση ανίχνευσης κάτω φράγματος

4 – Δυναμική χρήση του MIS (Maximum Independent set of constraints) κάτω φράγματος

-ln –εκμάθηση χρήσης στρατηγικής (1,2,3) (προεπιλογή n=1)

1 – Μάθηση όρου

2 – Μάθηση περιορισμού πληθικότητας (Cardinality Constraint Learning)

3 – Γενικός ψευδοδυαδικός περιορισμός (General Pseudo-Boolean Learning)

Στην εκπόνηση της διπλωματικής χρησιμοποίησα τις παραμέτρους –bn – ln και μέσω πειραμάτων βρήκα τον καλύτερο συνδυασμό ώστε να επιτυγχάνεται γρήγορα και αποτελεσματικά η βέλτιστη λύση για τον Πρόβλημα της Συνοχής.

Η χρήση του κατώτερου ορίου παίζει σημαντικό ρόλο στην αποδοτικότητα του αλγορίθμου. Ο λόγος που συμβαίνει αυτό είναι επειδή αν σε κάποιο σημείο βρούμε μια λύση όπου η εκτίμηση της τιμής του κατώτατου ορίου προκύπτει μεγαλύτερη ή ίση από την τιμή του άνω ορίου τότε κατά τη διάρκεια της αναζήτησης καλύτερης λύσης μπορούμε να αγνοήσουμε (κλαδέψουμε) το συγκεκριμένο κόμβο καθώς δεν μπορεί να βελτιωθεί με την τρέχουσα ανάθεση τιμών. Υπάρχουν πάρα πολλές τεχνικές οι οποίες μπορούν να χρησιμοποιηθούν για την εκτίμηση της τιμής του κατώτατου ορίου και πολλές από αυτές χρησιμοποιεί και ο αλγόριθμος BSOLO. Μερικές από αυτές είναι η μεγίστη ανεξαρτησία των περιορισμών καθώς και η χαλάρωση του γραμμικού προγραμματισμού.

4.4.3.1 Ανεξαρτησία των περιορισμών και Χαλάρωση του Γραμμικού Προγραμματισμού

Το μέγιστο ανεξάρτητο σύνολο περιορισμών (MIS) είναι μια μέθοδος για τον υπολογισμό χαμηλότερου ορίου για την τιμή της συνάρτησης κόστους που βασίζεται σε ένα ανεξάρτητο σύνολο περιορισμών. Γίνεται μια άπληστη διαδικασία για εξεύρεση μιας σειράς ασύνδετων, ανεξάρτητων περιορισμών, δηλαδή περιορισμών που δεν έχουν κοινά λεκτικά (literals) μεταξύ τους.

Το MIS κάτω φράγμα προτάθηκε αρχικά ως μια ειδική περίπτωση, όπου όλοι οι περιορισμοί είναι προτασιακοί όροι (clauses). Με αυτήν την προσέγγιση, θα συμπεριλαμβάνονται στο μέγιστο ανεξάρτητο σύνολο (MIS) οι όροι (clauses) που μεγιστοποιούν τη σχέση μεταξύ του βάρους με το ελάχιστο κόστος. Το ελάχιστο κόστος για την ικανοποίηση του ανεξάρτητου συνόλου των περιορισμών αποτελεί το κατώτατο όριο για την βέλτιστη λύση. [1,3, 19]

Το πρόβλημα του γραμμικού προγραμματισμού που προκύπτει από την αφαίρεση της απαίτησης ότι οι μεταβλητές πρέπει να είναι ακέραιες δηλώνεται ως "γραμμική χαλάρωση" (linear relaxation) του ακέραιου γραμμικού προβλήματος. [11]

4.4.3.2 Χρήση Στρατηγικής Εκμάθησης

Ψευδοδυαδικοί περιορισμοί μπορούν να χαρακτηριστούν ως προτασιακοί όροι (clauses), ως περιορισμοί πληθικότητας ή ως γενικοί ψευδοδυαδικοί περιορισμοί. Ένας περιορισμός όπου όλοι οι συντελεστές των λεκτικών είναι 1 και η δεξιά πλευρά του περιορισμού είναι επίσης 1 λέγεται ότι είναι μια προτασιακή πρόταση (clause). Σε έναν περιορισμό πληθικότητας , μπορεί όλοι οι συντελεστές των λεκτικών (literals) να είναι 1 αλλά το δεξιό μέρος του περιορισμού να είναι μεγαλύτερο του 1.
Όλα τα άλλα είδη περιορισμών εντάσσονται στους γενικούς ψευδοδυαδικούς περιορισμούς .
Μέσα από την διαδικασία ανάλυσης των συγκρούσεων υπάρχει η δυνατότητα μάθησης ενός καινούριου ψευδοδυαδικού περιορισμού. Η μάθηση των προτάσεων εξασφαλίζει ότι στη πορεία κατά την διαδικασία αναζήτησης μπορεί να αποφευχθεί η επίσκεψη σε αχρείαστα μέρη στο δέντρο αναζήτησης. [3]

4.5 Ο επιλυτής npSolver

Ο ψευδοδυαδικός επιλυτής npSolver κωδικοποιεί ψευδοδυαδικούς περιορισμούς σε SAT και λύνει τα προβλήματα βελτιστοποίησης καλώντας έναν επιλυτή SAT επαναληπτικά. Τα προβλήματα βελτιστοποίησης αντιμετωπίζονται από έναν άπληστο μηχανισμό κατώτατου ορίου. Βασίζεται σε μια μετάφραση ψευδοδυαδικών περιορισμών σε SAT μορφή. Με βάση μια εφαρμογή το εργαλείο αυτό μπορεί να χρησιμοποιήσει διάφορες κωδικοποιήσεις για να επιτευχθεί μια καλή φόρμουλα SAT. Με την μετατροπή στη φόρμουλα αυτή, ένας επιλυτής SAT καλείται να λύσει το πρόβλημα. Για τις περιπτώσεις βελτιστοποίησης, ο επιλυτής καλείται πολλαπλές φορές. Ο npSolver μπορεί να προσαρμοστεί εύκολα σε πάρα πολλές εφαρμογές.[6]

Μορφή αρχείου:

Ο επιλυτής npSolver απαιτεί μια συγκεκριμένη μορφή αρχείου ως είσοδο.

Χρειάζεται στην πρώτη γραμμή του αρχείου να εμφανίζεται ο αριθμός των μεταβλητών καθώς επίσης και ο αριθμός των περιορισμών όπως φαίνεται πιο κάτω:
* #variable= 147 #constraint= 100

Επιπρόσθετα απαιτείται να περιλαμβάνονται στο αρχείο τα ακόλουθα σχόλια:

****
* begin normalizer comments
* category= OPT-SMALLINT-LIN
* end normalizer comments
****
Απαραίτητο στοιχείο είναι η αντικειμενική συνάρτηση βελτιστοποίησης του προβλήματος:
min: 7 x103 +9 x104 +9 x105 +8 x106 ;

Ακολούθως περιλαμβάνει τους περιορισμούς σε μορφή που δέχεται ο επιλυτής.
-1 x01-1 x12+1 xx01>=-1;
-1 x02-1 x11+1 xx01>=-1;
-1 x01-1 x22+1 xx02>=-1;

Κάθε μεταβλητή αποτελείται από το πρόσημο της ακολουθούμενο από το 1, ακολουθεί κενό που συμβολίζει το πολλαπλασιασμό και το όνομα της μεταβλητής.
Στο δεξιό μέρος της ανίσωσης βρίσκεται ένας ακέραιος αριθμός τον οποίο προκύπτει από την μετατροπή περιορισμού από προτασιακής μορφής σε γραμμική ψευδοδυαδική μορφή.
Ο τρόπος κλήσης του επιλυτή npSolver είναι:
$ ./npSolver-pbo όνομα_αρχείου.opb TEMP -useInc=1

4.6 Μοντελοποίηση Προβλήματος της Συνοχής στους Επιλυτές

Στην παρούσα διπλωματική στόχος ήταν η μοντελοποίηση του Πρόβλημα της Συνοχής μέσω διάφορων τεχνικών ικανοποίησης περιορισμών. Σε πρώτο μέρος έγινε η μοντελοποίηση του προβλήματος ως πρόβλημα ικανοποίησης περιορισμών με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών, μοντελοποιημένο στην γλώσσα MiniZinc ώστε να τρέχει στους επιλυτές MiniZinc και Gecode. Ακολούθως σε δεύτερο μέρος έγινε η μοντελοποίηση του προβλήματος με ψευδοδυαδικούς περιορισμούς ώστε να μπορεί να τρέχει και σε Sat επιλυτές όπως ο Minisat+ αλλά και σε ψευδοδυαδικούς επιλυτές τον Bsolo και npSolver.

Το Πρόβλημα της Συνοχής όπως ορίστηκε και πιο πάνω λαμβάνει υπόψη ένα μεγάλο αριθμό στοιχείων που είναι συνεκτικά δηλαδή ταιριάζουν μεταξύ τους ή μη συνεκτικά δηλαδή δεν ταιριάζουν μεταξύ τους, και προσπαθούμε μέσω μιας διαδικασίας να δούμε πως μπορούμε να δεχτούμε κάποια από τα στοιχεία και να απορρίψουμε άλλα με έναν τρόπο που να μεγιστοποιείται η συνοχή. Η μεγιστοποίηση της συνοχής επιτυγχάνεται με τη διαίρεση των στοιχείων σε αποδεχτά και μη αποδεχτά σύνολα ώστε να ικανοποιούνται οι περιορισμοί του προβλήματος. Τα δύο αυτά σύνολα τα συμβολίζουμε ως σύνολο 1 και σύνολο 2.

Στην παρούσα διπλωματική η απεικόνιση των στοιχείων και οι μεταξύ τους συσχετίσεις (βάρος θετικών και αρνητικών περιορισμών μεταξύ δύο στοιχείων) δημιουργούνταν από ένα πρόγραμμα στη γλώσσα C το οποίο παίρνει ως είσοδο τον αριθμό των στοιχείων καθώς επίσης και μέσω ενός μενού έδινε δύο επιλογές σχετικά με την απεικόνιση της συνεκτικότητας των στοιχείων. Η πρώτη επιλογή σήμαινε ότι τα στοιχεία συνδέονταν μεταξύ τους (βάρος θετικών και αρνητικών περιορισμών δύο στοιχείων) με τυχαίους ακέραιους αριθμούς από το 10 μέχρι το -10. Αυτοί οι αριθμοί δείχνουν πόσο συνδεδεμένα, πόσο σχετίζονται τα στοιχεία μεταξύ τους, το ποσοστό συνεκτικότητάς τους. Για παράδειγμα αν το στοιχείο X1 με το X2 πήραν τον αριθμό 10 αυτό σημαίνει ότι είναι 100% συνεκτικά. Εάν έπαιρναν τον αριθμό -9 αυτό θα σήμαινε ότι είναι 90% ασύμβατα στοιχεία , δεν συσχετίζονται 90%. Η δεύτερη επιλογή σήμαινε πως τα στοιχεία αυτά θα συνδέονται με τυχαίους ακέραιους αριθμούς από το -1 μεχρι 1. Όπου το 1 δείχνει το 100% συσχετισμό ενώ το -1 δείχνει το 100% αταίριαστα, ασύμβατα στοιχεία. Στο πρόβλημα αυτό τα στοιχεία είτε ταιριάζουν 100% είτε δεν ταιριάζουν 100%. Τέλος το πρόγραμμα αυτό έπαιρνε ως είσοδο την πυκνότητα των

στοιχείων, ένα αριθμό από το 0 μέχρι 1 ακόμα και δεκαδικό όπου αυτός ο αριθμός απεικόνιζε το ποσοστό των συσχετισμένων στοιχείων. Για παράδειγμα αν δοθεί ο αριθμός 0.25 αυτό μας λέει ότι μόνο το 25% του συνολικού αριθμού των στοιχείων σχετίζονται με άλλα στοιχεία είτε αρνητικά που δείχνει ότι δεν ταιριάζουν είτε θετικά που δείχνει ότι είναι συμβατά και ταιριάζουν.

Ακολούθως μέσω επεξεργασίας των πιο πάνω στοιχείων το πρόγραμμα αυτό δίνει ως έξοδο ένα αρχείο δεδομένων με κατάληξη .dzn στην μορφή που αναγνωρίζει ο επιλυτής MiniZinc και Gecode. Το αρχείο αυτό περιλαμβάνει των αριθμό των στοιχείων, καθώς επίσης και ένα πίνακα που ανάλογα με την δοθείσα πυκνότητα δείχνει τα συσχετισμένα στοιχεία και το ποσοστό που συσχετίζονται τα στοιχεία αυτά με άλλα στοιχεία. Επιπρόσθετα το πρόγραμμα αυτό δίνει ως έξοδο το Πρόβλημα της Συνοχής μοντελοποιημένο σε μορφή ψευδοδυαδικών περιορισμών, στην μορφή αρχείων που απαιτούσε ο Minisat+, ο Bsolo_pb10 και ο npSolver.

4.6.1 Μοντελοποίηση προβλήματος στον επιλυτή MiniZinc:

Πιο κάτω φαίνεται ένα αρχείο .dzn, όπου περιέχει τα δεδομένα που θα δοθούν ως είσοδο στο μοντέλο .mzn:

```
n=5;
table=  [|0,4,1,-2,-4,
    |4,0,7,9,2,
    |1,7,0,-3,2,
    |-2,9,-3,0,4,
    |-4,2,2,4,0      |]
```

Μοντελοποίηση του προβλήματος στη γλώσσα MiniZinc στο αρχείο .mzn.

Οι δηλώσεις των μεταβλητών του προβλήματος.

```
int: n;
set of int:nrows= 1..n;
set of int:ncols= 1..n;
```

array[nrows,ncols] of int: table;
array[nrows, ncols] of var 0..1: pinakas;
var int: kostos;
array[ncols] of var 1..2:X;

- Η ακέραια μεταβλητή n αντιπροσωπεύει τον αριθμό των στοιχείων και η τιμή της δίνεται από το αρχείο .dzn
- Οι μεταβλητές ncols, nrows είναι σύνολα μεγέθους όσος ο αριθμός των στοιχείων και αντιστοιχούν στον αριθμό των γραμμών και στηλών που χρησιμοποιούνται στην δήλωση διαστάσεων των πινάκων.
- Ο πίνακας με την ονομασία pinakas είναι δυσδιάστατος πίνακας μεγέθους nXn όπου n αριθμός των στοιχείων και κάθε θέση του θα παίρνει τιμές 0 ή 1. Θα χρησιμοποιηθεί στον υπολογισμό του κόστους στο πρόβλημα.
- Ο πίνακας table είναι δυσδιάστατος πίνακας μεγέθους nXn όπου n αριθμός των στοιχείων. Είναι ο πίνακας συνδεσιμότητας-σχετικότητας των στοιχείων που παράγεται από το πρόγραμμα στη C και αναπαριστάται στο αρχείο δεδομένων .dzn
- Η μεταβλητή «kostos» είναι μεταβλητή απόφασης (η τιμή της αποφασίζεται καθώς επιλύεται το πρόβλημα) και κρατά το κόστος της μη ικανοποίησης των περιορισμών.
- Στον πίνακα απόφασης X κάθε θέση απεικονίζει ένα στοιχείο και η τιμή στην θέση αυτή (τιμή 1 ή τιμή 2) θα αποφασίζεται από τον επιλυτή ώστε να ικανοποιούνται όσο πιο πολλά οι περιορισμοί με το λιγότερο δυνατό κόστος. Με τον πίνακα αυτό θα φαίνεται η αναπαράσταση των στοιχείων σε σύνολα. Για παράδειγμα X[1]=2 , X[2]=1 αυτό σημαίνει ότι το στοιχείο 1 τοποθετήθηκε στο σύνολο 2 ,ενώ το στοιχείο 2 στο σύνολο 1.

<u>Η μοντελοποίηση των περιορισμών</u>

Στο πρόβλημα αυτό υπάρχουν δύο σημαντικές προϋποθέσεις που οδηγούν στην δημιουργία των περιορισμών για το πρόβλημα αυτό.

Η πρώτη προϋπόθεση είναι αν δύο στοιχεία είναι συμβατά με δεδομένο ότι υπάρχει ένας θετικός περιορισμός μεταξύ τους, δηλαδή συσχετίζονται θετικά από 1 μέχρι 10, ή

με την δεύτερη επιλογή συσχέτισης με 1 πρέπει να τοποθετηθούν στο ίδιο σύνολο. Ένας θετικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί με την αποδοχή και των δύο στοιχείων σε ένα από τα δύο σύνολα ή απορρίπτοντας και τα δυο στοιχεία από ένα σύνολο. Πιο ειδικά, είτε και τα δύο στοιχεία θα τοποθετηθούν στο σύνολο 1 είτε και τα δύο θα τοποθετηθούν στο σύνολο 2.

Η δεύτερη προϋπόθεση είναι αν δύο στοιχεία δεν ταιριάζουν, συσχετίζονται αρνητικά -10 έως -1, ή με την δεύτερη επιλογή συσχέτισης με -1, άρα υπάρχει ένας αρνητικός περιορισμός μεταξύ τους, πρέπει να τοποθετηθούν σε διαφορετικά σύνολα. Ένας αρνητικός περιορισμός μεταξύ δύο στοιχείων μπορεί να ικανοποιηθεί μόνο με την αποδοχή του ενός στοιχείου σε ένα από τα δύο σύνολα και την απόρριψη του άλλου στοιχείου από το σύνολο αυτό. Πιο ειδικά, εάν το ένα στοιχείο θα μπει στο σύνολο 1 το άλλο στοιχείο θα τοποθετηθεί στο σύνολο 2.

Η μεγιστοποίηση της Συνοχής επιτυγχάνεται με τη διαίρεση των στοιχείων στα δύο σύνολα 1 και 2 όπως περιγράφτηκε πιο πάνω ώστε να ικανοποιούνται οι δύο προϋποθέσεις του προβλήματος.

Κάθε φορά που παραβιάζεται μια από τις πιο πάνω προϋποθέσεις έχει ως επίπτωση την αύξηση της τιμής του κόστους κατά την απόλυτη τιμή της συσχέτισης των δύο στοιχείων. Στόχος μας είναι η μεγιστοποίηση της συνοχής άρα η ελαχιστοποίηση της μεταβλητής του κόστους.

Η απεικόνιση των περιορισμών σε MiniZinc έγινε με τρεις διαφορετικούς τρόπους. Δημιουργήθηκαν τρία διαφορετικά μοντέλα .mzn ώστε να διαπιστωθεί πιο από αυτά βοηθά τον επιλυτή στην πιο αποδοτική αναζήτηση λύσης.

Α΄ ΤΡΟΠΟΣ: Πρώτος τρόπος απεικόνισης των περιορισμών.

Με βάση τις δύο πιο πάνω προϋποθέσεις δημιουργήθηκε ο πρώτος περιορισμός:

```
constraint forall(i in 1..n) ((forall(j in 1..n where i!=j)
(((((X[i]==X[j]) /\ (table[i,j]<0))  \/ ((X[i]!=X[j]) /\ (table[i,j]>0)) ) ->(pinakas[i,j]=1 )
                                )) ) );
```

Ο πρώτος περιορισμός ικανοποιείται όταν παραβιάζονται οι πιο πάνω προϋποθέσεις, δηλαδή αν τα στοιχεία i,j στον πίνακα table συσχετίζονται αρνητικά (table[i,j]<0) και ο

επιλυτής τα τοποθετήσει στο ίδιο σύνολο, είτε τα στοιχεία συσχετίζονται θετικά (table[i,j]>0) και ο επιλυτής τα τοποθετεί σε διαφορετικά σύνολα, τότε (->) η θέση i,j στο πίνακα pinakas θα γίνει ίσο με 1. Που αυτό δείχνει ότι τα στοιχεία δεν τοποθετήθηκαν όπως όριζε το πρόβλημα και έτσι θα υπάρχει κάποιο κόστος.
Η σχέση συνεπαγωγής δηλώνει πως εάν και μόνο αν το αριστερά μέρος της συνεπαγωγής ισχύει και αποτιμάται σε αληθής (true) τότε και το δεξιά μέρος αναγκαστικά θα ισχύει και αποτιμάται σε αληθής (true) και άρα ικανοποιείται και ο περιορισμός. Αν το αριστερά μέρος της συνεπαγωγής δεν ισχύει και αποτιμάται σε ψευδής (false) τότε και το δεξιά μέρος αναγκαστικά δεν θα ισχύει και αποτιμάται σε ψευδής (false).

Ο δεύτερος περιορισμός ικανοποιείται όταν δεν παραβιάζονται οι δύο πιο πάνω προϋποθέσεις. Δηλαδή τα στοιχεία i,j στον πίνακα table συσχετίζονται θετικά (table[i,j]>0) και ο επιλυτής τα τοποθετεί στο ίδιο σύνολο, είτε τα στοιχεία συσχετίζονται αρνητικά (table[i,j]<0) και ο επιλυτής τα τοποθετεί σε διαφορετικά σύνολα, τότε η θέση i,j στο πίνακα pinakas θα γίνει ίσο με 0. Που αυτό δείχνει ότι τα στοιχεία τοποθετήθηκαν όπως όριζε το πρόβλημα και έτσι δεν θα υπάρχει κάποιο κόστος.

```
constraint forall(i in 1..n) ((forall(j in 1..n where i!=j)(
((((X[i]!=X[j]) /\ (table[i,j]<0)) \/ ((X[i]==X[j]) /\ (table[i,j]>0))) ->(pinakas[i,j]=0)
                                    ))));
```

Τέλος ο τρίτος περιορισμός έγκειται στο να διαβάσει όλες τις θέσεις του πίνακα pinakas[i,j] και στις θέσεις όπου υπάρχει ο αριθμός 1 , σημαίνει ότι υπήρχε παραβίαση των προϋποθέσεων του προβλήματος Συνοχής, δηλαδή μη ικανοποίηση του πρώτου περιορισμού και έτσι υπολογίζεται το κόστος για την παραβίαση αυτή. Για τον υπολογισμό του κόστους πολλαπλασιάζεται ο αριθμός 1 με την απόλυτη τιμή της συσχέτισης των δύο στοιχείων από τον πίνακα table[i.j]. Ο πιο πάνω περιορισμός μεταφράζεται ως εξής :

```
constraint ((sum (i in 1..n,j in i+1..n where j!=0)
                                    ((abs(table[i,j])) * pinakas[i,j]) )= kostos );
```

Τέλος κατευθύνουμε τον επιλυτή στο να αναζητήσει μια βέλτιστη λύση ώστε να επιτευχθεί το μικρότερο δυνατό κόστος (minimize kostos). Χρησιμοποιείται και ένα είδος ευρετικής αναζήτησης. Αφού η λύση θα είναι ακέραια χρησιμοποιοείται το ευρετικό
:: int_search(X,first_fail,indomain_min,complete). Το ευρετικό αυτό παίρνει ως παράμετρο τον πίνακα απόφασης που θα δείχνει σε πιο από τα δύο σύνολα (1 ή 2) θα ανήκει κάθε μεταβλητή. Με την τρίτη παράμετρο indomain_min επιλέγονται πρώτα οι μεταβλητές για να πάρουν τιμή αυτές με το μικρότερο πεδίο ορισμού (domain).

```
solve :: int_search(
        X,
        first_fail,
        indomain_min,
        complete
        )
        minimize kostos;
```

Απεικόνιση του αποτελέσματος βέλτιστης λύσης στην οθόνη με τον πιο κάτω τρόπο:

```
output[ show(X[i])| i in 1..n]++["kostos= "]++[show(kostos) ]++["\n"];
```

Θα τυπώνεται στην οθόνη ο πίνακας X[i] όπου $i \geq 0$ και $i \leq n$ (n = αριθμός των στοιχείων) και το συνολικό κόστος. Μέσω του πίνακα αυτού θα μπορούμε να δούμε πως διαιρέθηκαν τα στοιχεία στα σύνολα 1 και 2 ώστε να επιτύχουμε βέλτιστη λύση, με το λιγότερο δυνατό κόστος, άρα επιτυχία μεγιστοποίησης της Συνοχής.

Β΄ΤΡΟΠΟΣ: Δεύτερος τρόπος απεικόνισης πρώτου και δεύτερου περιορισμού.

Η μετάφραση των πιο πάνω περιορισμών έγινε με την ιδέα της απαλοιφής των συνεπαγωγών και των συζεύξεων και των διαζεύξεων. Αντικαταστάθηκαν με μαθηματικούς τελεστές (+,-,=) ώστε να διατηρείται το νόημα των περιορισμών όπως επεξηγήθηκαν πιο πάνω.

Ο πρώτος περιορισμός ικανοποιείται στις περιπτώσεις στις οποίες συνδέονται αρνητικά δύο στοιχεία και ο επιλυτής τοποθετεί τα δύο στοιχεία στο ίδιο σύνολο και έτσι στον πίνακα pinakas στην θέση i,j θα μπει η τιμή 1 που δείχνει ότι δεν τοποθετήθηκαν όπως όριζε το πρόβλημα και θα υπάρχει κόστος, καθώς περιλαμβάνει και την περίπτωση στην οποία τα στοιχεία συνδέονται αρνητικά και ο επιλυτής τα τοποθετήσει σε διαφορετικά σύνολα, άρα στον πίνακα pinakas στην θέση i,j θα μπει η τιμή 0 που δείχνει ότι τοποθετήθηκαν όπως όριζε το πρόβλημα και δεν θα υπάρχει κόστος. Ο περιορισμός αυτός μεταφράζεται όπως πιο κάτω:

```
constraint forall(i in 1..n) (( forall(j in 1..n where i!=j )((
        (( (abs(X[i]-X[j]))+ (table[i,j])+ (abs(table[i,j]))+ pinakas[i,j])=1 )
```

Ο δεύτερος περιορισμός ικανοποιείται στις περιπτώσεις στις οποίες συνδέονται θετικά δύο στοιχεία και ο επιλυτής τοποθετεί τα δύο στοιχεία σε διαφορετικά σύνολα και έτσι στον πίνακα pinakas στην θέση i,j θα μπει η τιμή 1 που δείχνει ότι δεν τοποθετήθηκαν όπως όριζε το πρόβλημα και θα υπάρχει κόστος, καθώς περιλαμβάνει και την περίπτωση στην οποία τα στοιχεία συνδέονται θετικά και ο επιλυτής τα τοποθετήσει στο ίδιο σύνολο, άρα στον πίνακα pinakas στην θέση i,j θα μπει η τιμή 0 που δείχνει ότι τοποθετήθηκαν όπως όριζε το πρόβλημα και δεν θα υπάρχει κόστος. Ο περιορισμός αυτός μεταφράζεται όπως πιο κάτω:

```
constraint forall(i in 1..n) (( forall(j in 1..n where i!=j )((
(table[i,j]+(abs(table[i,j]))+ pinakas[i,j]=table[i,j] +table[i,j]+((abs(-X[i]-X[j])) mod 2))
)))) ;
```

Γ΄ΤΡΟΠΟΣ: Τρίτος τρόπος απεικόνισης του πρώτου και δεύτερου περιορισμού.

Ο τρίτος τρόπος απεικόνισης έγινε με την ιδέα της αναπαράστασης της λύσης αντί σε πίνακα όπως με τον πρώτο τρόπο σε σύνολα.
Χρειάζεται να γίνει ενσωμάτωση του αρχείου "partition_set.mzn" όπως πιο κάτω:

```
include "partition_set.mzn";
```

Τώρα ο πίνακας απόφασης X δηλώνεται ως ένα πίνακας δύο θέσεων, θέση 1 και θέση 2 και συμβολίζουν τα δύο σύνολα. Κάθε θέση θα περιέχει ένα σύνολο με τις μεταβλητές (τα στοιχεία) που θα ανήκουν στις θέσεις αυτές.

```
array[1..2] of var set of nrows: X;
```

Προστίθεται ένας καινούριος περιορισμός ο οποίος δηλώνει το διαχωρισμό των στοιχείων σε δύο σύνολα, στο σύνολο 1 και στο σύνολο 2. Ο περιορισμός αυτός παίρνει ένα σύνολο στοιχείων και το σπάει σε υποσύνολα (1 ή 2). Στο πρόβλημα αυτό παίρνει παράμετρο το σύνολο από τα στοιχεία στο γράφο συνοχής και τα διαχωρίζει σε μια από τις δύο θέσεις του πίνακα απόφασης X. Μέσω της συνάρτησης partition_set διαχωρίζει κάθε στοιχείο σε ένα από τα δύο σύνολα δηλαδή σε μια από τις δύο θέσεις του πίνακα X (θέση X[1] ή θέση X[2]). Οι μεταβλητές είναι όσες ο αριθμός των στοιχείων.

```
constraint partition_set([X[i] | i in 1..2], nrows);
```

Τροποποιούνται ως εξής οι πρώτοι δύο περιορισμοί χωρίς να χάνουν το νόημα τους:

```
constraint forall(i in 1..nodes) ((forall(j in 1..nodes where i!=j)(
(((((i in X[1])==(j in X[1])) \/ (((i in X[2])==(j in X[2]))) ) /\ (table[i,j]<0))   \/
((((i in X[1])!=(j in X[1])) \/ (((i in X[2])!=(j in X[2]))) ) /\ (table[i,j]>0))
 -> (pinakas[i,j]=1 ))) )) ;
```

Ο περιορισμός ικανοποιείται στην περίπτωση όπου τα στοιχεία i,j τοποθετούνται στο ίδιο σύνολο, δηλαδή το στοιχείο i στη θέση 1 στον πίνακα X και το στοιχείο j στη θέση 1 στον πίνακα X ή το στοιχείο i στη θέση 2 στον πίνακα X και το στοιχείο j στη θέση 2 στον πίνακα X και συνδέονται αρνητικά τότε στον πίνακα pinakas στην θέση i,j μπαίνει η τιμή 1 που δείχνει ότι τα στοιχεία δεν τοποθετήθηκαν όπως όριζε το πρόβλημα και θα έχει κάποιο κόστος. Επίσης ο περιορισμός αυτός ικανοποιείται στην περίπτωση όπου τα στοιχεία i,j τοποθετούνται σε διαφορετικά σύνολα, δηλαδή το στοιχείο i στη θέση 1 στον πίνακα X και το στοιχείο j στη θέση 2 στον πίνακα X ή το στοιχείο i στη θέση 2 στον πίνακα X και το στοιχείο j στη θέση 1 στον πίνακα X και συνδέονται θετικά τότε στον πίνακα pinakas στην θέση i,j μπαίνει η τιμή 1 που δείχνει

ότι τα στοιχεία δεν τοποθετήθηκαν όπως όριζε το πρόβλημα και θα έχει κάποιο κόστος.

```
constraint forall(i in 1..nodes) ((forall(j in 1..nodes where i!=j)(
(( (((i in X[1])!=(j in X[1])) \/ (((i in X[2])!=(j in X[2]))) ) /\ (table[i,j]<0))  \/
((((i in X[1])==(j in X[1])) \/ (((i in X[2])==(j in X[2]))) ) /\ (table[i,j]>=0))
->(pinakas[i,j]=0 ))) )) ;
```

Ο περιορισμός αυτός ικανοποιείται στην περίπτωση όπου τα στοιχεία i,j τοποθετούνται στο ίδιο σύνολο δηλαδή το στοιχείο i στη θέση 1 στον πίνακα X και το στοιχείο j στη θέση 1 στον πίνακα X ή το στοιχείο i στη θέση 2 στον πίνακα X και το στοιχείο j στη θέση 2 στον πίνακα X και συνδέονται θετικά τότε στον πίνακα pinakas στην θέση i,j μπαίνει η τιμή 0 που δείχνει ότι τα στοιχεία τοποθετήθηκαν όπως όριζε το πρόβλημα και δεν θα έχει κάποιο κόστος. Επίσης ο περιορισμός αυτός ικανοποιείται στην περίπτωση όπου τα στοιχεία i,j τοποθετούνται σε διαφορετικά σύνολα, δηλαδή το στοιχείο i στη θέση 1 στον πίνακα X και το στοιχείο j στη θέση 2 στον πίνακα X ή το στοιχείο i στη θέση 2 στον πίνακα X και το στοιχείο j στη θέση 1 στον πίνακα X και συνδέονται αρνητικά τότε στον πίνακα pinakas στην θέση i,j μπαίνει η τιμή 0 που δείχνει ότι τα στοιχεία τοποθετήθηκαν όπως όριζε το πρόβλημα και δεν θα έχει κάποιο κόστος.

Αλλάζει και ο τρόπος αναζήτησης του επιλυτή. Χρησιμοποιείται το ευρετικό αναζήτησης σε πίνακα συνόλων, και όχι σε πίνακα ακεραίων όπως στο πρώτο παράδειγμα.

```
solve :: set_search(
        [X[i]|i in 1..2],
        first_fail,
        indomain_min,
        complete
        )
        minimize kostos;
```

4.6.2 Μοντελοποίηση στους επιλυτές Minisat+ και Bsolo

Η μοντελοποίηση του προβλήματος συνοχής σε μορφή ψευδοδυαδικών περιορισμών έγινε με την μετατροπή περιορισμών σε διαζευκτική μορφή και ακολούθως σε γραμμική ψευδοδυαδική μορφή περιορισμών που περιγράφτηκε στην ενότητα 3. Όλη η διαδικασία μοντελοποίησης των περιορισμών και της αντικειμενικής συνάρτησης στην μορφή που απαιτεί ο επιλυτής Minisat+,bsolo_pb10 και npSolver έγινε μέσω του προγράμματος στη γλώσσα C.

Μεταβλητές:

Τώρα οι μεταβλητές θα παίρνουν τιμές 0 και 1 αντί 1 ή 2 που έπαιρναν στο πρόβλημα ικανοποίησης περιορισμών (CSP). Η κωδικοποίηση των μεταβλητών αλλάζει. Οι μεταβλητές θα έχουν την μορφή Xi1 που αν πάρει την τιμή 1, συμβολίζει ότι το στοιχείο i τοποθετείται στο σύνολο 1 και Xi2 που αν πάρει την τιμή 2 ,συμβολίζει ότι το στοιχείο i τοποθετείται στο σύνολο 2. Το i έχει εμβέλεια όσα και τα στοιχεία. Επιπρόσθετα υπάρχουν ακόμα δύο ειδών μεταβλητές οι XX_{ij} και XM_{ij}. Όπου η XX_{ij} αν πάρει τιμή 1 σημαίνει ότι τα στοιχεία i και j τοποθετήθηκαν σε ξεχωριστά σύνολα, ενώ αν η XM_{ij} πάρει τιμή 1 σημαίνει ότι τα στοιχεία i και j τοποθετήθηκαν στο ίδιο σύνολο.

Περιορισμοί:

Για κάθε στοιχεία i,j όπου i≠j και συνδέονται αρνητικά στον πίνακα table[i,j] (γίνεται έλεγχος μέσα από το πρόγραμμα σε C), τότε τα στοιχεία αυτά σύμφωνα με τις δύο προϋποθέσεις του Προβλήματος Συνοχής πρέπει να τοποθετηθούν σε διαφορετικά σύνολα ώστε να μεγιστοποιηθεί η Συνοχή. Εάν το i στοιχείο θα τοποθετηθεί στο σύνολο 1 τότε το j θα πρέπει να τοποθετηθεί στο σύνολο 2, ή αν το i στοιχείο θα τοποθετηθεί στο σύνολο 2, το j θα πρέπει να τοποθετηθεί στο σύνολο1. Έτσι προκύπτουν δύο περιορισμοί για τα στοιχεία αυτά:

$1^{ος}$ περιορισμός:

X_i1 /\ X_j1 -> XM_{ij} το i στοιχείο τοποθετείται στο σύνολο 1 και το j στοιχείο τοποθετείται στο σύνολο 1 τότε συνεπάγεται XMij, όπου δηλώνει ότι τα στοιχεία i,j είναι τοποθετημένα στα ίδια σύνολα.

$2^{ος}$ περιορισμός:

X_i2 /\ X_j2 -> XM_{ij} το i στοιχείο τοποθετείται στο σύνολο 2 και το j στοιχείο τοποθετείται στο σύνολο 2 τότε συνεπάγεται XM_{ij}, όπου δηλώνει ότι τα στοιχεία i,j είναι τοποθετημένα στα ίδια σύνολα.

Για κάθε στοιχεία i,j όπου i≠j συνδέονται θετικά στον πίνακα table[i,j], γίνεται έλεγχος μέσα από το πρόγραμμα σε C, τότε τα στοιχεία αυτά σύμφωνα με τις δύο προϋποθέσεις του Προβλήματος Συνοχής πρέπει να τοποθετηθούν στα ίδια σύνολα ώστε να μεγιστοποιηθεί η Συνοχή. Είτε και τα δύο στοιχεία στο σύνολο 1 είτε και τα δυο στοιχεία στο σύνολο 2. Έτσι προκύπτουν οι ακόλουθοι περιορισμοί για τα στοιχεία αυτά:

$1^{ος}$ περιορισμός:

X_i1 /\ X_j2 -> XX_{ij} το i στοιχείο τοποθετείται στο σύνολο 1 και το j στοιχείο τοποθετείται στο σύνολο 2 τότε συνεπάγεται XX_{ij}, όπου δηλώνει ότι τα στοιχεία i,j είναι τοποθετημένα σε διαφορετικά σύνολα.

$2^{ος}$ περιορισμός:

Xi2 /\ Xj1 -> XXij το i στοιχείο τοποθετείται στο σύνολο 2 και το j στοιχείο τοποθετείται στο σύνολο 2 τότε συνεπάγεται XXij, όπου δηλώνει ότι τα στοιχεία i,j είναι τοποθετημένα σε διαφορετικά σύνολα.

Καθένας από τους πιο πάνω περιορισμούς με την μέθοδο απαλοιφής της συνεπαγωγής μετατρέπεται σε διαζευκτική μορφή. Για κάθε πρόταση σε διαζευκτική μορφή υπολογίζεται το πλήθος των αρνητικών λεκτικών (literals) στην πρόταση και το αφαιρούμε από το 1. Το αποτέλεσμα της αφαίρεσης δείχνει τον συνολικό αριθμό λεκτικών (literals) που πρέπει να ικανοποιούνται και το αποτέλεσμα μπαίνει στο δεξιό μέρος της ανίσωσης. Ακολούθως ο περιορισμός μετατρέπεται σε μορφή γραμμικών ψευδοδυαδικών περιορισμών όπως πιο κάτω:[8]

1ος περιορισμός (τα στοιχεία συσχετίζονται αρνητικά):

Xi1 /\ Xj1 -> XMij

= ┐ Xi1 \/ ┐ Xj1 \/ XMij (απαλοιφή συνεπαγωγής-διαζευκτική μορφή)

= (1- Xi1) + (1- Xj1) + XMij ≥1 (1- το πλήθος των αρνητικών όρων)

= (2-Xi1- Xj1) + XMij ≥1

Όλοι οι ακέραιοι μεταφέρονται στο δεξιά μέρος

=-Xi1- Xj1 + XMij ≥-1

Στον επιλυτή minisat+ η τελική μορφή του περιορισμού γράφεται στο αρχείο μέσω του προγράμματος στη C στην πιο κάτω μορφή :

=-1* Xi1- 1* Xj1 + 1* XMij ≥-1 ;

2ος περιορισμός (τα στοιχεία συσχετίζονται αρνητικά):

Xi2 /\ Xj2 -> XMij

= ┐ Xi2 \/ ┐ Xj2 \/ XMij (διαζευκτική μορφή)

= (1- Xi2) + (1- Xj2) + XMij ≥1 (1- το πλήθος των αρνητικών όρων)

= (2-Xi2- Xj2) + XMij ≥1

Όλοι οι ακέραιοι μεταφέρονται στο δεξιά μέρος

=-Xi2- Xj2 + XMij ≥-1

Στον επιλυτή minisat+ η τελική μορφή του περιορισμού γράφεται στο αρχείο μέσω του προγράμματος στη C στην πιο κάτω μορφή :

=-1* Xi2- 1* Xj2 + 1* XMij ≥-1 ;

1ος περιορισμός (τα στοιχεία συσχετίζονται θετικά):

Xi1 /\ Xj2 -> XXij

= ┐ Xi1 \/ ┐ Xj2 \/ XXij (διαζευκτική μορφή)

= (1- Xi1) + (1- Xj2) + XXij ≥1(1- το πλήθος των αρνητικών όρων)

= (2-Xi1- Xj2) + XXij ≥1

Όλοι οι ακέραιοι μεταφέρονται στο δεξιά μέρος

=-Xi1- Xj2 + XXij ≥-1

Στον επιλυτή minisat+ η τελική μορφή του περιορισμού γράφεται στο αρχείο μέσω του προγράμματος στη C στην πιο κάτω μορφή :

=-1* Xi1- 1* Xj2 + 1* XXij ≥-1 ;

2ος περιορισμός (τα στοιχεία συσχετίζονται θετικά):

Xi2 /\ Xj1 -> XXij

= ┐Xi2 ∨ ┐Xj1 ∨ XXij (διαζευκτική μορφή)

= (1- Xi2) + (1- Xj1) + XXij ≥1 (1- το πλήθος των αρνητικών όρων)

= (2-Xi2- Xj1) + XXij ≥1

Όλοι οι ακέραιοι μεταφέρονται στο δεξιά μέρος

=-Xi2- Xj1 + XXij ≥-1

Στον επιλυτή minisat+ η τελική μορφή του περιορισμού γράφεται στο αρχείο μέσω του προγράμματος στη C στην πιο κάτω μορφή :

=-1* Xi2- 1* Xj1 + 1* XXij ≥-1 ;

Προκύπτει ακόμα ένα είδος περιορισμού

Ο περιορισμός αυτός περιορίζει τις μεταβλητές ότι σε ένα από τα δύο σύνολα μπορούν να τοποθετηθούν. Είτε ένα στοιχείο θα τοποθετείται στο σύνολο 1 είτε ένα στοιχείο θα τοποθετείται στο σύνολο 2. Δεν μπορεί ένα στοιχείο να ανήκει και στο σύνολο 1 και στο σύνολο 2 γιατί έτσι δεν θα γινόταν σωστή διαίρεση στοιχείων όπως ορίζει το Πρόβλημα της Συνοχής. Ο περιορισμός αυτός είναι της μορφής Xi1∨ Xi2 ,όπου καθορίζει ότι το στοιχείο i θα τοποθετηθεί είτε στο σύνολο 1 είτε στο σύνολο 2.

Έτσι για κάθε μεταβλητή i, όπου i έχει εμβέλεια όσος ο αριθμός των στοιχείων ο πιο πάνω περιορισμός μετατρέπεται σε γραμμικό ψευδοδυαδικό περιορισμό στην μορφή που απαιτεί ο κάθε επιλυτής.

Στον επιλυτή Minisat+:

1* Xi1+1* Xi2 =1 ;

Ακολούθως θα πρέπει να οριστεί και η αντικειμενική συνάρτηση. Η αντικειμενική συνάρτηση ορίζει την μεγιστοποίηση της συνοχής και άρα την ελαχιστοποίηση του κόστους. Το κόστος θα αυξάνεται με την μη ικανοποίηση των πιο πάνω περιορισμών. Για κάθε ζεύγος στοιχείων i,j στον άνω τριγωνικό πίνακα (όπου i<j) table[i,j] που

δημιουργείται με βάση τα δεδομένα που παίρνει ως είσοδο το πρόγραμμα σε C γίνεται έλεγχος:

Αν συνδέονται αρνητικά δύο στοιχεία τότε θα πολλαπλασιαστεί η απόλυτη τιμή σύνδεσης των στοιχείων αυτών table[i,j] με την μεταβλητή XMij. Αν η μεταβλητή XMij πάρει την τιμή 1, αυτό δηλώνει ότι τα στοιχεία i,j δεν τοποθετήθηκαν όπως όριζε το πρόβλημα άρα υπάρχει κόστος ίσο με την απόλυτη τιμή του αριθμού στον πίνακα table[i,j]. Αν πάλι η μεταβλητή XMij πάρει την τιμή 0, δεν τοποθετήθηκαν στο ίδιο σύνολο, άρα δεν θα αυξηθεί το κόστος αφού ικανοποιεί τους περιορισμούς του προβλήματος συνοχής.

Αν από την άλλη συνδέονται θετικά δύο στοιχεία τότε θα πολλαπλασιαστεί η απόλυτη τιμή σύνδεσης των στοιχείων αυτών table[i,j] με την μεταβλητή XXij. Αν η μεταβλητή XXij πάρει την τιμή 1, αυτό δηλώνει ότι δεν τοποθετήθηκαν όπως όριζε το πρόβλημα, τα στοιχεία τοποθετήθηκαν σε ξεχωριστά σύνολα άρα υπάρχει κόστος ίσο με την απόλυτη τιμή του αριθμού στον πίνακα table[i,j]. Αν πάλι η μεταβλητή XXij πάρει την τιμή 0, τοποθετήθηκαν στο ίδιο σύνολο ,άρα δεν θα αυξηθεί το κόστος αφού ικανοποιεί τους περιορισμούς του Προβλήματος Συνοχής.

Τελική μορφή αντικειμενικής συνάρτησης που εξάγεται στο αρχείο από το πρόγραμμα στη C έχει την πιο κάτω γενική μορφή για τον επιλυτή Minisat+:

min: $\sum ((\text{abs}(\text{table}[i,j])) * X)$, όπου table[i,j]≠0 και X θα είναι η μεταβλητή XXij ή η μεταβλητή XMij ανάλογα όπως έχουν εξηγηθεί πιο πάνω.

Παρομοίως ίδια διαδικασία μοντελοποίησης του Προβλήματος Συνοχής γίνεται και στον επιλυτή Bsolo.

Ολοκληρωμένες μορφές των αρχείων Minisat+ και Bsolo_pb10 φαίνονται στα σχήματα 4.2 και 4.3 αντίστοιχα.

```
min:1*xm12+2*xm13+5*xm14+6*xm15+7*xm23+9*xm24+4*xm25;
1*x11+1*x12=1;
1*x21+1*x22=1;
1*x31+1*x32=1;
1*x41+1*x42=1;
1*x51+1*x52=1;
-1*x11-1*x21+1*xm12>=-1;
-1*x12-1*x22+1*xm12>=-1;
-1*x11-1*x31+1*xm13>=-1;
-1*x12-1*x32+1*xm13>=-1;
-1*x11-1*x41+1*xm14>=-1;
-1*x12-1*x42+1*xm14>=-1;
-1*x11-1*x51+1*xm15>=-1;
-1*x12-1*x52+1*xm15>=-1;
-1*x21-1*x31+1*xm23>=-1;
-1*x22-1*x32+1*xm23>=-1;
-1*x21-1*x41+1*xm24>=-1;
-1*x22-1*x42+1*xm24>=-1;
-1*x21-1*x51+1*xm25>=-1;
-1*x22-1*x52+1*xm25>=-1;
-1*x31-1*x42+1*xx34>=-1;
-1*x32-1*x41+1*xx34>=-1;
-1*x31-1*x52+1*xx35>=-1;
-1*x32-1*x51+1*xx35>=-1;
-1*x41-1*x52+1*xx45>=-1;
-1*x42-1*x51+1*xx45>=-1;
```

Σχήμα 4.2 : Αρχείο με ψευδοδυαδικούς περιορισμούς για Minisat+

```
min: 1 xm12 +2 xm13 +5 xm14 +6 xm15 +7 xm23 +9 xm24 +4 xm25 ;
 +1 x11 +1 x12 =  1 ;
 +1 x21 +1 x22 =  1 ;
 +1 x31 +1 x32 =  1 ;
 +1 x41 +1 x42 =  1 ;
 +1 x51 +1 x52 =  1 ;
-1 x11 -1 x21 +1 xm12 >= -1 ;
-1 x12 -1 x22 +1 xm12 >= -1 ;
-1 x11 -1 x31 +1 xm13 >= -1 ;
-1 x12 -1 x32 +1 xm13 >= -1 ;
-1 x11 -1 x41 +1 xm14 >= -1 ;
-1 x12 -1 x42 +1 xm14 >= -1 ;
-1 x11 -1 x51 +1 xm15 >= -1 ;
-1 x12 -1 x52 +1 xm15 >= -1 ;
-1 x21 -1 x31 +1 xm23 >= -1 ;
-1 x22 -1 x32 +1 xm23 >= -1 ;
-1 x21 -1 x41 +1 xm24 >= -1 ;
-1 x22 -1 x42 +1 xm24 >= -1 ;
-1 x21 -1 x51 +1 xm25 >= -1 ;
-1 x22 -1 x52 +1 xm25 >= -1 ;
-1 x31 -1 x42 +1 xx34 >= -1 ;
-1 x32 -1 x41 +1 xx34 >= -1 ;
-1 x31 -1 x52 +1 xx35 >= -1 ;
-1 x32 -1 x51 +1 xx35 >= -1 ;
-1 x41 -1 x52 +1 xx45 >= -1 ;
-1 x42 -1 x51 +1 xx45 >= -1 ;
```

Σχήμα 4.3: Αρχείο με ψευδοδυαδικούς περιορισμούς για Bsolo_pb10

4.6.3 Μοντελοποίηση στον επιλυτή npSolver

Η μοντελοποίηση του Προβλήματος Συνοχής στον npSolver διαφέρει ελαφρώς από τους δύο προηγούμενους επιλυτές. Υπάρχει μια διαφοροποίηση στην κωδικοποίηση των μεταβλητών. Δεν αναγνωρίζεται η μεταβλητή XXij ή XMij. Ο επιλυτής αυτός δέχεται μόνο μεταβλητές της μορφής Xnum όπου num ένας αριθμός. Για το λόγο αυτό μετατράπηκαν οι μεταβλητές αυτές στην μορφή που μας επέτρεπε ο επιλυτής. Ο τρόπος με τον οποίο έγινε αυτό ήταν βρίσκοντας με βάση τον αριθμό των στοιχείων την τελευταία χρησιμοποιημένη μεταβλητή. Αν για παράδειγμα είχα 5 στοιχεία η τελευταία μεγαλύτερη και χρησιμοποιούμενη μεταβλητή θα ήταν η X52 που υποδήλωνε ότι η μεταβλητή 5 στο σύνολο 2, όμως ο npSolver καταλαβαίνει ότι είναι η μεταβλητή με αριθμό 52. Έτσι ακολούθως όπου έπρεπε να χρησιμοποιηθούν οι μεταβλητές XXij και XMij αυξανόταν ο αριθμός αυτός (52).

Επιπρόσθετα απαραίτητο στον επιλυτή αυτό είναι η πρώτη γραμμή του αρχείου να ορίζει πόσες μεταβλητές χρησιμοποιούνταν και πόσους περιορισμούς περιλαμβάνονται στο αρχείο αυτό. Ο αριθμός των μεταβλητών θα είναι όσος η μεγαλύτερη χρησιμοποιούμενη μεταβλητή συν τα μη μηδενικά στοιχεία του άνω τριγωνικού πίνακα table[i,j].

Ο αριθμός των περιορισμών που προκύπτουν θα είναι όσος και ο αριθμός των μη μηδενικών στοιχείων * 2 + r όπου r είναι ο αριθμός των στοιχείων.

Μέσω επεξεργασίας των στοιχείων και βασισμένο στις πιο πάνω αλλαγές έγινε η μοντελοποίηση του αρχείου το οποίο γίνεται δεχτό μέσω από το πρόγραμμα στη γλώσσα C. Ένα εμφανές παράδειγμα μορφής του npSolver φαίνεται στο σχήμα 4.4 όπου μέσα στο κόκκινο κουτί φαίνεται η τελευταία χρησιμοποιούμενη μεταβλητή για το παράδειγμα 5 στοιχείων. Έτσι οι περιορισμοί που θα χρησιμοποιούσαν τις μεταβλητές XXij και XMij αντικαθιστούνται με αύξων αριθμό από την τελευταία μεγαλύτερη και χρησιμοποιούμενη μεταβλητή.

```
* #variable= 62 #constraint= 25
**************************************
* begin normalizer comments
* category= OPT-SMALLINT-LIN
* end normalizer comments
**************************************
min: 1 x53 +2 x54 +5 x55 +6 x56 +7 x57 +9 x58 +4 x59 +0 x60 +0 x61 +0 x62
+1 x11 +1 x12 =  1 ;
+1 x21 +1 x22 =  1 ;
+1 x31 +1 x32 =  1 ;
+1 x41 +1 x42 =  1 ;
+1 x51 +1 x52 =  1 ;
-1 x11 -1 x21 +1 x53 >= -1 ;
-1 x12 -1 x22 +1 x53 >= -1 ;
-1 x11 -1 x31 +1 x54 >= -1 ;
-1 x12 -1 x32 +1 x54 >= -1 ;
-1 x11 -1 x41 +1 x55 >= -1 ;
-1 x12 -1 x42 +1 x55 >= -1 ;
-1 x11 -1 x51 +1 x56 >= -1 ;
-1 x12 -1 x52 +1 x56 >= -1 ;
-1 x21 -1 x31 +1 x57 >= -1 ;
-1 x22 -1 x32 +1 x57 >= -1 ;
-1 x21 -1 x41 +1 x58 >= -1 ;
-1 x22 -1 x42 +1 x58 >= -1 ;
-1 x21 -1 x51 +1 x59 >= -1 ;
-1 x22 -1 x52 +1 x59 >= -1 ;
-1 x31 -1 x42 +1 x60 >= -1 ;
-1 x32 -1 x41 +1 x60 >= -1 ;
-1 x31 -1 x52 +1 x61 >= -1 ;
-1 x32 -1 x51 +1 x61 >= -1 ;
-1 x41 -1 x52 +1 x62 >= -1 ;
-1 x42 -1 x51 +1 x62 >= -1 ;
```

Σχήμα 4.4 Μορφή αρχείου με ψευδοδυαδικούς περιορισμούς στον npSolver

4.6.4 Μοντελοποίηση στον επιλυτή Gecode

Στην παρούσα διπλωματική το μοντέλο το οποίο έτρεχε στον επιλυτή MiniZinc, το ίδιο μοντέλο αναλλοίωτο τρέχει και στον Gecode.

4.6.4 .1 Μοντελοποίηση με χρήση συνδυαστών αναζήτησης

Ο επιλυτής αυτός όπως ανάφερα και πιο πάνω έχει την ιδιαιτερότητα ενσωμάτωσης ευρετικών αναζήτησης. Μέσω μελέτης πολλών ευρετικών που υπάρχουν υλοποιημένα, το ευρετικό το οποίο είχε την καλύτερη απόδοση στο Πρόβλημα της Συνοχής ήταν το ευρετικό branch-and-bound σε συνδυασμό με άλλα ευρετικά αναζήτησης που φαίνεται πιο κάτω:

```
annotation bab(var int: obj, ann:s) =
   let {
     svar int: best = 1000000
   } in (
```

```
    post(obj < lv("best"),
      and(s,assign(best,obj))
    )
  );
```

Το ευρετικό αυτό περιλαμβάνει μια μεταβλητή «best» η οποία αρχικά παίρνει μια πολύ μεγάλη τιμή 1000000 για ελαχιστοποίηση. Για κάθε κόμβο υπολογίζεται με βάση τους περιορισμούς μια νέα τιμή της αντικειμενικής μεταβλητής και συγκρίνεται με την «best». Όταν μια νέα λύση βρεθεί, η τιμή αυτή ενημερώνεται μέσω της συνάρτησης(annotation) assign.

Το ευρετικό αυτό σε συνδυασμό με άλλα ευρετικά ενσωματώνονται στο μοντέλο. Η διαφορά είναι στο σημείο καλέσματος του επιλυτή στο να βρει την βέλτιστη λύση όπου γίνεται με τον εξής τρόπο:

Solve:: print(X,bab(kostos,int_search(X,random_order,assign_lb))) **satisfy**;

Καθώς επίσης πρέπει να συμπεριλαμβάνονται στο αρχείο τα πρωτότυπα συναρτήσεων (annotation) αυτών που χρησιμοποιούμε και τα υποστηρίζει ο επιλυτής αυτός, όπως για παράδειγμα:

```
ann: random_order = sh_var_random_order;
annotation sh_var_min_lb;

annotation sh_print_int(array[int] of var int: x, ann:s);
annotation sh_val_assign_lb;
ann: assign_lb = sh_val_assign_lb;

annotation sh_int_search(array[int] of var int: x, ann:varsel,ann:valsel);
```

Κεφάλαιο 5

Πειραματική Ανάλυση και Συμπεράσματα

5.1 Πειραματική Ανάλυση.....58
5.1.1 Πειράματα διαφορετικών μοντέλων σε MiniZinc.....58
5.1.2 Πειράματα διαφορετικών παραμέτρων στον Bsolo.....62
5.1.3 Πειράματα μεταξύ όλων των επιλυτών ανάλογα με πυκνότητα.....65
5.1.4 Πειράματα μεταξύ όλων των επιλυτών με διαφορετικό τρόπο απεικόνισης της συσχέτισης δύο στοιχείων.....73

5.1 Πειραματική Ανάλυση

Στο κεφάλαιο αυτό μετά από την μοντελοποίηση του Προβλήματος Συνοχής στους διάφορους επιλυτές μέσω διαφόρων τεχνικών ικανοποίησης περιορισμών, έγιναν διάφορα πειράματα. Τα πειράματα αυτά στόχο είχαν να προσδιοριστεί ποιος από τους επιλυτές είναι ο πιο αποδοτικός για το Πρόβλημα της Συνοχής.

5.1.1 Πειράματα διαφορετικών μοντέλων σε MiniZinc

Το πρώτο πείραμα το οποίο έγινε κατά την διάρκεια της εκπόνησης της παρούσας διπλωματικής, ήταν η εύρεση του πιο γρήγορου από τα τρία μοντέλα στον επιλυτή MiniZinc τα οποία επεξηγήθηκαν στο κεφάλαιο 4.
Στον πίνακα 5.1 αναφέρονται συγκριτικά οι χρόνοι αναζήτησης βέλτιστης λύσης με N αριθμό στοιχείων, όπου N{5-22,23,25,27,29}. Τα στοιχεία συσχετίζονται μεταξύ τους με ακέραιους αριθμούς από -10 μέχρι 10. Η πυκνότητα των στοιχείων είναι 100%. Αυτά ισχύουν και για τα τρία διαφορετικά μοντέλα σε MiniZinc για το Πρόβλημα της Συνοχής. Τα μοντέλα αυτά είναι το coher_set2.mzn το οποίο είναι υλοποιημένο μέσω του partition_set και το αποτέλεσμα αναπαριστάται σε 2 σύνολα, όπου κάθε σύνολο

περιέχει τα στοιχεία που ανήκουν σε αυτό. Το finaltest2.mzn το οποίο είναι υλοποιημένο με περιορισμούς σε μορφή αριθμητικών περιορισμών. Και το coher.mzn το οποίο οι περιορισμοί μου είναι υλοποιημένοι με χρήση συνεπαγωγών.

Για τις μετρήσεις όπως είναι εμφανές και στον Πίνακα 5.1 , το ίδιο αρχείο δεδομένων .dzn ήταν είσοδος και στα τρία μοντέλα .mzn. Πάρθηκε δείγμα πέντε αρχείων με ίδιο αριθμό στοιχείων και τα στοιχεία συσχετίζονται μεταξύ τους με ακεραίους από -10 μέχρι 10, ώστε να προκύψουν ορθά συμπεράσματα. Στον Πίνακα 5.1 αναγράφονται οι χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στην μορφή λεπτά : δευτερόλεπτα (0:00.00). Για αρχεία πάνω των 25 στοιχείων για τον λόγο ότι τα αρχεία απαιτούσαν πάνω από 20 λεπτά για να τερματίσουν, ο επιλυτής MiniZinc τέθηκε να τερματίζει στα 600 δευτερόλεπτα να σταματά να τρέχει και κάνοντας χρήση της εντολής –all-solutions , όπου η εντολή αυτή καθώς ο επιλυτής αναζητά βέλτιστη λύση φαίνονται στην οθόνη οι λύσεις που συνεχώς βελτιώνονται μέχρι να βρεθεί η βέλτιστη και έτσι πάρθηκε στα 10 λεπτά το κόστος της τελευταίας λύσης που βρέθηκε. Στον πίνακα οι περιπτώσεις αυτές απεικονίζονται με - / και σε παρένθεση το κόστος της μέχρι τότε βέλτιστης λύσης, ώστε να μπορούν να συγκριθούν.

ΑΡΧΕΙΑ	Coher_set2.mzn	Finaltest2.mzn	Coher.mzn
5a.dzn	0:00.07	0:00.08	0:00.06
5b.dzn	0:00.07	0:00.07	0:00.06
5c.dzn	0:00.07	0:00.(	0:00.05
5d.dzn	0:00.07	0:00.06	0:00.06
5e.dzn	0:00.13	0:00.06	0:00.06

6a.dzn	0:00.09	0:00.07	50
6b.dzn	0:00.09	0:00.07	55
6c.dzn	0:00.09	0:00.07	50
6d.dzn	0:00.09	0:00.08	45
6e.dzn	0:00.09	0:00.07	53
7a.dzn	0:00.11	0:00.09	0:00.08
7b.dzn	0:00.11	0:00.08	0:00.08
7c.dzn	0:00.10	0:00.08	0:00.08
7d.dzn	0:00.10	0:00.08	0:00.08
7e.dzn	0:00.10	0:00.09	0:00.07
8a.dzn	0:00.13	0:00.10	0:00.09
8b.dzn	0:00.12	0:00.10	0:00.09
8c.dzn	0:00.14	0:00.11	0:00.09
8d.dzn	0:00.14	0:00.10	0:00.10
8e.dzn	0:00.13	0:00.11	0:00.10
9a.dzn	0:00.16	0:00.12	0:00.10
9b.dzn	0:00.16	0:00.11	0:00.10
9c.dzn	0:00.17	0:00.12	0:00.11
9d.dzn	0:00.17	0:00.12	0:00.11
9e.dzn	0:00.17	0:00.13	0:00.11
10a.dzn	0:00.23	0:00.17	0:00.15
10b.dzn	0:00.25	0:00.18	0:00.15
10c.dzn	0:00.23	0:00.17	0:00.14
10d.dzn	0:00.24	0:00.18	0:00.15
10e.dzn	0:00.23	0:00.17	0:00.14
11a.dzn	0:00.30	0:00.25	0:00.18
11b.dzn	0:00.29	0:00.24	0:00.17
11c.dzn	0:00.27	0:00.21	0:00.16
11d.dzn	0:00.29	0:00.23	0:00.17
11e.dzn	0:00.30	0:00.25	0:00.18
12a.dzn	0:00.41	0:00.40	0:00.25
12b.dzn	0:00.39	0:00.34	0:00.23
12c.dzn	0:00.44	0:00.43	0:00.27
12d.dzn	0:00.36	0:00.31	0:00.21
12e.dzn	0:00.48	0:00.49	0:00.30
13a.dzn	0:00.50	0:00.48	0:00.30
13b.dzn	0:00.75	0:00.83	0:00.45
13c.dzn	0:00.51	0:00.49	0:00.30
13d.dzn	0:00.54	0:00.53	0:00.32
13e.dzn	0:00.53	0:00.51	0:00.32
14a.dzn	0:00.66	0:00.65	0:00.39
14b.dzn	0:00.88	0:00.94	0:00.52
14c.dzn	0:00.89	0:00.98	0:00.53
14d.dzn	0:00.98	0:01.09	0:00.59
14e.dzn	0:00.91	0:01.07	0:00.59
15a.dzn	0:01.88	0:02.42	0:01.21
15b.dzn	0:01.13	0:01.42	0:00.71
15c.dzn	0:02.26	0:03.10	0:01.46
15d.dzn	0:01.84	0:02.42	0:01.21
15e.dzn	0:02.31	0:03.18	0:01.55
16a.dzn	0:02.73	0:03.68	0:01.71
16b.dzn	0:03.44	0:04.61	0:02.17
16c.dzn	0:03.66	0:04.81	0:02.21
16d.dzn	0:02.96	0:03.97	0:01.82
16e.dzn	0:02.46	0:03.26	0:01.50
17a.dzn	0:06.60	0:08.82	0:03.95
17b.dzn	0:05.46	0:07.13	0:03.26

17c.dzn	0:07.81	0:10.59	0:04.75
17d.dzn	0:05.48	0:07.58	0:03.35
17e.dzn	0:04.15	0:05.32	0:02.53
18a.dzn	0:08.38	0:12.77	0:05.50
18b.dzn	0:11.60	0:17.99	0:07.83
18c.dzn	0:07.66	0:11.14	0:04.97
18d.dzn	0:10.88	0:14.84	0:06.91
18e.dzn	0:08.40	0:12.80	0:05.67
19a.dzn	0:30.69	0:41.35	0:14.96
19b.dzn	0:39.89	0:50.99	0:18.52
19c.dzn	0:26.04	0:34.89	0:12.73
19d.dzn	0:34.23	0:43.51	0:16.40
19e.dzn	0:20.45	0:28.24	0:10.23
20a.dzn	0:42.57	0:57.97	0:23.48
20b.dzn	1:04.61	1:31.53	0:36.43
20c.dzn	0:49.49	0:56.29	0:21.82
20d.dzn	1:16.28	1:45.97	0:43.50
20e.dzn	0:35.54	0:50.46	0:19.38
21a.dzn	1:09.65	1:41.55	0:43.52
21b.dzn	1:00.41	1:28.82	0:38.50
21c.dzn	1:34.30	2:16.70	1:00.48
21d.dzn	0:48.01	1:10.70	0:29.49
21e.dzn	1:39.33	2:27.13	1:04.89
22a.dzn	1:44.58	2:37.51	1:14.74
22b.dzn	1:40.21	2:34.18	1:04.44
22c.dzn	3:11.30	4:33.99	2:13.94
22d.dzn	2:54.37	4:07.63	2:02.72
22e.dzn	2:56.44	4:12.04	2:08.65
23a.dzn	6:17.40	8:53.018	5:01.80
23b.dzn	6:22.958	8:42.66	5:20.659
23c.dzn	7:29.575	10:00.666	6:24.647
23d.dzn	6:37.642	9:16.186	6:22.383
23e.dzn	5:41.581	8:02.642	4:41.149
25a.dzn	-/(kostos=504)	-/(kostos=504)	-/(kostos=489)
25b.dzn	-/(kostos=440)	-/(kostos=440)	-/(kostos=440)
25c.dzn	-/(kostos=476)	-/(kostos=476)	-/(kostos=476)
25d.dzn	-/(kostos=446)	-/(kostos=446)	-/(kostos=446)
25.e.dzn	-/(kostos=481)	-/(kostos=492)	-/(kostos=481)
27a.dzn	-/(kostos=584)	-/(kostos=505)	-/(kostos=584)
27b.dzn	-/(kostos=541)	-/(kostos=541)	-/ (kostos=541)
27c.dzn	-/(kostos=515)	-/(kostos=536)	-/(kostos=489)
27d.dzn	-/(kostos=577)	-/(kostos=577)	-/(kostos=577)
27.e.dzn	-/(kostos=604)	-/(kostos=618)	-/(kostos=604)
29a.dzn	-/(kostos=614)	-/(kostos=614)	-/(kostos=614)
29b.dzn	-/(kostos=618)	-/(kostos=618)	-/(kostos=618)
29c.dzn	-/(kostos=616)	-/(kostos=616)	-/(kostos=608)
29d.dzn	-/(kostos=673)	-/(kostos=673)	-/(kostos=668)
29e.dzn	-/(kostos=639)	-/(kostos=639)	-/(kostos=639)

Πίνακας 5.1: Χρόνοι εύρεσης βέλτιστης λύσης τριών μοντέλων στον MiniZinc

Τα συμπεράσματα που προκύπτουν από το πιο πάνω πείραμα είναι τα εξής:

Σύμφωνα με το πρώτο πείραμα που έγινε στην παρούσα διπλωματική, το οποίο αφορούσε τρία διαφορετικά μοντέλα υλοποιημένα στην γλώσσα MiniZinc και τρέχοντάς τα στον επιλυτή MiniZinc που έχει περιγραφεί πιο πάνω παρατηρείται ότι συγκριτικά το πιο γρήγορο μοντέλο στην αναζήτηση της βέλτιστης λύσης είναι το μοντέλο coher.dzn το οποίο είναι υλοποιημένο με την πιο απλή διατύπωση περιορισμών, με συνεπαγωγές.

Συγκριτικά με τα άλλα δύο μοντέλα το μοντέλο αυτό τερμάτισε δίνοντας την βέλτιστη λύση στα περισσότερα από τα 125 παραδείγματα στο λιγότερο χρόνο. Ακόμα και σε μεγάλο αριθμό στοιχείων ο επιλυτής τέθηκε να τερματίζει στα δέκα λεπτά ο επιλυτής με το μοντέλο αυτό έβρισκε λύση με μικρότερο κόστος σε σχέση με τα άλλα δύο μοντέλα.

5.1.2 Πειράματα διαφορετικών παραμέτρων στον Bsolo

Το πείραμα αυτό έγινε με σκοπό να εξακριβωθεί με βάση το πρόβλημά μας ποια από τις παραμέτρους του επιλυτή bsolo_pb10 βρίσκει πιο γρήγορα την βέλτιστη λύση με βάση το πρόβλημα που μελετάται στην παρούσα διπλωματική, το Πρόβλημα της Συνοχής.

Για τις παραμέτρους (-b0 -l1, -b0 -l2, -b0 -l3, -b1 -l1, -b1 -l2,-b1 -l3, -b3 -l1, -b3 -l2, -b3 -l3) που ξεπερνούσαν κατά πολύ το χρόνο συγκριτικά με τις υπόλοιπες παραμέτρους και δεν πλησίαζαν την βέλτιστη λύση, γινόταν τερματισμός του επιλυτή και στον Πίνακα 5.2 δεν καταγραφόταν κάποια τιμή για τις παραμέτρους αυτές.

Στον Πίνακα 5.2 απεικονίζονται οι χρόνοι εκτέλεσης σε δευτερόλεπτα για να βρεθεί η βέλτιστη λύση και δίπλα σε παρένθεση απεικονίζεται το κόστος της βέλτιστης λύσης. Ο επιλυτής bsolo_pb10 τερματίζει από μόνος του στα 1800 δευτερόλεπτα δίνοντας την τελευταία βέλτιστη λύση που βρήκε. Πάρθηκε δείγμα πέντε αρχείων, ώστε να προκύψουν ορθά συμπεράσματα με πυκνότητα 100%, με ίδιο αριθμό στοιχείων και τα στοιχεία συσχετίζονταν μεταξύ τους με ακέραιες τιμές από -10 μέχρι 10.

Αρχεία	-b0 -l1	-b0 -l2	-b0 -l3	-b1 -l1	-b1 -l2	-b1 -l3	-b2 -l1	-b2 –l2	-b2 -l3	-b3 -l1	-b3 - l2	-b3 -l3
file25na	-----	-----	-----	-----	-----	-----	34.349s/(507)	40.9s/(507)	40.916s/(507)	-----	-----	-----
file25nb	-----	-----	-----	-----	-----	-----	21.95s/(493)	21.227s/(493)	21.172s(493)	-----	-----	-----
file25nc	-----	-----	-----	-----	-----	-----	22.452s/(502)	28.403s/(502)	40.132s/(502)	-----	-----	-----
file25nd	-----	-----	-----	-----	-----	-----	47.381s/(509)	46.75s/(509)	46.77s/(509)	-----	-----	-----
file25ne	-----	-----	-----	-----	-----	-----	36.426s/(488)	44.951s/(488)	45.115s(488)	-----	-----	-----
file26na	-----	-----	-----	-----	-----	-----	72.239/(549)	49.106(549)	80.768(549)	-----	-----	-----
file26nb	-----	-----	-----	-----	-----	-----	72.11s/(543)	99.038s/(543)	98.494s/(543)	-----	-----	-----
file26nc	-----	-----	-----	-----	-----	-----	28.63s/(550)	30.119s(550)	30.217s(550)	-----	-----	-----
file26nd	-----	-----	-----	-----	-----	-----	90.186s/(576)	84.722s/(576)	100.719s/(576)	-----	-----	-----
file26ne	-----	-----	-----	-----	-----	-----	60.775s/(558)	85.86s/(558)	85.651s(558)	-----	-----	-----
file35na	-----	-----	-----	-----	-----	-----	1800s/(1053)	1800s/(1053)	1800s/(1053)	-----	-----	-----
file35nb	-----	-----	-----	-----	-----	-----	1800s/(1052)	1800s/(1057)	1800s/(1057)	-----	-----	-----
file35nc	-----	-----	-----	-----	-----	-----	1800s/(1006)	1800s/(1004)	1800s/(1004)	-----	-----	-----
file35nd	-----	-----	-----	-----	-----	-----	1800s/(1039)	1800s/(1040)	1800s/(1040)	-----	-----	-----
file35ne	-----	-----	-----	-----	-----	-----	1800s/(1048)	1800s/(1054)	1800s/(1054)	-----	-----	-----
file40na	-----	-----	-----	-----	-----	-----	1800s/(1456)	1800s/(1456)	1800s/(1456)	-----	-----	-----
file40nb	-----	-----	-----	-----	-----	-----	1800s/(1455)	1800s/(1455)	1800s/(1456)	-----	-----	-----
file40nc	-----	-----	-----	-----	-----	-----	1800s/(1462)	1800s/(1462)	1800s/(1462)	-----	-----	-----
file40nd	-----	-----	-----	-----	-----	-----	1800s/(1482)	1800s/(1482)	1800s/(1482)	-----	-----	-----
file40ne	-----	-----	-----	-----	-----	-----	1800s/(1487)	1800s/(1487)	1800s/(1487)	-----	-----	-----
file45na	-----	-----	-----	-----	-----	-----	1800s/(1945)	1800s/(1945)	1800s/(1945)	-----	-----	-----
file45nb	-----	-----	-----	-----	-----	-----	1800s/(1928)	1800s/(1928)	1800s/(1928)	-----	-----	-----
file50na	-----	-----	-----	-----	-----	-----	1800s/(2532)	1800s/(2532)	1800s/(2532)	-----	-----	-----
file50nb	-----	-----	-----	-----	-----	-----	1800s/(2542)	1800s/(2542)	1800s/(2542)	-----	-----	-----

Πίνακας 5.2: Χρόνοι εύρεσης βέλτιστης λύσης BSOLO_PB10 με παραμέτρους

Τα συμπεράσματα που προκύπτουν από το πιο πάνω πείραμα είναι τα εξής:

Στο πιο πάνω πείραμα παρατηρείται ότι στο Πρόβλημα της Συνοχής σε Γράφους Συνοχής μέσω τεχνικών ικανοποίησης περιορισμών καλώντας τον επιλυτή BSOLO με την παράμετρο b2-l1 πετυχαίνει εύρεση της βέλτιστης λύσης πιο γρήγορα σε σχέση με

τις υπόλοιπες παραμέτρους. Στον Πίνακα 5.2 φαίνονται οι περιπτώσεις όπου με χρήση συνδυασμού των παραμέτρων -b2 και -l1, επιτυγχάνεται σε αποδοτικό χρόνο η εύρεση της βέλτιστης λύσης παρά με οποιοδήποτε άλλο συνδυασμό παραμέτρων. Όπως έχουν εξηγηθεί και πιο πάνω η παράμετρος -b2 αποσκοπεί στην δυναμική χρήση του LPR κάτω φράγματος. Ενώ η παράμερος -l1 γίνεται μάθηση ενός καινούριου clause. Και οι δύο αυτές παράμετροι φαίνεται να βοηθούν τον επιλυτή στην αναζήτηση της βέλτιστης λύσης πολύ γρήγορα.

Η χρήση του κατώτερου ορίου παίζει σημαντικό ρόλο στην αποδοτικότητα του αλγορίθμου. Ο λόγος που συμβαίνει αυτό είναι επειδή αν σε κάποιο σημείο ο επιλυτής βρει λύση όπου η εκτίμηση της τιμής του κατώτατου ορίου προκύπτει μεγαλύτερη ή ίση από την τιμή του άνω ορίου τότε κατά τη διάρκεια της αναζήτησης καλύτερης λύσης μπορεί να αγνοηθεί (κλαδευτεί) ο συγκεκριμένος κόμβος καθώς δεν μπορεί να βελτιωθεί με την τρέχουσα ανάθεση τιμών. Υπάρχουν πάρα πολλές τεχνικές οι οποίες μπορούν να χρησιμοποιηθούν για την εκτίμηση της τιμής του κατώτατου ορίου και πολλές από αυτές χρησιμοποιεί και ο αλγόριθμος BSOLO. Η πιο αποδοτική στο Πρόβλημα της Συνοχής φαίνεται να είναι η τεχνική χαλάρωσης γραμμικού προγραμματισμού όπου λύνεται το πρόβλημα Ακέραιου Προγραμματισμού χωρίς τους περιορισμούς ακεραιότητας. Όσο αφορά την δεύτερη παράμετρο –l1 μέσα από την διαδικασία ανάλυσης των συγκρούσεων υπάρχει η δυνατότητα μάθησης ενός καινούριου ψευδοδυαδικού περιορισμού. Η μάθηση των προτάσεων εξασφαλίζει ότι στη πορεία κατά την διαδικασία αναζήτησης μπορεί να αποφευχθεί η επίσκεψη σε αχρείαστα μέρη στο δέντρο αναζήτησης.

5.1.3 Πειράματα μεταξύ όλων των επιλυτών ανάλογα με την πυκνότητα

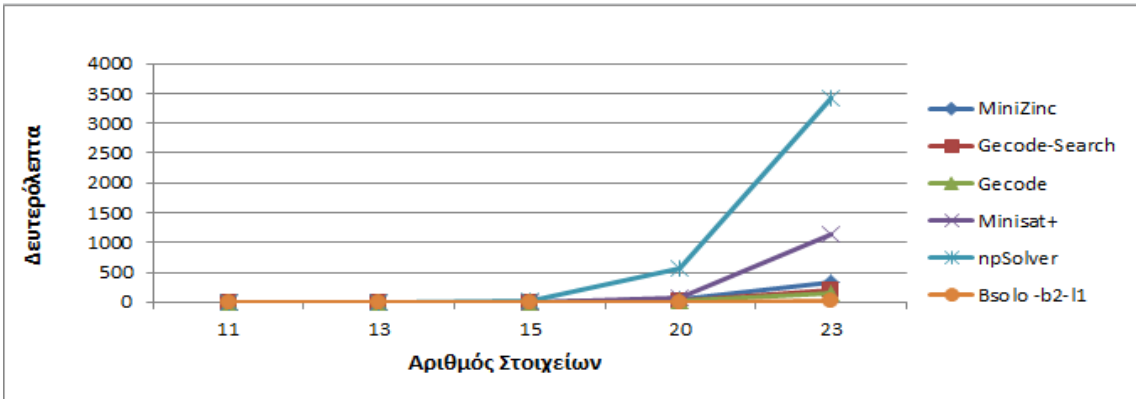
Μετά από την εύρεση του ποιο κατάλληλου μοντέλου σε MiniZinc καθώς και το συνδυασμό των πιο αποτελεσματικών παραμέτρων του BSOLO_PB10, έγιναν μετρήσεις με βάση τα ίδια δεδομένα (ίδιος αριθμός στοιχείων και πυκνότητα στοιχείων, με τυχαίες συσχετίσεις μεταξύ των στοιχείων κάθε φορά από -10 μέχρι 10) σε όλους τους επιλυτές που μελετήθηκαν στην διπλωματική αυτή ώστε να φανεί ποιος από αυτούς είναι ο πιο γρήγορος στην επίλυση του προβλήματος της συνοχής.

Στους πίνακες 5.3, 5.4, 5.5, 5.6 απεικονίζονται οι χρόνοι εκτέλεσης για να βρεθεί η βέλτιστη λύση σε δευτερόλεπτα και δίπλα σε παρένθεση απεικονίζεται το κόστος της βέλτιστης λύσης. Στους Πίνακες 5.3, 5.4, 5.5, 5.6 φαίνονται οι χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων από -10 έως 10 και πυκνότητα στοιχείων 100%, 25%, 50%, 75% αντίστοιχα.

Για την ορθότητα των συμπερασμάτων πάρθηκε δείγμα πέντε αρχείων (ίδιος αριθμός στοιχείων και πυκνότητα, με τυχαίες συσχετίσεις μεταξύ των στοιχείων κάθε φορά από -10 μέχρι 10). Ποιο εμφανές φαίνονται τα αποτελέσματα στα σχήματα 5.1,5.2,5.3,5.4 όπου για κάθε επιλυτή πάρθηκε ο μέσος χρόνος από τα πέντε δείγματα και ανάλογα με τον αριθμό των στοιχείων δημιούργησα γραφικές παραστάσεις. Να σημειωθεί επίσης ότι στην περίπτωση του npSolver στην προσπάθεια αναζήτησης λύσης κάποιες φορές διέκοπτε την αναζήτηση. Έτσι στους πίνακες θα αναγράφεται ο χρόνος που έτρεχε μέχρι να διακοπεί, το σύμβολο / και δίπλα η λέξη aborted αφού αδυνατούσε να βρει λύση.

ΑΡΧΕΙΑ	Minizinc	GECODE	Gecode-search_combinator	Minisat+	NPSOLVER	BSOLO_PB10
Filenp11.opb	0.16/66	0.03/66	0.03/66	0.050992/66	1.429s/(66)	0.046s/(66)
Filenp11a.opb	0.20/74	0.05/74	0.04/74	0.124981/74	1.981s/(74)	0.066s/(74)
Filenp11b.opb	0.19/73	0.04/73	0.05/73	0.077988/73	1.347s/(73)	0.049s/(73)
Filenp11d.opb	0.17/80	0.04/80	0.14/80	0.044993/80	1.390s/(80)	0.03s/(80)
Filenp11e.opb	0.18/75	0.04/75	0.15/75	0.045993/75	1.500s/(75)	0.04s/(80)
Filenp13.opb	0.31/90	0.09/90	0.11/90	0.156976/90	0.400s/(90)	0.08s/(90)
Filenp13b.opb	0.39/101	0.15/101	0.16/101	0.156976/101	2.031s/(101)	0.086s/(101)
Filenp13c.opb	0.42/127	0.16/127	0.14/127	0.255961/127	3.289s/(127)	0.117s/(127)
Filenp13d.opb	0.42/108	0.16/108	0.16 /108	0.231964/108	17.127s/(108)	0.08s/(108)
Filenp13e.opb	0.40/100	0.15/100	0.14 /100	0.245961/100	3.567s/(100)	0.08s/(108)
Filenp15.opb	1.08/156	0.54/156	0.54/156	1.73374/156	27.405s/(156)	0.215s/(156)
Filenp15a.opb	0.99/150	0.48/150	0.53/150	0.466929/150	6.001s/(150)	0.218s/(150)
Filenp15b.opb	1.12/157	0.53/157	0.52/157	0.713891/157	9.426s/(157)	0.195s/(157)
Filenp15c.opb	0.77/149	0.37/149	1.86/149	1.3228/149	54.512s/(149)	0.134s/(149)
Filenp15d.opb	0.81/159	0.38/159	0.54/159	0.85287/159	4.011s/(159)	0.293s/(159)
Filenp20.opb	29.31/298	18.92/298	20.30/298	38.4791/298	372,77s/(298)	2.466s/(298)
Filenp20a.opb	28.42/309	18.45/309	0.54/309	146.377/309	386,78s(aborted)	3.129s/(309)
Filenp20b.opb	41.76/306	25.79/306	24.11/306	69.3525/306	126,05 s/(aborted)	3.927s/(306)
Filenp20c.opb	32.44/323	20.98/323	20.92/323	31.6992/323	364,30s/(323)	2.59s/(323)
Filenp20d.opb	34.90/315	21.57/315	22.40/315	48.8106/315	946,75s /(315)	4.689s/(315)
Filenp23.opb	326,03/410	153,49/410	168,44/410	413.039s/410	19:31.296s/(aborted)	15.437s/(410)
Filenp23a.opb	317,08/417	154,48/417	214,33/417	1183.4/417	21:54.234s/(aborted)	18.191s/(417)
Filenp23b.opb	361,19/438	172,46/438	214,18/438	1515.47s/438	8:56.368s/(aborted)	22.249s/(438)
Filenp23c.opb	290,06/436	139,95/436	182,84/436	1381.51s/436	11:57.899s/(aborted)	15.925s(436)
Filenp23d.opb	331,39/418	146,98/418	170,47/418	774.797s/418	3,431,64s /(418)	8.526s/(418)

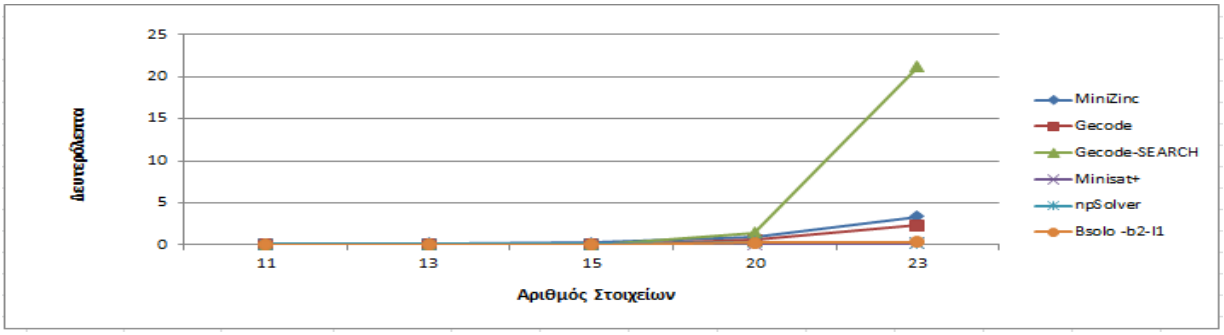
Πίνακας 5.3 : Χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων με -10 έως 10 και πυκνότητα στοιχείων 100%



Σχήμα 5.1: Μέσος χρόνος δειγμάτων από πίνακα 5.3. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των στοιχείων και ο κάθετος άξονας αναπαριστά το χρόνο σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης.

ΑΡΧΕΙΑ	Minizinc	Gecode-	GECODE search_combinator	Minisat+	NPSOLVER	BSOLO_PB10
Filenp11.opb	0.15/12	0:00.02/12	0:00.02/12	0.001999/12	0m0.086s/12	0.011s/12
Filenp11a.opb	0.13/4	0:00.01/4	0:00.01/4	0.000000/4	0m0.067s/4	0.004s/4
Filenp11b.opb	0.13/10	0:00.01/10	0:00.02/10	0.001999/10	0m0.091s/10	0.017s/10
Filenp11d.opb	0.13/9	0:00.02/9	0:00.02/9	0.000999/9	0m0.073s/9	0.007s/9
Filenp11e.opb	0.12/8	0:00.02/8	0:00.02/8	0.000999/8	0m0.0652s/8	0.005s/8
Filenp13.opb	0.17/6	0.02/6	0.02/6	0.001999/6	0.079s/6	0.02s/6
Filenp13b.opb	0.18/10	0.02/10	0.03/10	0.004999/10	0.097s/10	0.021s/10
Filenp13c.opb	0.20/16	0.03/16	0.04/16	0.004999/16	0.094s/16	0.031s/16
Filenp13d.opb	0.18/16	0.03/16	0.05/16	0.00399/16	0.094s/16	0.023s/16
Filenp13e.opb	0.16/5	0.02/5	0.03/5	0.00299/5	0.085s/5	0.022s/5
Filenp15a.opb	0.29/6	0.05/6	0.03/6	0.004999/6	0.095s/6	0.007s/6
Filenp15b.opb	0.25/19	0.03/19	0.11/19	0.008998/19	0.146s/19	0.04s/19
Filenp15c.opb	0.26/14	0.05/14	0.06/14	0.004999/14	0.082s/14	0.03s/14
Filenp15d.opb	0.28/8	0.04/8	00.03/8	0.004999/8	0.107s/8	0.019s/8
Filenp15e.opb	0.24/7	00.02/7	0.01/7	0.005999/7	0.125s/7	0.099s/7
Filenp20a.opb	1.27/40	0.82/40	1.73/40	0.032994/40	0.183s/40	0.283s/40
Filenp20b.opb	0.70/33	0.28/33	1.62/33	0.073988/33	0.243s/33	0.33s/33
Filenp20c.opb	1.54/42	1.21/42	1.56/42	0.068989/42	0.614s/42	0.216s/42
Filenp20d.opb	0.70/42	0.29/42	1.26/42	0.030995/42	0.350s/42	0.227s/42
Filenp20e.opb	0.60/41	0.27/41	1.23/41	0.020955/41	0.250s/41	0.23s/41
Filenp23a.opb	4.12/62	2.99/62	17.52/62	0.308953/62	0.313s/62	0.771s/62
Filenp23b.opb	3.95/55	3.35/55	14.31/55	0.092985/55	0.239s/55	0.371s/55
Filenp23c.opb	2.27/65	1.30/65	31.10/65	0.151976/65	0.379s/65	0.565s/65
Filenp23d.opb	3.84/61	2.60/61	22.10/61	0.277957/61	0.345s/61	0.08s/61
Filenp23e.opb	2.85/63	1.60/63	21.10/63	0.217157/63	0.243s/63	0.07s/63

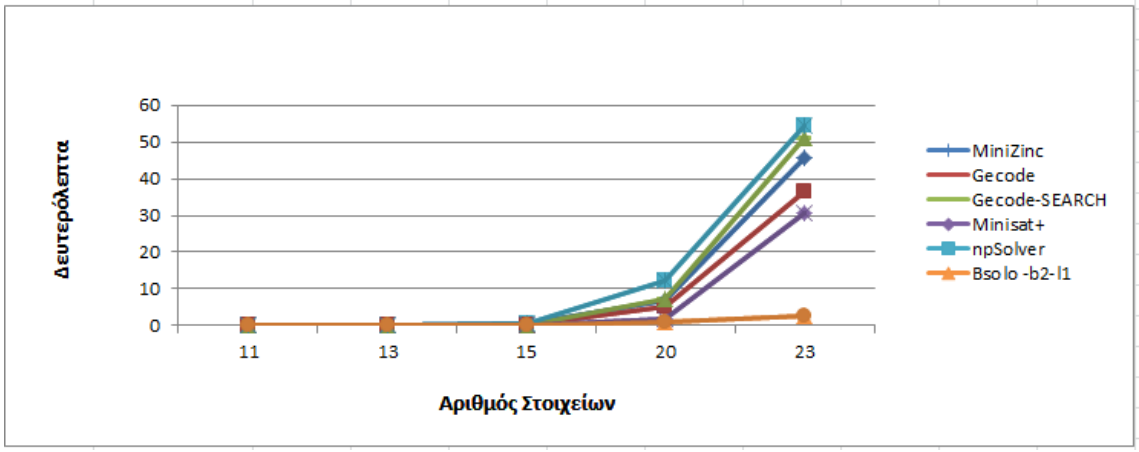
Πίνακας 5.4 : Χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων με -10 έως 10 και πυκνότητα στοιχείων 25%



Σχήμα 5.2: Μέσος χρόνος δειγμάτων από πίνακα 5.4. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των στοιχείων και ο κάθετος άξονας αναπαριστά το χρόνο σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης.

ΑΡΧΕΙΑ	Minizinc	Gecode-	GECODE search combinator	Minisat+	NPSOLVER	BSOLO_PB10
Filenp11.opb	0.15/26	0.03/26	0.06/26	0.005999/26	0.118/26	0.019s/26
Filenp11a.opb	0.14/25	0.02/25	0.02/25	0.002999/25	0.154/25	0.014s/25
Filenp11b.opb	0.15/27	0.03/27	0.02/27	0.016997/27	0.145/27	0.025s/27
Filenp11d.opb	0.14/25	0.02/25	0.02/25	0.014997/25	0.103/25	0.011s/25
Filenp11e.opb	0.13/23	0.02/23	0.02/23	0.013997/23	0.102/23	0.010s/23
Filenp13.opb	0.19/36	0.04/36	0.05/36	0.040993/36	0.136/36	0.071s/36
Filenp13b.opb	0.23/36	0.05/36	0.05/36	0.073988/36	0.137/36	0.057s/36
Filenp13c.opb	0.24/36	0.00/36	0.05/36	0.031995/36	0.229/36	0.019s/36
Filenp13d.opb	0.22/32	0.05/32	0.06/32	0.01996/32	0.187/32	0.021s/32
Filenp13e.opb	0.21/31	0.05/31	0.04/31	0.01895/31	0.177/31	0.021s/31
Filenp15a.opb	0.43/48	0.17/48	0.16/48	0.040993/48	0.318/48	0.049s/48
Filenp15b.opb	0.50/75	0.22/75	0.33/75	0.280957/75	2.273/75	0.118s/75
Filenp15c.opb	0.57/50	0.27/50	0.16/50	0.144977/50	0.507/50	0.073s/50
Filenp15d.opb	0.57/74	0.27/74	0.26/74	0.167974/74	0.0920/74	0.134s/74
Filenp15e.opb	0.55/62	0.25/62	0.24/62	0.155973/62	0.085/62	0.133s/62
Filenp20a.opb	3.17/129	2.57/129	5.64/129	2.02169/129	6.468/129	0.469s/129
Filenp20b.opb	6.73/124	5.21/124	8.19/124	0.759884/124	34.424/124	0.746/124
Filenp20c.opb	4.84/116	3.94/116	5.94/116	1.46678/116	8.395/116	0.987s/116
Filenp20d.opb	11.57/125	9.03/125	10.36/125	2.87856/125	7.970/125	1.192s/125
Filenp20e.opb	7.57/123	5.03/123	6.35/123	2.786/123	4.97/123	0.992s/123
Filenp23a.opb	32.82/154	25.59/154	45.73/154	22.9065/154	45.136/154	1.542s/154
Filenp23b.opb	47.41/172	32.84/172	59.94/172	75.5825/172	56.471/172	2.259s/172
Filenp23c.opb	21.57/138	16.45/138	35.69/138	9.54955/138	24.050/138	2.041s/138
Filenp23d.opb	68,12 /176	45.01/176	67,12 /176	22.5406/176	81,63 /176	4.195s/176
Filenp23e.opb	58,13/140	63.02/140	46.12/140	23.530/176	65.63/140	3.184s/176

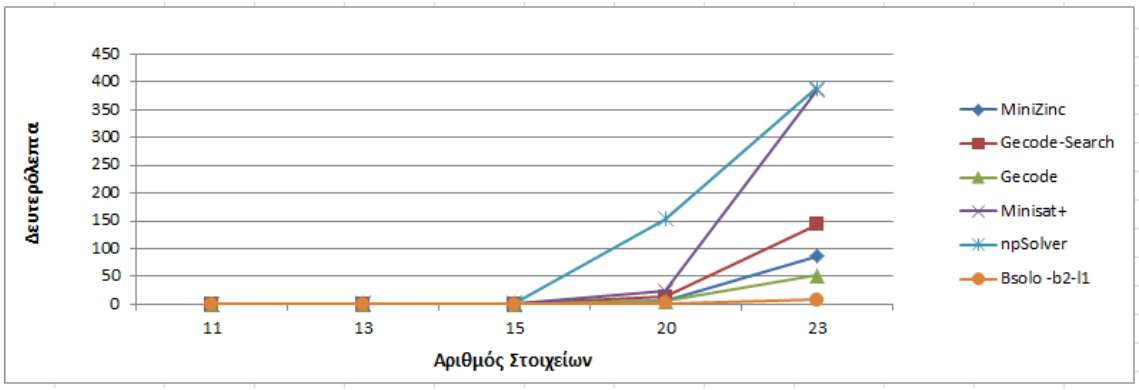
Πίνακας 5.5 : Χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων με -10 έως 10 και πυκνότητα στοιχείων 50%



Σχήμα 5.3: Μέσος χρόνος δειγμάτων από πίνακα 5.5. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των στοιχείων και ο κάθετος άξονας αναπαριστά το χρόνο σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης.

ΑΡΧΕΙΑ	Minizinc	GECODE	Gecode-search_combinator	Minisat+	NPSOLVER	BSOLO
Filenp11.opb	0.15/50	0.03/50	0.02/50	0.037994 s/50	0.126s/50	0.026s/50
Filenp11a.opb	0.15/41	0.03/41	0.02/41	0.029995 s/41	0.113s/41	0.017s/41
Filenp11b.opb	0.14/41	0.02/41	0.02/41	0.032994 s/41	0.148s/41	0.025s/41
Filenp11d.opb	0.15/39	0.03/39	0.02/39	0.017997 s/39	0.127s/39	0.014s/39
Filenp11e.opb	0.14/44	0.02/44	0.02/44	0.037994 s/44	0.248s/44	0.019s/44
Filenp13.opb	0.20/45	0.05/45	0.07/45	0.023996 s/45	0.129s/45	0.055s/45
Filenp13b.opb	0.29/73	0.10/73	0.10/73	0.133979 s/73	0.470s/73	0.114s/73
Filenp13c.opb	0.45/79	0.18/79	0.11/79	0.12698 s/79	0.372s/79	0.104s/79
Filenp13d.opb	0.29/67	0.10/67	0.09/67	0.084987 s/67	0.256s/67	0.112s/67
Filenp13e.opb	0.27/75	0.09/75	0.11/75	0.139978 s/75	5.008s/75	0.054s/75
Filenp15a.opb	0.65/100	0.30/100	0.36/100	0.216967 s/100	0.453s/100	0.105s/100
Filenp15b.opb	0.53/91	0.23/91	0.29/91	0.336948 s/91	0.544s/91	0.106s/91
Filenp15c.opb	0.44/85	0.17/85	0.30/85	0.218966 s/85	0.394s/85	0.085s/85
Filenp15d.opb	0.43/96	0.17/96	0.26/96	0.485926 s/96	3.792s/96	0.147s/96
Filenp15e.opb	0.53/123	0.23/123	0.45/123	0.404938 s/123	0.773s/123	0.168s/123
Filenp20a.opb	7.27/206	5.19/206	15.98/206	38.1712 s/206	95.43 s/206	2.26s/206
Filenp20b.opb	9.74/225	7.15/225	13.26/225	40.6348 s/225	234.64 s/225	1.02/225
Filenp20c.opb	5.44/194	4.09/194	7.63/194	13.32 s/194	134.38 /194	1.217s/194
Filenp20d.opb	6.72/200	4.90/200	11.95/200	10.6844 s/200	45.585s/200	0.778s/200
Filenp20e.opb	6.37/213	4.96/213	15.58/213	12.7121 s/213	255.47 s/213	1.599s/213
Filenp23a.opb	75.01/320	46.20/320	164.16/320	736.241 s/320	10:3.974s/aborted	10.099s/320
Filenp23b.opb	107.27/297	64.37 /297	153.71/297	390.846 s/297	7:58.696s/aborted	11.647s/297
Filenp23c.opb	61.91/306	38.27/306	107.88/306	126.417 s/306	4:5.138s/aborted	6.042s/306
Filenp23d.opb	126.79/319	70.76 /319	158.15/319	369.052 s/319	2:17.969s/587	6.727s/319
Filenp23e.opb	63.42/296	41.04/296	129.83/296	303.36 s /296	7:59.803s/296	4.068s/296

Πίνακας 5.6 : Χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων με -10 έως 10 και πυκνότητα στοιχείων 75%



Σχήμα 5.4: Μέσος χρόνος δειγμάτων από Πίνακα 5.6. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των στοιχείων και ο κάθετος άξονας αναπαριστά το χρόνο σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης.

.

Για να είναι πιο εμφανές τα αποτελέσματα δημιούργησα γραφικές παραστάσεις για το μέσο όρο χρόνου που απαιτεί κάθε επιλυτής ξεχωριστά στην εύρεση της βέλτιστης λύσης ανάλογα με την πυκνότητα των στοιχείων. Στο Σχήμα 5.5 και 5.6 φαίνονται για κάθε επιλυτή ξεχωριστά ο μέσος όρος χρόνου (πέντε δειγμάτων) ανάλογα με τα στοιχεία που απαιτούν για εύρεση της βέλτιστης λύσης σε δευτερόλεπτα με διαφορετική πυκνότητα (0.25,0.5,0.75,1) στοιχείων.

	MiniZinc/0.25	MiniZinc/0.5	MiniZinc/0.75	MiniZinc/1
11	0,132	0,142	0,146	0,1800
13	0,178	0,218	0,3	0,3880
15	0,264	0,524	0,516	0,9540
20	0,962	6,776	7,108	33,3660
23	3,406	45,61	86,88	325,0522

	GECODE_sea/0.25	GECODE_sea/0.5	GECODE_sea/0.75	GECODE_sea/1
11	0,018	0,028	0,02	0,0820
13	0,034	0,05	0,096	0,1400
15	0,048	0,23	0,332	0,7980
20	1,48	7,298	12,88	17,6540
23	21,226	50,92	142,746	190,0520

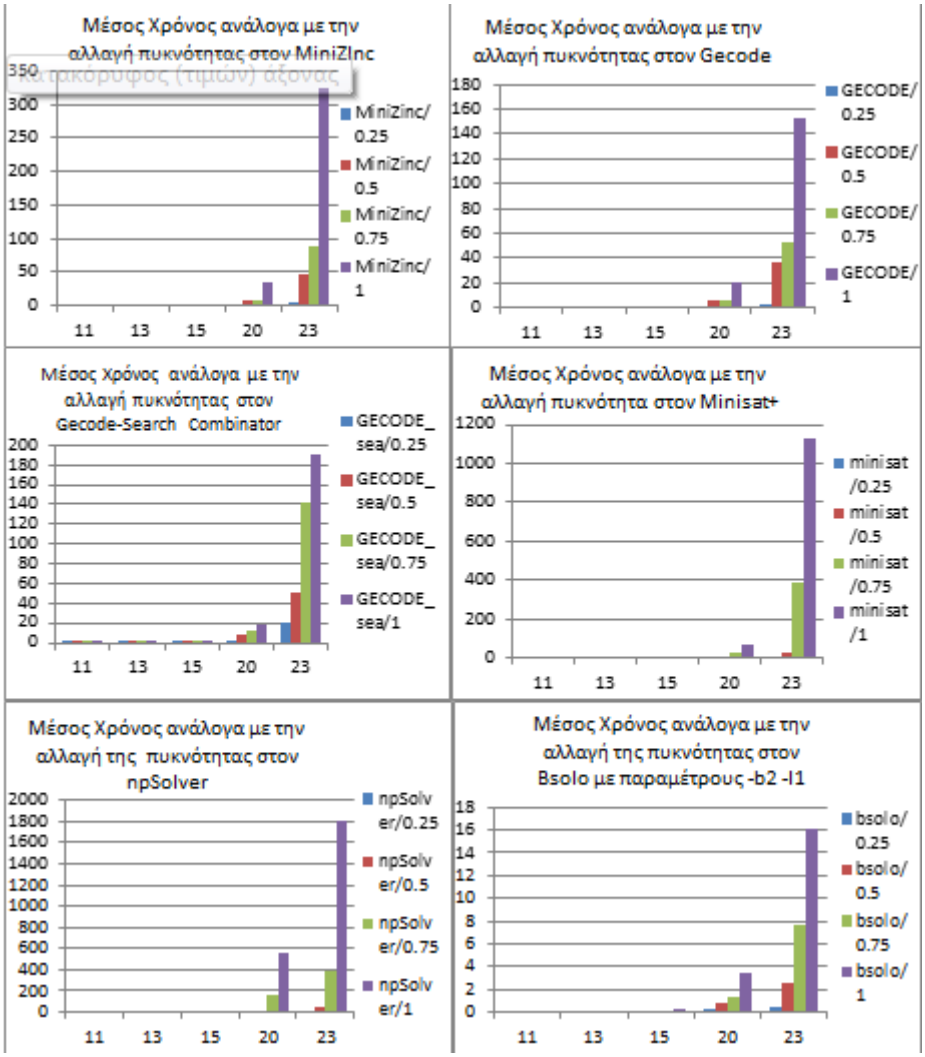
	GECODE/0.25	GECODE/0.5	GECODE/0.75	GECODE/1
11	0,016	0,024	0,026	0,0400
13	0,024	0,038	0,104	0,1420
15	0,038	0,236	0,22	0,4600
20	0,574	5,156	5,258	21,1420
23	2,368	36,582	52,128	153,4720

	minisat/0.25	minisat/0.5	minisat/0.75	minisat/1
11	0,0011992	0,0109978	0,0313948	0,0689894
13	0,0037954	0,0371772	0,101984	0,2095676
15	0,0059988	0,1581748	0,332749	1,018046
20	0,0455842	1,9825028	23,1045	70,10641111
23	0,2098056	30,82183	385,1832	1124,821444

	npSolver/0.25	npSolver/0.5	npSolver/0.75	npSolver/1
11	0,07644	0,1244	0,1524	1,5294
13	0,0898	0,1732	1,247	5,2828
15	0,111	0,655	1,1912	20,271
20	0,328	12,4454	153,101	561,2733
23	0,3038	54,5834	389,116	1800

	bsolo/0.25	bsolo/0.5	bsolo/0.75	bsolo/1
11	0,0088	0,0158	0,0202	0,0462
13	0,0234	0,0376	0,0878	0,0906
15	0,021	0,1014	0,1222	0,211
20	0,2572	0,8772	1,3748	3,459555556
23	0,3714	2,6442	7,7166	16,13544444

Σχήμα 5.5: Απεικόνιση μέσου όρου του χρόνου σε δευτερόλεπτα για εύρεση βέλτιστης λύσης από τον κάθε επιλυτή ανάλογα με τα στοιχεία και την πυκνότητα των στοιχείων.



Σχήμα 5.6: Στο πιο πάνω σχήμα φαίνεται για κάθε επιλυτή πως επηρεάζεται ο χρόνος εύρεσης βέλτιστης λύσης με την αλλαγή της πυκνότητας των στοιχείων. Στον οριζόντιο άξονα είναι ο αριθμός των στοιχείων και στον κάθετο άξονα είναι ο μέσος όρος του χρόνου από τα πέντε δείγματα στον κάθε επιλυτή.

Τα συμπεράσματα από τα πιο πάνω πειράματα που προκύπτουν είναι τα εξής:

Τα πιο πάνω πειράματα που έγιναν είχαν δύο βασικούς στόχους. Ο πρώτος στόχος ήταν να διαπιστωθεί ποιος είναι ο πιο γρήγορος επιλυτής στο να βρίσκει βέλτιστη λύση για το Πρόβλημα της Συνοχής σε Γράφους Συνοχής, ενώ ο δεύτερος στόχος ήταν να δούμε πως η αύξηση της πυκνότητας των στοιχείων επηρεάζει τον χρόνο εύρεσης της βέλτιστης λύσης. Από τα σχήματα 5.1, 5.2, 5.3, 5.4 φαίνεται ξεκάθαρα πως για μικρό αριθμό στοιχείων οι επιλυτές βρίσκουν όλοι την βέλτιστη λύση σε ελάχιστο χρόνο. Για μεγάλο αριθμό στοιχείων φαίνεται πως ο γρηγορότερος επιλυτής είναι ο bsolo. Οι πιο

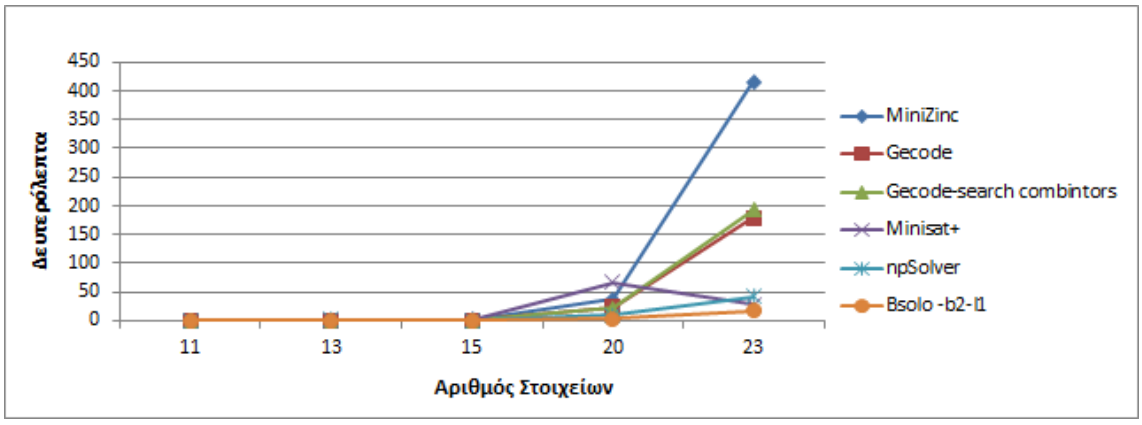
αργός επιλυτής είναι κατά μέσο όρο ο npSolver ο οποίος για μεγάλο αριθμό στοιχείων και μεγάλο ποσοστό πυκνότητας στοιχείων αποτυγχάνει να βρει την βέλτιστη λύση (aborted) τις περισσότερες φορές. Όσο αφορά τους επιλυτές MiniZinc, Gecode, Gecode με search combinator οι οποίοι είναι επιλυτές σε προβλήματα ικανοποίησης περιορισμών ο πιο γρήγορος από τους τρείς στο Πρόβλημα της Συνοχής είναι ο Gecode ο οποίος τρέχει με το ίδιο μοντέλο που τρέχει και ο MiniZinc. Ακολουθεί ο επιλυτής MiniZinc ενώ πιο αργός επιλυτής για μεγάλο αριθμό στοιχείων είναι ο Gecode ο οποίος τρέχει με το μοντέλο το οποίο χρησιμοποιεί συνδυαστές αναζήτησης. Ο συνδυασμός συνδυαστών αναζήτησης που χρησιμοποιείται στο μοντέλο αυτό καθιστά τον επιλυτή πιο αργό σε σχέση με το μοντέλο που τρέχει στον Gecode και MiniZinc το οποίο ενσωματώνει μόνο ένα ευρετικό αναζήτησης. Η ευρετική αναζήτηση κάνει συχνά τη διαφορά στο να επιλύει αποτελεσματικά και με επιτυχία ένα πρόβλημα ικανοποίησης περιορισμών. Επιπρόσθετα η χρήση συνδυαστών αναζήτησης είναι πιο παραγωγική από την προσφυγή σε μια γενικής χρήσης γλώσσα, όμως το να προσθέτεις ένα περισσότερο συνδυαστή αναζήτησης επιφέρει ένα μικρό κόστος ως προς το χρόνο, όπως έχει διαπιστωθεί και στα πειράματα που έγιναν όπου ο χρόνος επηρεάστηκε ως προς το χειρότερο. Αξίζει να σημειωθεί πως για μικρό αριθμό στοιχείων ο συνδυασμός του Gecode με το συγκεκριμένο συνδυασμό ευρετικών αναζήτησης (bab-branch and bound μαζί με άλλους συνδυαστές αναζήτησης) είναι πολύ πιο αποδοτικός από τον MiniZinc.
Επιπρόσθετα ακόμα ένα συμπέρασμα που προκύπτει από τα σχήματα 5.5 και 5.6 μπορεί να παρατηρηθεί ότι όσο πιο μικρή η πυκνότητα των στοιχείων τόσο πιο γρήγορα κάθε επιλυτής βρίσκει την βέλτιστη λύση. Καθώς αυξάνεται η πυκνότητα αυξάνεται και ο χρόνος αναζήτησης της βέλτιστης λύσης σε όλους τους επιλυτές.

5.1.4 Πειράματα μεταξύ όλων των επιλυτών με διαφορετικό τρόπο απεικόνισης της συσχέτισης δύο στοιχείων

Ακολούθως έγινε ακόμα ένα πείραμα. Οι συσχετίσεις στοιχείων αντί τώρα να απεικονίζονται με -10 έως 10, θα απεικονίζονται με -1 έως 1. Στον Πίνακα 5.7 και στο Σχήμα 5.7 φαίνονται οι μετρήσεις που πήρα σε όλους τους επιλυτές. Να σημειωθεί εδώ ότι η πυκνότητα μεταξύ των στοιχείων στο πείραμα αυτό ήταν 100% ώστε να συγκριθεί με τις συσχετίσεις στοιχείων -10 έως 10 με πυκνότητα 100% (πίνακας 6.1).

ΑΡΧΕΙΑ	Minizinc	Gecode-	GECODE-search_combinator	Minisat+	NPSOLVER	BSOLO_PB10
Filenp11.opb	0.20/15	0.05/15	0.03/15	0.011998/15	0.152s/15	0.04s/15
Filenp11a.opb	0.20/17	0.05/17	0.04/17	0.015997/17	0.244s/17	0.052s/17
Filenp11b.opb	0.16/13	0.04/13	0.03/13	0.015997/13	0.225s/13	0.037s/13
Filenp11d.opb	0.18/13	0.04/16	0.03/16	0.022996/16	0.235s/16	0.028s/16
Filenp11e.opb	0.17/14	0.04/14	0.03/14	0.023996/14	0.245s/14	0.029s/14
Filenp13.opb	0.42/24	0.15/24	0.15/24	0.053991/24	0.457s/24	0.13s/24
Filenp13b.opb	0.52/26	0.20/26	0.17/26	0.065989/26	2.373s/26	0.129s/26
Filenp13c.opb	0.37/23	0.14/23	0.13/23	0.072988/23	3.329s/23	0.102s/23
Filenp13d.opb	0.46/24	0.18/24	0.16/24	0.066299/24	0.406s/24	0.125s/24
Filenp13e.opb	0.41/23	0.17/23	0.16/23	0.056299/23	0.506s/23	0.135s/23
Filenp15a.opb	1.69/34	0.72/34	0.74/34	0.247962/34	0.650s/34	0.282s/34
Filenp15b.opb	1.61/34	0.76/34	0.64/34	0.221966/34	0.925s/34	0.21s/34
Filenp15c.opb	1.54/35	1.74/35	1.70/35	0.180972/35	1.379s/35	0.24s/35
Filenp15d.opb	1.51/33	0.73/33	0.64/33	0.174973/33	0.817s/33	0.282s/33
Filenp15e.opb	1.52/32	0.73/32	0.65/32	0.164973/32	0.717s/32	0.183s/33
Filenp20a.opb	34.44/64	0:21.40/64	22.31/64	146.377/64	14.545s/64	4.474s/64
Filenp20b.opb	44.24/67	0:27.42/67	27.89/67	69.3525/67	10.929s/67	3.995s/67
Filenp20c.opb	36.63/63	0:23.06/63	24.05/63	31.6992/63	9.036s/63	3.688s/63
Filenp20d.opb	33.90/62	0:21.62/62	19.67/62	48.8106/62	8.062s/62	1.85s/62
Filenp20e.opb	35.90/65	0:22.32/65	18.57/65	38.8105/65	7.063s/62	1.73s/65
Filenp23a.opb	489.6 /88	204.33 /88	201,87 /88	24.3213/88	47.091s/88	15.409s/88
Filenp23b.opb	466.1 /88	198.18 /88	211,53 /88	35.8076/88	46.870s/88	17.176s/88
Filenp23c.opb	286.1 /83	144.79 /83	165,38 /83	8.88065/83	33.529s/83	13.002s/83
Filenp23d.opb	418.51 /88	185.24 /88	219,48 /88	33.9238/88	46.744s/88	20.651s/88
Filenp23e.opb	415.31/89	165.23/89	163.47/89	35.9238/89	36.632s/89	18.431s/85

Πίνακας 5.7: Χρόνοι εκτέλεσης εύρεσης βέλτιστης λύσης στους επιλυτές με συσχετίσεις στοιχείων με -1 έως 1 και πυκνότητα στοιχείων 100%



Σχήμα 5.7: Μέσος χρόνος δειγμάτων από πίνακα 5.7. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των στοιχείων και ο κάθετος άξονας αναπαριστά το χρόνο σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης.

Για πιο εμφανή σύγκριση του τελευταίου πειράματος με διαφορετική απεικόνιση της συσχέτισης των στοιχείων, με βάση τις μετρήσεις που πήρα υπολογίζοντας τον μέσο όρο χρόνου σε δευτερόλεπτα των πέντε δειγμάτων κάθε περίπτωσης όπως φαίνονται στον Σχήμα 5.8 , τα αποτελέσματα αυτά τα απεικόνισα σε γραφικές παραστάσεις οι οποίες φαίνονται στο σχήμα 5.9.

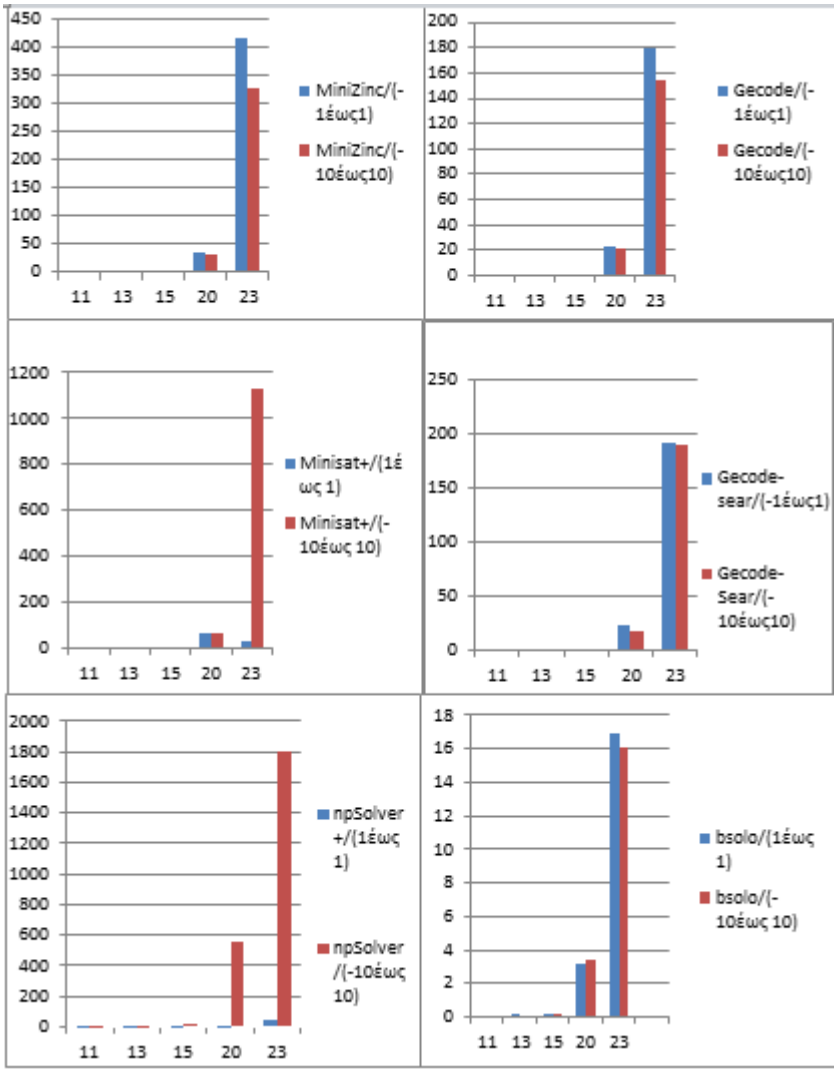
Να σημειωθεί ότι στην περίπτωση του npSolver ο οποίος στην προσπάθεια αναζήτησης λύσης κάποιες φορές διάκοπτε την αναζήτηση (aborted), για τον υπολογισμό του μέσου όρου έβαλα μια μεγαλύτερη τιμή η οποία φαίνεται και στο σχήμα 5.8 με κόκκινο ώστε να φαίνεται ότι δεν μπορούσε να βρει λύση.

	MiniZinc/(-1έως1)	MiniZinc/(-10έως10)		Gecode/(-1έως1)	Gecode/(-10έως10)
11	0,182	0,18	11	0,044	0,04
13	0,436	0,388	13	0,168	0,142
15	1,574	0,954	15	0,936	0,46
20	37,022	33,366	20	23,164	21,142
23	415,124	325,0522222	23	179,554	153,472

	Minisat+/(1έως 1)	Minisat+/(-10έως 10)		Gecode-sear/(-1έως1)	Gecode-Sear/(-10έως10)
11	0,0181968	0,0689894	11	0,032	0,082
13	0,0631132	0,2095676	13	0,154	0,14
15	0,1981692	1,018046	15	0,674	0,798
20	67,00996	70,10641111	20	22,498	17,654
23	27,77143	1124,821444	23	192,346	190,052

	npSolver+/(1έως 1)	npSolver/(-10έως 10)		bsolo/(1έως 1)	bsolo/(-10έως 10)
11	0,2202	1,5294	11	0,0372	0,0462
13	1,4142	5,2828	13	0,1242	0,0906
15	0,8976	20,271	15	0,2394	0,211
20	9,927	561,2733333	20	3,1474	3,459555556
23	42,1732	1800	23	16,9338	16,13544444

Σχήμα 5.8: Για κάθε επιλυτή ξεχωριστά φαίνονται οι μέσοι χρόνοι που προέκυψαν για εύρεση της βέλτιστης λύσης με τις δυο διαφορετικές προσεγγίσεις απεικόνισης των συσχετίσεων μεταξύ των στοιχείων (ίδιο αριθμό στοιχείων 11,13,15,20,23, ίδια πυκνότητα στοιχείων 100%, διαφορετική απεικόνιση συσχετίσεων των στοιχείων), δηλαδή με απεικόνιση συσχετίσεων δύο στοιχείων από το -1 έως 1 και με απεικόνιση συσχετίσεων δύο στοιχείων από το -10 έως 10.



Σχήμα 5.9: Ο οριζόντιος άξονας είναι ο αριθμός των στοιχείων, ο κάθετος άξονας είναι ο μέσος όρος χρόνου των πέντε δειγμάτων (ίδιος αριθμός στοιχείων και πυκνότητα) σε δευτερόλεπτα που απαιτούν οι επιλυτές για εύρεση της βέλτιστης λύσης με διαφορετικές προσεγγίσεις απεικόνισης των συσχετίσεων των στοιχείων.

Τα συμπεράσματα που προκύπτουν από το πιο πάνω πείραμα είναι τα εξής:

Στο τελευταίο πείραμα το οποίο έγινε κρατώντας την πυκνότητα των στοιχείων 100% και αλλάζοντας την απεικόνιση των συσχετίσεων των στοιχείων σε -1 έως 1 παρατηρούμε από το σχήμα 5.9 ότι στους επιλυτές Minisat+ και npSolver το πρόβλημα επιλύεται πιο γρήγορα από την απεικόνιση των συσχετίσεων των στοιχείων με -10 έως 10. Έτσι αν το σύνολο από τα στοιχεία μας, είτε συσχετίζονται μεταξύ τους 100%, είτε

δεν συσχετίζονται μεταξύ τους 100%, τότε οι καλύτεροι επιλυτές για το Πρόβλημα της Συνοχής εκτός από τον bsolo_pb10 με παραμέτρους –b2 –l1, είναι οι επιλυτές Minisat+ και npSolver .

Κεφάλαιο 6

Γενικά Συμπεράσματα και Μελλοντική Εργασία

6.1 Γενικά Συμπεράσματα Πειραμάτων.…….....78
6.2 Μελλοντική Εργασία……...79

6.1 Γενικά Συμπεράσματα

Μέσα από την μοντελοποίηση του Προβλήματος Συνοχής μέσω διάφορων τεχνικών ικανοποίησης περιορισμών στους διάφορους επιλυτές προέκυψαν κάποια συμπεράσματα.

Όπως έχει παρατηρηθεί το Πρόβλημα της Συνοχής είναι ένα θέμα που απασχολεί διαφορετικούς τομείς της επιστήμης, όπως την μεταφυσική, την επιστημολογία, καθώς και στον τομέα της λογικής και της ηθικής υπάρχουν πάρα πολλοί υποστηρικτές της θεωρίας αυτής. Επίσης οι ψυχολόγοι έχουν ασχοληθεί έντονα με την έννοια της συνοχής για την κατανόηση των διαδικασιών αντίληψης του κειμένου και αντίληψης του λόγου. Έτσι η περεταίρω γνώση του προβλήματος αυτού μπορεί να χρησιμοποιηθεί για επεξήγηση και επίλυση πολλών προβλημάτων στους τομείς αυτούς.

Μέσα από την διπλωματική αυτή διαπιστώθηκε πως η πιο απλή διατύπωση περιορισμών στον επιλυτή MiniZinc είναι πιο αποδοτική παρά με την προσφυγή σε μια πιο περίπλοκη διατύπωση όπως με διατύπωση περιορισμών σε μορφή αριθμητικών πράξεων ή με απεικόνιση περιορισμών μέσω συνόλων.

Ένα άλλο συμπέρασμα που διαπιστώθηκε μέσω της παρούσας διπλωματικής είναι ότι οι ψευδοδυαδικοί επιλυτές είναι πολύ πιο αποδοτικοί από τους επιλυτές προβλημάτων ικανοποίησης περιορισμών με ακέραιες τιμές στο πεδίο ορισμού των μεταβλητών στην εύρεση της βέλτιστης λύσης για το Πρόβλημα της Συνοχής.

Παρατηρείται επίσης πως η πυκνότητα των στοιχείων στο γράφο συνοχής επηρεάζει την αποδοτικότητα του κάθε επιλυτή. Όσο μεγαλύτερη η πυκνότητα τόσο πιο πολύ χρόνο χρειάζεται ο κάθε επιλυτής στην εύρεση της βέλτιστης λύσης για το Πρόβλημα

της Συνοχής. Επίσης όσο πιο πολλά τα στοιχεία τόσος πιο πολύς είναι και ο χρόνος που απαιτεί ο κάθε επιλυτής στην εύρεση της βέλτιστης λύσης για το Πρόβλημα της Συνοχής.

Επιπρόσθετα είναι εμφανές πως για μεσαίο αριθμό στοιχείων (μέχρι 35 στοιχεία) ο πιο αποδοτικός επιλυτής είναι ο bsolo για το Πρόβλημα της Συνοχής με παραμέτρους -b2-l1. Για μεγαλύτερο αριθμό στοιχείων όπως φάνηκε οι επιλυτές οι οποίοι μελετήθηκαν στην παρούσα διπλωματική δεν ήταν και τόσο αποδοτικοί. Έτσι για το πρόβλημα αυτό μπορεί να δοκιμαστούν νέες τεχνικές ικανοποίησης περιορισμών σε άλλους επιλυτές.

6.2 Μελλοντική Εργασία

Όπως έχει παρατηρηθεί το Πρόβλημα της Συνοχής είναι ένα θέμα που απασχολεί διαφορετικούς τομείς της επιστήμης, έτσι μια μελλοντική εργασία μπορεί να είναι η μοντελοποίηση του προβλήματος αυτού σε ένα από τους πιο πάνω τομείς που αναφέρθηκαν έχοντας ως δεδομένα πραγματικά στοιχεία και τις μεταξύ τους συσχετίσεις. Για παράδειγμα στην ψυχολογία αναφέρεται πως χρησιμοποιείται η θεωρία του προβλήματος συνοχής στο σχηματισμό εντύπωσης για ένα άτομο. Στο γράφο συνοχής τα στοιχεία μπορεί να είναι τα χαρακτηριστικά του ατόμου και με την εφαρμογή της θεωρίας της Συνοχής που επεξηγήθηκε στην παρούσα διπλωματική, να γίνουν δεκτά κάποια από αυτά και να απορριφθούν άλλα με τρόπο ώστε να εξάγεται σωστή εντύπωση για το άτομο αυτό.

Μια μελλοντική επέκταση της παρούσας διπλωματικής μπορεί να είναι η μοντελοποίηση του προβλήματος αυτού και σε άλλους επιλυτές που ίσως να επιλύουν το πρόβλημα αυτό σε αποδοτικότερο χρόνο από τους επιλυτές που μελετήθηκαν στην παρούσα διπλωματική. Για τον λόγο ότι οι μέθοδοι ικανοποίησης περιορισμών που χρησιμοποιήθηκαν στην παρούσα διπλωματική δεν φαίνονται να είναι αποδοτικοί για μεγάλο αριθμό στοιχείων έτσι μια επέκταση στη διπλωματική αυτή είναι η μοντελοποίηση του προβλήματος με άλλες τεχνικές ικανοποίησης περιορισμών σε επιλυτές που να είναι αποδοτικοί για μεγαλύτερο αριθμό στοιχείων.

Επιπρόσθετα μπορεί να γίνει μια μελέτη για την δημιουργία ενός αποτελεσματικότερου συνδυαστή αναζήτησης αφού υπάρχει η δυνατότητα στον επιλυτή Gecode για ορισμό δικών σου συνδυαστών αναζήτησης. Η ευρετική αναζήτηση κάνει συχνά τη διαφορά στο να επιλύει αποτελεσματικά και με επιτυχία ένα πρόβλημα ικανοποίησης περιορισμών.

Βιβλιογραφία

[1] Chekuri Chandra , Approximation Algorithms, Maximum Independent Set, Packing Integer Programs, 2011

[2] Fadi A. Aloul, "Search techniques for SAT-based Boolean optimization"The Franklin Institute. Department of Computer Engineering, American University of Sharjah, Sharjah, UAE, Published by Elsevier Ltd, 2006.

[3] Federico Heras,Vasco Manquinho ,Joao Marques-Silva"On Applying Unit Propagation-Based Lower Bounds in Pseudo-Boolean Optimization",Association for the Advancement of Artificial Intelligence (www.aaai.org) ,2008

[4] Manquinho Vasco M., João P. Marques-Silva "Satisfiability-based algorithms for Boolean optimization", Annals of Mathematics and Artificial Intelligence 40: 353–372, 2004.

[5] Manquinho Vasco , Roussel Olivier, Input / Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers , 2012

[6] Manthey Norbert and Peter Steinke, npSolver – A SAT Based Solver for Optimization Problems , Germany

[7] Marriott Kim , Stuckey J. Peter, MiniZinc Tutorial , 2013

[8] Niklas Een, Niklas Sorensson, "Translating Pseudo-Boolean Constraints into SAT", Journal on Satisfiability, Boolean Modeling and Computation 2, 2006

[9] Paul Thagard and Karsten Verbeurgt, “Coherence as Constraint Satisfaction”, Feb 11, 2010, Philosophy Department, University of Waterloo, cognitive science vol 22, 1998 ,pp.1-24

[10] Schrijvers Tom, Guido Tack, Pieter Wuille, Horst Samulowitz, Peter J. Stuckey, "Search Combinators", 6 Marc 2012

[11] Trevisan Luca, Optimization and Algorithmic Paradigmς, Linear programming relaxation of vertex cover, 2011

[12] Βλαχάβας Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας , Ικανοποίηση Περιορισμών, Βιβλίο «Τεχνητή Νοημοσύνη», Β' Έκδοση

[13] Κουμπαράκης Μανώλης , Προβλήματα Ικανοποίησης Περιορισμών, Τεχνητή Νοημοσύνη, 2009

[14] Μιχαήλ Δημήτρης, Αλγόριθμοι και Πολυπλοκότητα, NP και Υπολογιστική Δυσεπιλυσιμότητα, Αθήνα

[15] Μποναζούντας Μάρκος, Ακέραιος Προγραμματισμός , 2001

[16] Παπαδόπουλος Αντώνιος, Ακέραιος Προγραμματισμός: Μέθοδοι και Εφαρμογές, Αθήνα, 2013

[17] Ρεφανίδης Γιάννης, Προβλήματα Ικανοποίησης Περιορισμών, Λογικός Προγραμματισμός με Περιορισμούς

[18] Σεργίου Κώστας, Αναπαράσταση Γνώσης με Λογική, Προτασιακή Λογική, Λογική Πρώτης Τάξης

[19] Φωτάκης Δημήτρης, Θεωρία Γραφημάτων: Σύνολα Ανεξαρτησίας, Σύνολα Κάλυψης, και Χρωματικός Αριθμός, 2005

[20] Χατζηλυγερούδης Ιωάννης, Ικανοποίηση Περιορισμών , (Βάση το Κεφ.6 του βιβλίου «Τεχνητή Νοημοσύνη»)

Παράρτημα Α

Πιο κάτω φαίνονται τα τρία μοντέλα .mzn το οποίο έγινε σε MiniZinc. Τα μοντέλα αυτά είναι το coher_set2.mzn το οποίο είναι υλοποιημένο μέσω του partition_set και το αποτέλεσμα αναπαριστάται σε 2 σύνολα, όπου κάθε σύνολο περιέχει τα στοιχεία που ανήκουν σε αυτό. Το finaltest2.mzn το οποίο είναι υλοποιημένο με περιορισμούς σε μορφή πράξεων. Και το coher.mzn το οποίο οι περιορισμοί μου είναι υλοποιημένοι με χρήση συνεπαγωγών. Το coher.mzn είναι με το μοντέλο στο οποίο ο επιλυτής MiniZinc βρίσκει την βέλτιστη λύση πιο γρήγορα. Το μοντέλο αυτό παίρνει σαν είσοδο και ο επιλυτής Gecode.

Μοντέλο coher.mzn:

```
int: n;                         %arithmos twn komvwn
set of int:nrows= 1..n;    %arithmos grammwn
set of int:ncols= 1..n;    %arithmos stilwn

array[nrows,ncols] of int: table;  %pinakas geitniasis apo to arxeio .c

array[nrows, ncols] of var 0..1: pinakas;        %pinakas pou tha gemizei me 0 kai 1 an isxiei i den isxiei

var int: kostos;

array[ncols] of var 1..2:X;         %variable pinaka me domain metavlitwn 1 i 2

%prwtos periorismos- gia kathe paravasi(idio sinolo kai sindeontai arnitika i diaforetika sinola kai sindeontai thetika) pinakas[i,j]=1
constraint forall(i in 1..n) ( (forall(j in 1..n where i!=j)(
 ((((X[i]==X[j]) /\ (table[i,j]<0))  \/ ((X[i]!=X[j]) /\ (table[i,j]>0)) ) ->(pinakas[i,j]=1 )
                                )) ) );
```

```
%deuteros periorismos- gia kathe epitrepta(idio sinolo kai sindeontai thetika i diaforetika sinola kai sindeontai arnitika) pinakas[i,j]=0
constraint forall(i in 1..n) ((forall(j in 1..n where i!=j)(
 ((((X[i]!=X[j]) /\ (table[i,j]<0)) \/ ((X[i]==X[j]) /\ (table[i,j]>0))) ->(pinakas[i,j]=0)
                                ));

%tritos periorismos- ipologismos athroismatos kostou, opou ston pinakas[i,j]=1

constraint ((sum (i in 1..n,j in i+1..n where j!=0) (
        (abs(table[i,j])) * pinakas[i,j]) )=kostos );

%vres lisi me minimize to kostos
solve :: int_search(
        X,
        first_fail,
        indomain_min,
        complete
        )
        minimize kostos;

output[ show(X[i])| i in 1..n]++["kostos= "]++[show(pinakas[i,j])| i,j in 1..n ]
++["\n"]++[show(kostos) ]++["\n"];          %parousiasi apotelesmatos
```

Μοντέλο finaltest2.mzn

```
int: nodes;           %arithmos twn komvwn
set of int:nrows= 1..nodes;    %arithmos grammwn
set of int:ncols= 1..nodes;    %arithmos stilwn

array[nrows,ncols] of int: table;  %pinakas geitniasis apo to arxeio .c

array[nrows, ncols] of var 0..1: pinakas;        %pinakas pou tha gemizei me 0 kai 1 an exei kostos i den exei

var int: kostos; %variable metavliti pou tha kanw minimize

array[ncols] of var 1..2:X;         %variable pinaka me domain metavlitwn 1 i 2

constraint forall(i in 1..nodes) ((
                    forall(j in 1..nodes where i!=j )((
%sto idio sinolo kai sindeontai arnitika ara mpainei ston pinakas[i,j]=1 kai se diaforetika sinola kai sindeontai arnitika ara mpainei ston pinakas[i,j]=0
(( (abs(X[i]-X[j]))+ (table[i,j])+ (abs(table[i,j]))+ pinakas[i,j])=1 )
                                                \/
%se diaforetika sinola kai sindeontai thetika ara mpainei ston pinakas[i,j]=1 kai se idia sinola kai sindeontai thetika ara mpainei ston pinakas[i,j]=0
(table[i,j]+(abs(table[i,j]))+ pinakas[i,j]=table[i,j] +table[i,j]+((abs(-X[i]-X[j])) mod 2))

                    ))
                    )) ;
 %tritos periorismos- ipologismos athroismatos kostou, opou ston pinakas[i,j]=1
constraint ((sum (i in 1..nodes,j in i+1..nodes where j!=0) ( (abs(table[i,j])) * pinakas[i,j]) )=kostos );

solve :: int_search(
          X,
```

```
        first_fail,
        indomain_min,
        complete
        )
minimize kostos;

output[ show(X[i])| i in 1..nodes]++["kostos= "]++[show(pinakas[i,j])| i,j in 1..nodes
]++["\n"] ++[show(kostos) ]++["\n"];          %parousiasiasi  apotelesmatos
```

Μοντέλο coher_set2.mzn

```
include "partition_set.mzn";
int: nodes;           %arithmos twn komvwn
set of int: nrows= 1..nodes;    %arithmos grammwn
set of int: ncols= 1..nodes;    %arithmos stilwn

array[ncols, ncols] of int: table;  %pinakas geitniasis apo to arxeio .c

array[ncols, ncols] of var 0..1: pinakas; %pinakas pou tha gemizei me 0 kai 1 an isxiei i den isxiei

%array[ncols] of var 1..2:X;          %variable pinaka me domain metavlitwn 1 i 2
array[1..2] of var set of nrows: X;
%array[1..3,1..6] of var bool: charact;
var int: kostos;

constraint partition_set([X[i]|i in 1..2], nrows);

constraint forall(i in 1..nodes) ((forall(j in 1..nodes where i!=j)(
(((((i in X[1])==(j in X[1])) \/ (((i in X[2])==(j in X[2]))) ) /\ (table[i,j]<0))
                              \/
((((i in X[1])!=(j in X[1])) \/ (((i in X[2])!=(j in X[2]))) ) /\ (table[i,j]>0))
->(pinakas[i,j]=1 ))) )) ;

constraint forall(i in 1..nodes) ((forall(j in 1..nodes where i!=j)(
(((((i in X[1])!=(j in X[1])) \/ (((i in X[2])!=(j in X[2]))) ) /\ (table[i,j]<0))
```

```
                    \/
((((i in X[1])==(j in X[1])) \/ (((i in X[2])==(j in X[2]))) ) /\ (table[i,j]>0))
                    ->(pinakas[i,j]=0 )
                    )) )) ;


constraint ((sum (i in 1..nodes,j in i+1..nodes where j!=0) ( (abs(table[i,j])) *
pinakas[i,j]) )=kostos );

solve :: set_search(
        [X[i]|i in 1..2],
        first_fail,
        indomain_min,
        complete
        )
        minimize kostos;


output [show(X)]++["\n"]++[show(kostos)];
```

Παράρτημα Β

Εδώ τοποθετείται το αρχείο .mzn το οποίο χρησιμοποιεί το bab (branch and bound) search combinator.

```
include "stdlib.mzn";

annotation sh_letvar(string:x);
annotation sh_intvar(var int: x);
annotation sh_int(int: x);
annotation sh_plus(ann:x,ann:y);
annotation sh_minus(ann:x,ann:y);
annotation sh_mult(ann:x,ann:y);
annotation sh_mod(ann:x,ann:y);
annotation sh_idiv(ann:x,ann:y);
annotation sh_uminus(ann:x);
annotation sh_cond_eq(ann:x,ann:y);
annotation sh_cond_nq(ann:x,ann:y);
annotation sh_cond_lt(ann:x,ann:y);
annotation sh_cond_lq(ann:x,ann:y);
annotation sh_cond_gt(ann:x,ann:y);
annotation sh_cond_gq(ann:x,ann:y);
annotation sh_cond_and(ann:x,ann:y);
annotation sh_cond_or(ann:x,ann:y);
annotation sh_cond_not(ann:x);
annotation sh_cond_impl(ann:x,ann:y);
annotation sh_cond_rimpl(ann:x,ann:y);
annotation sh_cond_equiv(ann:x,ann:y);
annotation sh_cond_xor(ann:x,ann:y);
annotation sh_cond_false;
annotation sh_cond_true;
annotation sh_solutioncount(ann:x, ann:y);
annotationsh_nodecount(ann:x, ann:y);
```

```
annotation sh_failurecount(ann:x, ann:y);
annotation sh_depthcount(ann:x, ann:y);
annotation sh_discrepancycount(ann:x, ann:y);
annotation sh_timecount(ann:x, ann:y);
annotation sh_prune;
annotation sh_let(ann:x,ann:e,ann:s);
annotation sh_assign(ann:x,ann:e);
annotation sh_post(ann:x,ann:s);
annotation sh_post_succeed(ann:x);
annotation sh_ifthenelse(ann:c,ann:s1,ann:s2);
annotation sh_and(array[int] of ann:s);
annotation sh_or(array[int] of ann:s);
annotation sh_portfolio(array[int] of ann:s);
annotation sh_restart(ann:c, ann:s);
annotation sh_print_int(array[int] of var int: x, ann:s);
annotation sh_print_bool(array[int] of var bool: x, ann:s);
annotation sh_trace_sol_string(string: s, ann:a);
annotation sh_trace_sol_int(int: s, ann:a);
annotation trace_sol(int: s, ann:a) = sh_trace_sol_int(s,a);
annotation trace_sol(string: s, ann:a) = sh_trace_sol_string(s,a);
annotation sh_val_exclude_lb;
ann: exclude_lb = sh_val_exclude_lb;
annotation sh_val_assign_ub;
ann: assign_ub = sh_val_assign_ub;
annotation sh_val_exclude_ub;
ann: exclude_ub = sh_val_exclude_ub;
annotation sh_val_assign_median;
ann: assign_median = sh_val_assign_median;
annotation sh_val_exclude_median;
ann: exclude_median = sh_val_exclude_median;
annotation sh_val_assign_random;
ann: assign_random = sh_val_assign_random;
annotation sh_val_exclude_random;
```

```
ann: exclude_random = sh_val_exclude_random;
annotation sh_val_bisect_low;
ann: bisect_low = sh_val_bisect_low;
annotation sh_val_bisect_high;
ann: bisect_high = sh_val_bisect_high;
annotation sh_var_max_afc;
ann: max_afc = sh_var_max_afc;
ann: prune = sh_prune;
annotation sh_int_search(array[int] of var int: x, ann:varsel,ann:valsel);
annotation int_search(array[int] of var int: x, ann:varsel,ann:valsel)
  = sh_int_search(x,varsel,valsel);
annotation sh_var_random_order;
ann: random_order = sh_var_random_order;
annotation sh_val_assign_lb;
ann: assign_lb = sh_val_assign_lb;

% ilopoisi tou annotation bab
annotation bab(var int: obj, ann:s) =
    let {
      svar int: best = 1000000
    } in (
      post(obj < lv("best"),
        and(s,assign(best,obj))
      )
```

```
   );


%--------------------------------------------------------------------------%

% Include solver-specific redefinitions for any FlatZinc built-ins.
%
include "redefinitions.mzn";

%--------------------------------------------------------------------------%
%THE PROGRAM START
%--------------------------------------------------------------------------%

int: n;           %arithmos twn komvwn
set of int:nrows= 1..n;    %arithmos grammwn
set of int:ncols= 1..n;    %arithmos stilwn

array[nrows,ncols] of int: table;  %pinakas geitniasis apo to arxeio .c

array[nrows, ncols] of var 0..1: pinakas;        %pinakas pou tha gemizei me 0 kai 1 an isxiei i den isxiei

var int: kostos;

array[ncols] of var 1..2: X;         %variable pinaka me domain metavlitwn 1 i 2

%prwtos periorismos- gia kathe paravasi(idio sinolo kai sindeontai arnitika i diaforetika sinola kai sindeontai thetika) pinakas[i,j]=1
constraint forall(i in 1..n) ((forall(j in 1..n where i!=j)(
((((X[i]==X[j]) /\ (table[i,j]<0))  \/ ((X[i]!=X[j]) /\ (table[i,j]>0)) )
->(pinakas[i,j]=1 )
```

```
)) ) );

%deuteros periorismos- gia kathe epitrepta(idio sinolo kai sindeontai thetika i diaforetika sinola kai sindeontai arnitika) pinakas[i,j]=0
constraint forall(i in 1..n) ((forall(j in 1..n where i!=j)(
((((X[i]!=X[j]) /\ (table[i,j]<0)) \/ ((X[i]==X[j]) /\ (table[i,j]>0))) ->(pinakas[i,j]=0)
                    ))));


%tritos periorismos- ipologismos athroismatos kostou, opou ston pinakas[i,j]=1

constraint ((sum (i in 1..n,j in i+1..n where j!=0) ( (abs(table[i,j])) * pinakas[i,j]) )=kostos );


%vres lisi me minimize to kostos xrisimopoiontas search combinator-bab(branch and bound)
solve :: print(X,bab(kostos,int_search(X,random_order,assign_lb))) satisfy;


%OUTPUT
%parousiasi apotelesmatos
output[ show(X[i])| i in 1..n]++["kostos= "]++["\n"]++[show(kostos) ]++["\n"];
```

Παράρτημα Γ

Εδώ φαίνεται το πρόγραμμα στη γλώσσα προγραμματισμού c το οποίο παίρνει ως είσοδο των αριθμό των στοιχείων, τον τρόπο απεικόνισης της συσχέτισης των στοιχείων καθώς και την πυκνότητα των στοιχείων και ως έξοδο δημιουργούνται τέσσερα αρχεία στην μορφή που απαιτεί κάθε επιλυτής. Το ένα είναι το αρχείο final_file.dzn το οποίο παίρνουν ως είσοδο οι επιλυτές MiniZinc και Gecode, το αρχείο file που παίρνει ως είσοδο ο bsolo, το αρχείο final το οποίο παίρνει ως είσοδο ο Minisat+ και το αρχείο filenp.opb το οποίο παίρνει ως είσοδο ο επιλυτής npSolver

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

//sinartisi pou epistrefei random akeraious metaksi tou Min kai Max
int RandRange(int Min, int Max) {
                                        int diff = Max - Min;
                                        return (int) (((double) (diff + 1) / RAND_MAX) *
rand() + Min);
}

//sinartisi pou ilopoiei to arxeio me pseudoboolean constraints kai tin antikeimeniki
sinartisi gia ton npSolver
void concat3(float pik, int r, FILE* outp3, int **K, int diagwnios, int var) {

                                        char str1[20];
                                        int k;
                                        int p;
```

```
                    //dilwseis strings gia apothikeusi twn metavlitwn Xi1,
Xi2,Xnum
                    char str2[20];
                    char str3[20];
                    char buffer[2];
                    char buffer2[1];
                    char buffer3[2];
                    char buffer3e[2];
                    char buffer1[1];
                    char stra[20];
                    char strb[20];
                    char buffer2a[2];
                    char buffer2b[2];
                    char buffer2c[2];
                    char buffer2d[2];
                    char buffer21[2];
                    char buffer31[2];
                    char buffer3a[2];
                    char buffer22b[2];
                    char buffer22c[2];
                    char buffer22d[2];
                    char buffer221[2];
                    static int count;
                    static int piknotita;
                    piknotita = var;

                    int rold = r;                    //rold krata twn arithmo twn
stoixeiwn
                    char *test = (char*) malloc(rold * sizeof(char));
                    int i, j;
                    int variables = 0;
                    int constraint = 0;
```

```
                int *table; //dilwsi kai desmeusi xwrou gia ton pinaka
me ta strings-pinakas pou tha filagei tis metavlites pou tha simmetexoun stin
antikeimeniki sinartisi

                table = (int *) malloc(diagwnios * sizeof(int));
                for (i = 1; i <= diagwnios; i++) {
                        table[i] = 0;
                }

                sprintf(str1, "%d", r);
                sprintf(str2, "%d", 2);
                strcat(str1, str2);

                int num = atoi(str1); //vriskei tin teleutaia
xrisimopoiimeni metavliti

//vriskw poses metavlites tha periexonatai sto arxeio gia ton npSolver
                variables = num + var;
//vriskw posous periorismous tha periexonatai sto arxeio gia ton npSolver
                constraint = var * 2 + rold;

//grapsimo sto arxeio npSolver
                fprintf(outp3, "* #variable= %d #constraint= %d\n",
variables, constraint); //prwti grammi sto arxeio gia npSolver
//grapsimo twn pio katw comments sto arxeio npSolver
                fprintf(outp3, "%s \n",
"*****************************************");
                fprintf(outp3, "%s \n", "* begin normalizer comments");
                fprintf(outp3, "%s \n", "* category= OPT-SMALLINT-
LIN");
                fprintf(outp3, "%s \n", "* end normalizer comments");
                fprintf(outp3, "%s \n",
"*****************************************");
```

```
//grapsimo antikeimenikis sinartisis sto arxeio npSolve
                                        fprintf(outp3, "%s ", "min:");

////////////////////////////
                                        p = 0;
                                        k = 1;
                                        for (i = 1; i <= rold; i++) {
                                                for (j = 1; j <= rold; j++) {

                                                        if (i < j && K[i][j] != 0) {
                                                                strcpy(buffer2, "x"); //apothikeutai
sto string buffer2 to x
                                                                num = num + 1;//meta tin euresi tis
teleytaias xrisimopoioumenis metavlitis auksanetai o arithmos autos gia apeikonisi twn
ipoloipwn metavlitwn
                                                                table[k] = num; //filagontai oi
arithmoi metavlitwn pou tha xrisimopoiithoun stous periorismous
                                                                k++;
                                                                sprintf(str3, "%d", num);
                                                                strcat(buffer2, str3);  //dimiourgia
tou string xnum
                                                                test = buffer2;
                                                                sprintf(str3, "%d", abs(K[i][j]));
//kostos
                                                                strcat(str3, " ");
                                                                strcat(str3, buffer2); //kostos*
xnum
                                                                count++;
                                                                fprintf(outp3, "%s", str3);
//grapsimo sto arxeio kostos * xnum
```

```
                                        if (piknotita != count) { //enosw
den einai to teleutaio mi mediniko stoixeio tis panw diagwniou tipwnei +

                                                fprintf(outp3, "%s", " +");

                                        } else {
                                    //alliws tipwnei erwtimatiko

                                                fprintf(outp3, "%s ", " ; ");
                                                break;
                                        }

                                    }

                            }
                    }

//exei dimiourgithei i antikeimeniki sinartisi

                    fprintf(outp3, "%s", "\n");

                    i = 1;
                    //dimiourgia periorismwn-ena stoixeio se ena mono
sinolo.oxi kai sta dio
                    for (i = 1; i <= rold; i++) {
                            sprintf(stra, "%d", i);
                            sprintf(strb, "%d", i);
                            strcpy(buffer2a, "x");
                            strcpy(buffer3a, "x");

                            strcat(buffer2a, stra);
                            strcat(buffer3a, stra);
                            strcat(buffer2a, "1");   //xi1
```

```
                    strcat(buffer3a, "2");   //xi2

                    fprintf(outp3, "%s", "+1 ");
                    fprintf(outp3, "%s", buffer2a); //+1* xi1
                    fprintf(outp3, "%s", " +");
                    fprintf(outp3, "%s", "1 ");
                    fprintf(outp3, "%s", buffer3a);          //+1* xi2
                    fprintf(outp3, "%s", " = ");
                    fprintf(outp3, "%s", " 1");
                    fprintf(outp3, "%s", " ;");
                    fprintf(outp3, "%s", "\n");

                }
//dimiourgia sinthetwn periorismwn me vasi to kostos

//dilwseis strings pou tha apothikeuoun metavlites tis morfis Xnum
                    char str11[20];
                    char str11a[20];
                    char str11b[20];
                    char str112[20];
                    char str113[20];
                    char s11[20];
                    char s11a[20];
                    char s112[20];
                    char s11b[20];
                    char buffer12[2];      //xx
                    char buffer13[2]; //xm
                    char buffer11a[2]; //x
                    char buffer11b[2]; //x
                    char buffer11c[2]; //x
                    char buffer11d[2]; //x
                    char buffer12a[2]; //x
                    char buffer12b[2]; //x
```

```
char buffer12c[2]; //x
char buffer12d[2]; //x
char buffer121[2]; //xx
char buffer131[2]; //xx
char buffer223a[2]; //x
char strnum[20];
char strnum1[20];
char strnum2[20];
char strnum3[20];

int i1 = 0;
k = 1;
for (i1 = 1; i1 <= rold; i1++) {
        for (j = 1; j <= rold; j++) {
                sprintf(str11, "%d", i1);
                sprintf(str11a, "%d", i1);
                sprintf(str112, "%d", j);
                sprintf(str11b, "%d", j);
                strcat(str11a, str11b); //enwsi twn dio
arithmwn

                sprintf(s11, "%d", i1);
                sprintf(s11a, "%d", i1);
                sprintf(s112, "%d", j);
                sprintf(s11b, "%d", j);

                if (i1 < j && (K[i1][j] != 0)) {
                        //anw trigwniko

                        if ((K[i1][j]) > 0) { //an ston pinaka
ta stoixeia sindeontai thetika

                                strcpy(buffer12, "x");
```

```
                                        strcpy(buffer13, "x");
                                        strcpy(buffer11a, "x");
                                        strcpy(buffer11b, "x");
                                        strcpy(buffer11c, "x");
                                        strcpy(buffer11d, "x");
                                        strcpy(buffer121, "x");
                                        strcpy(buffer131, "xm");
                                        sprintf(strnum, "%d",
table[k]);//pernei to ena num to opoio einai apothikeumeno sto pinaka table k ws
apeikonisi tou xxij i tou xmij
                                        strcat(buffer12, strnum);
//dimiourgia tou string xxij alla edw to string auto tha einai tis morfis xnum

                                        strcat(buffer11a, str11);
                                        strcat(buffer11a, "1");
                                //xi1

                                        strcat(buffer11b, str112);
                                        strcat(buffer11b, "2");
                                //xj2

                                        strcat(buffer11c, str11);
                                        strcat(buffer11c, "2");
                                //xi2

                                        strcat(buffer11d, str112);
                                        strcat(buffer11d, "1");
                                //xj1

                                        fprintf(outp3, "%s", "-1 ");
                                        fprintf(outp3, "%s",
buffer11a);
```

```
                                                    fprintf(outp3, "%s", " -1 ");
                                                    fprintf(outp3, "%s",
buffer11b);
                                                    fprintf(outp3, "%s", " +1 ");
                                                    fprintf(outp3, "%s",
buffer12);                    //tipwnw +1*xnum sto arxeio

                                                    fprintf(outp3, "%s", " >= -1
;");

                                                    fprintf(outp3, "%s", "\n");

                                                    sprintf(strnum1, "%d",
table[k]);
                                                    strcat(buffer121, strnum1);
//dimiourgia tou string xnum s

                                                    fprintf(outp3, "%s", "-1 ");

                                                    fprintf(outp3, "%s",
buffer11c);

                                                    fprintf(outp3, "%s", " -1 ");

                                                    fprintf(outp3, "%s",
buffer11d);

                                                    fprintf(outp3, "%s", " +1 ");

                                                    fprintf(outp3, "%s",
buffer121); //tipwnw +1*xnum sto arxeio
```

```
                                        fprintf(outp3, "%s", " >= -1
;");

                                        fprintf(outp3, "%s", "\n");

                                        k++;

                                    }

                                    else {

                                        //strcpy(buffer2, "x");
                                        strcpy(buffer3, "x");
                                        strcpy(buffer223a, "x");
                                        strcpy(buffer22b, "x");
                                        strcpy(buffer22c, "x");
                                        strcpy(buffer22d, "x");
                                        strcpy(buffer221, "xx");
                                        strcpy(buffer31, "x");

                                        sprintf(strnum2, "%d",
table[k]);
                                        strcat(buffer3, strnum2);
////dimiourgia tou string xnum opou num to pairnoume apo ton pinaka table[k];

                                        strcat(buffer223a, s11);
                                        strcat(buffer223a, "1");
                                    //xi1

                                        strcat(buffer22b, s112);
                                        strcat(buffer22b, "1");
                                    //xj1
```

```
                                                    strcat(buffer22c, s11);
                                                    strcat(buffer22c, "2");
                                        //xi2

                                                    strcat(buffer22d, s112);
                                                    strcat(buffer22d, "2");
                                        //xj2

                                                    fprintf(outp3, "%s", "-1 ");

                                                    fprintf(outp3, "%s",
buffer223a);

                                                    fprintf(outp3, "%s", " -1 ");

                                                    fprintf(outp3, "%s",
buffer22b);

                                                    fprintf(outp3, "%s", " +1 ");

                                                    fprintf(outp3, "%s",
buffer3);

                                                    fprintf(outp3, "%s", " >= -1
;");

                                                    fprintf(outp3, "%s", "\n");

                                                    sprintf(strnum3, "%d",
table[k]);
                                                    strcat(buffer31,
strnum3);//dimiourgia tou string xnum sto str3
```

```
                                        fprintf(outp3, "%s", "-1 ");
                                        fprintf(outp3, "%s",
buffer22c);
                                        fprintf(outp3, "%s", " -1 ");
                                        fprintf(outp3, "%s",
buffer22d);
                                        fprintf(outp3, "%s", " +1 ");
                                        fprintf(outp3, "%s",
buffer31); //tipwnw +1*xnum sto arxeio
                                        fprintf(outp3, "%s", " >= -1
;");
                                        fprintf(outp3, "%s", "\n");
                                        k++;
                                    }
                                }
                            }
                        }
//////////////////////////
}
```

```
//i sinartisi concat tipwnei tin antikeimeniki sinartisi
void concat(float pik, int x, int y, int r, int value, FILE *outp1, FILE* outp2,
                                        int **K) {

                                        char str1[20];
                                        char str2[20];
                                        char str3[20];
                                        char str4[20];
                                        char buffer[2];
                                        char buffer2[2];        //xx
                                        char buffer3[2]; //xm
                                        char buffer1[1]; //*
                                        int p, i;
                                        int val = 0;
                                        val = value;
                                        int new = abs(val);
                                        static int count = 0;

                                        static int piknotita1;
                                        piknotita1 = pik * r * (r - 1) / 2;

                                        int number = 0;
                                        strcpy(buffer2, "xx");
                                        strcpy(buffer3, "xm");

                                        sprintf(str1, "%d", x);
                                        sprintf(str2, "%d", y);
                                        sprintf(str3, "%d", new);
                                        sprintf(str4, "%d", new);
                                        strcat(str1, str2);
                                        strcat(str3, "*"); //kostos*
                                        strcat(str4, " "); //kostos space
```

```
                    if (value > 0) {
                        strcat(buffer2, str1);  //dimiourgia tou string xxij
                        strcat(str3, buffer2); //kostos* xx
                        strcat(str4, buffer2); //kostos space xx
                        fprintf(outp1, "%s", str3);
                        fprintf(outp2, "%s", str4);
                        count++;

                    } else {
                        strcat(buffer3, str1);
                        strcat(str3, buffer3); //kostos* xm
                        strcat(str4, buffer3); //kostos space xm
                        fprintf(outp1, "%s", str3);
                        fprintf(outp2, "%s", str4);
                        count++;
                    }

                    if (count != piknotita1) { //enosw den einai to count den
einai iso me tin piknotita, ara akoma den graftikan ola ta stoixeia stin antikeimeniki
sinartisi kai typwnei +
                        fprintf(outp1, "%s", "+");
                        fprintf(outp2, "%s", " +");

                    } else {
                    //alliws tipwnei erwtimatiko
                        fprintf(outp1, "%s", ";");
                        fprintf(outp2, "%s", " ;");
                    }

}

//i sinartisi concat2 tipwnei tous ipoloipous periorismous
```

```
void concat2(int x, int y, int r, int value, FILE *outp1, FILE *outp2,
                                        int **K) {

                                    char str1[20];
                                    char stra[20];
                                    char strb[20];
                                    char str2[20];
                                    char str3[20];
                                    char buffer[2];
                                    char buffer2[2];        //xx
                                    char buffer3[2]; //xm
                                    char buffer1a[2]; //x
                                    char buffer1b[2]; //x
                                    char buffer1c[2]; //x
                                    char buffer1d[2]; //x
                                    char buffer2a[2]; //x
                                    char buffer2b[2]; //x
                                    char buffer2c[2]; //x
                                    char buffer2d[2]; //x
                                    char buffer21[2]; //xx
                                    char buffer31[2]; //xx

                                    int i, j;

                                    for (i = 1; i <= r; i++) {
                                            for (j = 1; j <= r; j++) {
                                                    sprintf(str1, "%d", i);
                                                    sprintf(stra, "%d", i);
                                                    sprintf(str2, "%d", j);
                                                    sprintf(strb, "%d", j);
                                                    strcat(stra, strb); //enwsi twn dio arithmwn

                                                    if (i < j && K[i][j] != 0) {
```

```
                                        if (K[i][j] > 0) {

                                                strcpy(buffer2, "xx");
                                                strcpy(buffer3, "xm");
                                                strcpy(buffer1a, "x");
                                                strcpy(buffer1b, "x");
                                                strcpy(buffer1c, "x");
                                                strcpy(buffer1d, "x");
                                                strcpy(buffer21, "xx");
                                                strcpy(buffer31, "xm");

                                                strcat(buffer2, stra);
//dimiourgia tou string xxij

                                                strcat(buffer1a, str1);
                                                strcat(buffer1a, "1");
                                        //xi1

                                                strcat(buffer1b, str2);
                                                strcat(buffer1b, "2");
                                            //xj2

                                                strcat(buffer1c, str1);
                                                strcat(buffer1c, "2");
                                            //xi2

                                                strcat(buffer1d, str2);
                                                strcat(buffer1d, "1");
                                            //xj1

                                                fprintf(outp1, "%s", "-1*");
                                                fprintf(outp2, "%s", "-1 ");
```

```
                                                    fprintf(outp1, "%s",
buffer1a);
                                                    fprintf(outp2, "%s",
buffer1a);
                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", " -1 ");
                                                    fprintf(outp1, "%s",
buffer1b);
                                                    fprintf(outp2, "%s",
buffer1b);
                                                    fprintf(outp1, "%s", "+1*");
                                                    fprintf(outp2, "%s", " +1 ");
                                                    fprintf(outp1, "%s",
buffer2);                       //tipwnw +1*xxij sto arxeio
                                                    fprintf(outp2, "%s",
buffer2);
                                                    fprintf(outp1, "%s", ">=-
1;");
                                                    fprintf(outp2, "%s", " >= -1
;");
                                                    fprintf(outp1, "%s", "\n");
                                                    fprintf(outp2, "%s", "\n");

                                                    strcat(str1, str2);
                                                    strcat(buffer21, str1);
//dimiourgia tou string xxij

                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", " -1 ");
                                                    fprintf(outp1, "%s",
buffer1c);
                                                    fprintf(outp2, "%s",
buffer1c);
```

```
                                        fprintf(outp1, "%s", "-1*");
                                        fprintf(outp2, "%s", " -1 ");
                                        fprintf(outp1, "%s",
buffer1d);
                                        fprintf(outp2, "%s",
buffer1d);
                                        fprintf(outp1, "%s", "+1*");
                                        fprintf(outp2, "%s", " +1 ");
                                        fprintf(outp1, "%s",
buffer21); //tipwnw +1*xxij sto arxeio
                                        fprintf(outp2, "%s",
buffer21);
                                        fprintf(outp1, "%s", ">=-
1;");
                                        fprintf(outp2, "%s", " >= -1
;");
                                        fprintf(outp1, "%s", "\n");
                                        fprintf(outp2, "%s", "\n");

                                }

                                else {

                                        strcpy(buffer2, "xx");
                                        strcpy(buffer3, "xm");
                                        strcpy(buffer2a, "x");
                                        strcpy(buffer2b, "x");
                                        strcpy(buffer2c, "x");
                                        strcpy(buffer2d, "x");
                                        strcpy(buffer21, "xx");
                                        strcpy(buffer31, "xm");
```

```
                                                    strcat(buffer3, stra);
////dimiourgia tou string xmij

                                                    strcat(buffer2a, str1);
                                                    strcat(buffer2a, "1");
                                        //xi1

                                                    strcat(buffer2b, str2);
                                                    strcat(buffer2b, "1");
                                            //xj1

                                                    strcat(buffer2c, str1);
                                                    strcat(buffer2c, "2");
                                            //xi2

                                                    strcat(buffer2d, str2);
                                                    strcat(buffer2d, "2");
                                            //xj2

                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", "-1 ");
                                                    fprintf(outp1, "%s",
buffer2a);
                                                    fprintf(outp2, "%s",
buffer2a);
                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", " -1 ");
                                                    fprintf(outp1, "%s",
buffer2b);
                                                    fprintf(outp2, "%s",
buffer2b);
                                                    fprintf(outp1, "%s", "+1*");
                                                    fprintf(outp2, "%s", " +1 ");
```

```
                                                    fprintf(outp1, "%s",
buffer3);                       //tipwnw +1*xmij sto arxeio
                                                    fprintf(outp2, "%s",
buffer3);
                                                    fprintf(outp1, "%s", ">=-
1;");
                                                    fprintf(outp2, "%s", " >= -1
;");
                                                    fprintf(outp1, "%s", "\n");
                                                    fprintf(outp2, "%s", "\n");

                                                    strcat(buffer31, stra);
                                //dimiourgia tou string xmij

                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", " -1 ");
                                                    fprintf(outp1, "%s",
buffer2c);
                                                    fprintf(outp2, "%s",
buffer2c);
                                                    fprintf(outp1, "%s", "-1*");
                                                    fprintf(outp2, "%s", " -1 ");
                                                    fprintf(outp1, "%s",
buffer2d);
                                                    fprintf(outp2, "%s",
buffer2d);
                                                    fprintf(outp1, "%s", "+1*");
                                                    fprintf(outp2, "%s", " +1 ");
                                                    fprintf(outp1, "%s",
buffer31); //tipwnw +1*xmij sto arxeio
                                                    fprintf(outp2, "%s",
buffer31);
```

```
                                                    fprintf(outp1, "%s", ">=-
1;");
                                                    fprintf(outp2, "%s", " >= -1
;");
                                                    fprintf(outp1, "%s", "\n");
                                                    fprintf(outp2, "%s", "\n");

                                                }

                                            }

                                        }

                                    }

}

int main(int argc, char **argv) {

                                    FILE* outp1; //dilwsi arxeiou gia minisat+
                                    FILE* outp2; //dilwsi arxeiou gia bsolo
                                    FILE* outp3; //dilwsi arxeiou gia npSolver
                                    FILE* outp; //dilwsi arxeiou .dzn
                                    int **B;            //dilwsi disdiastatou pinaka
                                    int i, j;
                                    int r;
                                    float pik;
                                    int timi;
                                    int diagwnios = 0;
                                    int piknotita;
                                    int value;
                                    int ans = 0;
////////////////////////////////////////////////////////////////////
```

```
//Dedomena eisodou sto programma
////////////////////////////////////////////////////////////////
                                printf("DWSE ARITHMO STOIXEIWN: ");
                                scanf("%d", &r);
                                //printf("%d\n", r);

                                printf("\n");

//zitw apo to xristi na mou dwsei piknotita
                                printf("DWSE PIKNOTITA TWN STOIXEIW:[0
MEXRI 1] ");
                                scanf("%f", &pik);
                                int pososto = pik * 100;
                                printf(
                                                "MONO TO %d % TWN STOIXEIWN
SISXETIZONTAI METAKSI TOUS EITE THETIKA EITE ARNITIKA\n",
                                                pososto);

                                if (pik < 0 || pik > 1) {
                                      do {
//zitw apo to xristi na mou dwsei piknotita
                                                printf("DWSE PIKNOTITA TWN
STOIXEIW:[0 MEXRI 1] ");
                                                scanf("%f", &pik);
                                                printf(
                                                                "MONO TO %d % TWN
STOIXEIWN SISXETIZONTAI METAKSI TOUS EITE THETIKA EITE
ARNITIKA\n",
                                                                pososto);
                                      } while (pik < 0 || pik > 1);

                                };
```

```
                printf("\n");

//zitw apo to xristi na mou dwsei piknotita
                printf(
                        "DIALEKSE PWS THA APEIKONIZEIS
TIS SISXETISEIS METAKSI TWN STOIXEIWN \n");
                printf(

                "##########################################
########################################\n");
                printf(
                        "OI SISXETISEIS TWN STOIXEIWN
NA APEIKONIZONTAI ME (-10) - (10) TOTE PLIKTROLOGISE 1:\n");
                printf(
                        "OI SISXETISEIS TWN STOIXEIWN
NA APEIKONIZONTAI ME (-1 ) - (1 ) TOTE PLIKTROLOGISE 2:\n");
                printf(

                "##########################################
########################################\n");
                printf("DWSE TIN EPILOGI SOU:");
                scanf("%d", &ans);

                if (ans != 1 && ans != 2) {
//zitw apo to xristi na mou dwsei piknotita
                    do {
                        printf(
                                "DIALEKSE PWS THA
APEIKONIZEIS TIS SISXETISEIS METAKSI TWN STOIXEIWN \n");
                        printf(

                "##########################################
########################################\n");
```

```
                                        printf(
                                                "OI SISXETISEIS TWN
STOIXEIWN NA APEIKONIZONTAI ME (-10) - (10) TOTE PLIKTROLOGISE
1:\n");
                                        printf(
                                                "OI SISXETISEIS TWN
STOIXEIWN NA APEIKONIZONTAI ME (-1 ) - (1 ) TOTE PLIKTROLOGISE
2:\n");
                                        printf(

                                "##########################################
########################################\n");
                                        printf("DWSE TIN EPILOGI SOU:");
                                        scanf("%d", &ans);
                                } while (ans != 1 && ans != 2);
                        }

                        printf("\n");

                        printf(

                        "##########################################
########################################\n");
                        printf("TA ARXEIA STIS MORFES TWN SOLVERS
EXOUN DIMIOURGITHEI\n");
                        printf("TELOS PROGRAMMAYOS\n");
                        printf(

                        "##########################################
########################################\n");
////////////////////////////////////////////////////////////////////////////////

                        int max = 1;
```

```
int **K;
int **thelp;
int min = -1;
char **num;
int plithos = 0;
int p = 0;
int var = 0;

int count = 0; //metritis gia string_table

char stra[20];
char strb[20];
char strc[20];
char buffer2a[2];      //x
char buffer3a[2];      //x
char snum[2];
char **pinakas = (char**) malloc(r * sizeof(char*));
//dilwsi kai desmeusi xwrou gia ton pinaka me ta strings

for (p = 1; p <= r; ++p) {
        pinakas[p] = (char*) malloc(r * sizeof(char));
//desmesusi dinamikou pinaka
}

//evresi stoixeia tou anw trigwnikou pinka
diagwnios = ((r * r) - r) / 2;

//anoigma twn arxeiwn
outp1 = fopen("final", "w");
outp2 = fopen("file", "w");
outp3 = fopen("filenp.opb", "w");
//grapsimo sto arxeio
fprintf(outp1, "min:");
```

```
                    fprintf(outp2, "min: ");

                    srand (time(NULL)); //enarksi xronometrou voitha tin
random sinartisi

                    //dinamiki desmeusi xwrou gia ton pinaka sysxetisewn
twn stoixeiwn
B                   = (int **) malloc(r * sizeof(int *));
                    for (i = 1; i <= r; i++) {
                        B[i] = (int *) malloc(r * sizeof(int));
                    }
//arxikopoisi pinaka me miden
                    for (i = 1; i <= r; i++) {
                        for (j = 1; j <= r; j++) {
                            B[i][j] = 0;
                        }
                    }
//dinamiki desmeusi xwrou gia ton pinaka sysxetisewn twn stoixeiwn
                    K = (int **) malloc(r * sizeof(int *));
                    for (i = 1; i <= r; i++) {
                        K[i] = (int *) malloc(r * sizeof(int));
                    }
//arxikopoisi pinaka me miden
                    for (i = 1; i <= r; i++) {
                        for (j = 1; j <= r; j++) {
                            K[i][j] = 0;
                        }
                    }

//dinamiki desmeusi xwrou gia ton pinaka sysxetisewn twn stoixeiwn
                    thelp = (int **) malloc(r * sizeof(int *));
                    for (i = 1; i <= r; i++) {
                        thelp[i] = (int *) malloc(r * sizeof(int));
```

```
                                        }
//arxikopoisi pinaka me miden
                                        for (i = 1; i <= r; i++) {
                                                for (j = 1; j <= r; j++) {
                                                        thelp[i][j] = 0;
                                                }
                                        }
//////////////////////////////////////////////////////////////////
//ipologizetai posa stoixeia sindeontai metaksi tous simfwna me tin piknotitaa pou
dothike
//osa stoixeia tha sindeontai apeikonizontai ston disdiastato pinaka thelp me asso
//////////////////////////////////////////////////////////////////


                                        do {
                                                piknotita = pik * r * (r - 1) / 2; // ekteleitai mexri
na mpoun oses akmes prepei
                                                for (i = 1; i <= r; i++) {
                                                        for (j = 1; j <= r; j++) {
                                                                if (i == j) {           // stin diagwnio
vazoume miden
                                                                        thelp[i][j] = 0;
                                                                } else if (i < j) { //gia na pernw tin
panw diagwnio-epeidi einai simmetrika
                                                                        thelp[i][j] = 1;
                                                                        plithos++;
                                                                        piknotita--;
                                                                        if (piknotita == 0)
                                                                                break;      // av
evala oses akmes eprepe tote stamatw
                                                                }
                                                        }
                                                        if (piknotita == 0)     // av evala oses
akmes eprepe tote stamatw
```

```
                                        break;
                                }

                        } while (piknotita != 0);

//an dothike ws eisodo sto menu epilogwn sisxetisewn twn stoixeiwn i epilogi 1 tote ta stoixeia tha sindeontai me akeraious arithmous apo to 10-10

                        if (ans == 1) {

//////////////////////////////////////////////////////////////////////////////
                                for (i = 1; i <= r; i++) {
                                        for (j = 1; j <= r; j++) {
                                                if (i == j)
                                                        value = 0;

                                                if (thelp[i][j] == 1) {   //an ston pinaka thelp [i][j] exei 1 tote ta i,j sindeontai kai etsi ipologizetai enas akeraios random arithmos apo -10 mexri 10

                                                        value = RandRange(-10, 10);             //sinartisi i opoia epistrefei randoma akeraio apo to -10 ews 10
                                                        do {
                                                                value = RandRange(-10, 10);             //sinartisi i opoia epistrefei randoma akeraio apo to -10 ews 10
                                                                if (value == 0) {//enosw epistrefei 0 tote ksanakaleitai i sinartisi
                                                                        value = RandRange(-10, 10); //sinartisi i opoia epistrefei randoma akeraio apo to -10 ews 10
                                                                }
```

```
                                                    } while (value >= 10 || value
<= -10 || value == 0);

                                                    K[i][j] = value;//apothikeusi
timis ston anw trigwniko  pinaka K
                                                    K[j][i] = value; //pinakas K
simmetrikos etsi ginetai apothikeusi timis sto katw trigwniko pinaka K

                                                    concat(pik, i, j, r, value,
outp1, outp2, K); //kaleitai i sinartisi concat gia dimiourgia tis antikeimenikis sinartisis

                                                }

                                            }

                                        }

                                    }

//an dothike ws eisodo sto menu epilogwn sisxetisewn twn stoixeiwn i epilogi 1 tote ta
stoixeia tha sindeontai me akeraious arithmous apo to -1 ews 1
                                    if (ans == 2) { //me dedomena apo -1-1

//////////////////////////////////////////////////////////////////////////
                                        for (i = 1; i <= r; i++) {
                                            for (j = 1; j <= r; j++) {

                                                if (i == j) {
                                                //sti diagwnio vazoume 0
                                                    value = 0;
                                                }
```

```
                                        if (thelp[i][j] == 1) {   //an ston
```
pinaka thelp [i][j] exei 1 tote ta i,j sindeontai kai etsi ipologizetai enas akeraios random arithmos apo -1 mexri 1

```
                                                value = (rand() % (max + 1
```
- min)) + min; //epistrefetai akeraios apo -1 mexri 1

```
                                                do {
                                                        value = (rand() %
```
(max + 1 - min)) + min;

```
                                                        if (value == 0)
```
{//enosw epistrefei 0 tote ksanakaleitai i sinartisi

```
                                                                value =
```
(rand() % (max + 1 - min)) + min; //epistrefetai akeraios apo -1 mexri 1

```
                                                        }

                                                } while (value == 0);

                                                K[i][j] = value;//apothikeusi
```
timis ston anw trigwniko pinaka K

```
                                                K[j][i] = value;//pinakas K
```
simmetrikos etsi ginetai apothikeusi timis sto katw trigwniko pinaka K

```
                                                concat(pik, i, j, r, value,
```
outp1, outp2, K); //kaleitai i sinartisi concat gia dimiourgia tis antikeimenikis sinartisis

```
                                        }

                                }

                        }
                }
```

//grapsimo sta arxeia minisat+ kai bsolo

```
                fprintf(outp1, "%s", "\n");
                fprintf(outp2, "%s", "\n");

                i = 0;
                //dimiourgia periorismwn monadikoitas gia ta arxeia
minisat+ kai bsolo .Se ena apo ta dio sinola mia metavliti, oxi kai sta dio
                for (i = 1; i <= r; i++) {
                        sprintf(stra, "%d", i);
                        sprintf(strb, "%d", i);
                        strcpy(buffer2a, "x");
                        strcpy(buffer3a, "x");

                        strcat(buffer2a, stra);
                        strcat(buffer3a, stra);
                        strcat(buffer2a, "1");
                        strcat(buffer3a, "2");

                        fprintf(outp1, "%s", "1*");
                        fprintf(outp1, "%s", buffer2a);
                        fprintf(outp1, "%s", "+");
                        fprintf(outp1, "%s", "1*");
                        fprintf(outp1, "%s", buffer3a);
                        fprintf(outp1, "%s", "=");
                        fprintf(outp1, "%s", "1");
                        fprintf(outp1, "%s", ";");
                        fprintf(outp1, "%s", "\n");

                        fprintf(outp2, "%s", " +1 ");
                        fprintf(outp2, "%s", buffer2a);
                        fprintf(outp2, "%s", " +");
                        fprintf(outp2, "%s", "1 ");
                        fprintf(outp2, "%s", buffer3a);
                        fprintf(outp2, "%s", " = ");
```

```
                fprintf(outp2, "%s", " 1");
                fprintf(outp2, "%s", " ;");
                fprintf(outp2, "%s", "\n");

        }

        for (i = 1; i <= r; i++) {
                for (j = 1; j <= r; j++) {
                        if (i < j) {
                                if (K[i][j] != 0)
                                        var = var + 1; //metavliti gia
ipologismo sinolo metavlitwn
                        }
                }
        }

        concat2(i, j, r, value, outp1, outp2, K); //kalesma
sinartisew gia tin dimiourgia twn ipoloipwn periorismwn
        concat3(pik, r, outp3, K, diagwnios, var); //kalesma
sinartisis concat3 gia tin dimiourgia tou arxeiou me touw periorismous ston npSolver

////////////////////////////////////////////////////////////

        outp = fopen("final_file.dzn", "w"); //anoigma arxeiou
final_file.dzn" gia grapsimo

        fprintf(outp, "n=");
        fprintf(outp, "%d", r);
        fprintf(outp, ";");
        fprintf(outp, "\n");

        fprintf(outp, "table="); //tipwnete o pinakas me tis
sisxetiseis twn sttoixeiwn stin morfi pou anagnwrizei to montelo .mzn
```

```
fprintf(outp, "  ");
fprintf(outp, "[");
for (i = 1; i <= r; i++) {
        fprintf(outp, "|");
        for (j = 1; j <= r; j++) {

                fprintf(outp, "%d", K[i][j]);

                if (i == j && j == (r)) {
                        fprintf(outp, "");
                } else if (j != (r)) {
                        fprintf(outp, ",");
                }

                else {
                        fprintf(outp, ",\n");
                }

        }

        if (!(i == j && j == (r))) {
                fprintf(outp, "      ");
        }
        printf("\n");

}

fprintf(outp, "|]");

fclose(outp); //kleisimo arxeiou

return 0;
}
```