Ατομική Διπλωματική Εργασία

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΔΡΑΣΗΣ ΜΕ ΧΡΗΣΗ ΨΕΥΔΟΔΥΑΔΙΚΗΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

Μαριλένη Δικωμίτου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Επίλυση προβλημάτων προγραμματισμού δράσης με χρήση ψευδοδυαδικής βελτιστοποίησης

Μαριλένη Δικωμίτου

Επιβλέπων Καθηγητής
Δρ. Γιάννης Δημόπουλος

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

Μάιος 2010

Ευχαριστίες

Θα ήθελα καταρχήν, να ευχαριστήσω τον επιβλέποντα καθηγητή, Δρ. Γιάννη Δημόπουλο, για την συνεχή βοήθεια και την στήριξη που μου έχει προσφέρει κατά την διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω ιδιαιτέρως την οικογένεια μου, που με στήριξε και με ενθάρρυνε να προχωρήσω.

Λίστα Ακρωνύμων

STRIPS (Stanford Research Institute Problem Solver): είναι η βασική γλώσσα αναπαράστασης ενός κλασσικού σχεδιαστή.

Καλά ορισμένες φόρμουλες(well formed formulas): είναι λέξεις που απαρτίζουν μια τυπική γλώσσα.

Όρος(Literal) : είναι ο όρος x ή η άρνηση του όρου $\neg x$

Πρόταση (Clause): είναι μια διάζευξη από όρους $x \vee y \vee z \vee \neg w$

Unit Clause κανόνας / Unit Propagation: αν υπάρχει μια unit clause σε μια CNF φόρμουλα F τότε δημιουργείται μια καινούργια φόρμουλα F' διαγράφοντας όλα τα clauses της F που περιέχουν τη unit clause. Αν η φόρμουλα F' είναι άδεια τότε η φόρμουλα F δεν είναι ικανοποιήσιμη.

Boolean constraint propagation: η επαναλαμβανόμενη εφαρμογή του κανόνα unit clause.

Περίληψη

Σε αυτή τη διπλωματική εργασία μελετάται το πρόβλημα του προγραμματισμού δράσης. Το πρόβλημα του προγραμματισμού δράσης αποτελεί μέρος της ερευνητικής μελέτης της τεχνητής νοημοσύνης. Στόχος του είναι να εντοπίσει μια ακολουθία από δράσεις όπου από την αρχική κατάσταση ενός προβλήματος, μεταβαίνει στο στόχο του και στην επίλυση του. Η επίλυση του προβλήματος του προγραμματισμού δράσης με χρήση βασικών αλγόριθμων δεν έφερε επαρκή αποτελέσματα. Αυτό συνέβαινε επειδή τα εργαλεία κατακλύζονταν με αχρείαστες και περιττές δράσεις. Τότε ο αλγόριθμος αναζήτησης αναγκαζόταν να ελέγχει το αποτέλεσμα τις κάθε δράσης για να βρει ποία ικανοποιεί το στόχο.

Για την καλύτερη επίλυση του προβλήματος του προγραμματισμού δράσης έγινε η διάσπαση του σε καταστάσεις, πράξεις και στόχους. Αυτό βοηθούσε τους αλγορίθμους να εκμεταλλεύονται πλήρως την λογική δομή του προβλήματος. Βοηθητικό εργαλείο για την πιο πάνω διάσπαση συντέλεσε η γλώσσα STRIPS. Η γλώσσα STRIPS είναι αρκετά εκφραστική και μπορεί να αναπαραστήσει ένα ευρύ φάσμα προβλημάτων. Είναι δομημένη γλώσσα, γεγονός που επιτρέπει σε αποδοτικούς αλγορίθμους να την διαχειρίζονται.

Κατά την διάρκεια των χρόνων μέσα από αρκετούς αλγόριθμους πραγματοποιήθηκε προσπάθεια για εύρεση αποδοτικότερων λύσεων για τα προβλήματα. Τους αλγόριθμους αυτούς διαδέχθηκε το σύστημα SatPlan όπου ήταν στην πραγματικότητα ένα σύνολο συμβάσεων για την κωδικοποίηση προβλημάτων γραμμικού περιορισμού της μορφής STRIPS, σε προτασιακό σχήμα δράσεων.

Στην παρούσα εργασία μελετάται η επίλυση προβλημάτων προγραμματισμού δράσης με χρήση ψευδοδυαδικής βελτιστοποίησης. Στους

ψευδοδυαδικούς αλγόριθμους βελτιστοποίησης στόχος είναι η εύρεση μιας ανάθεσης τιμών η οποία θα ελαχιστοποιεί το κόστος της συνάρτησης αλλά ταυτοχρόνως ικανοποιεί και όλους τους περιορισμούς. Βασική πρόκληση ήταν η δημιουργία μιας συνάρτησης βελτιστοποίησης που στόχο έχει την βελτιστοποίηση των λύσεων αλλά και την εύρεση της βέλτιστης λύσης σε πιο αποδοτικό χρόνο.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

1. Εισαγωγή

1. Εισαγωγή

Η συνεχής ανάπτυξη και μελέτη της τεχνητής νοημοσύνης προϋποθέτει τη συνεχή εύρεση καινούργιων μεθόδων αλλά και αλγόριθμων για εύρεση λύσεων με όση το δυνατό καλύτερη απόδοση. Μέρος της έρευνας της τεχνητής νοημοσύνης είναι και το πρόβλημα του προγραμματισμού δράσης που στοχεύει στην επίτευξη της δημιουργίας αυτόνομου έλεγχου σε περίπλοκα συστήματα.

Στην διάρκεια της μελέτης του κλασσικού προβλήματος του προγραμματισμού δράσης, παρατηρήθηκε ότι είναι μια μορφή αφαιρετικής λογικής. Οι πρώτες προσπάθειες για επίλυση του χρησιμοποιούσαν γενικούς αλγόριθμους, όμως η λύση ήταν αδύνατη. Η έρευνα μεταφέρθηκε στη συνεχεία και επικεντρώθηκε στην δημιουργία εξειδικευμένων αλγορίθμων.

Το γεγονός ότι χρειάζονται εξειδικευμένοι αφαιρετικοί αλγόριθμοι αμφισβητήθηκε από τους Kautz και Selman[7] όταν το σύστημα τους το ονομαζόμενο SATPLAN έδειξε ότι η μετάφραση του προβλήματος δράσης σε ένα πρόβλημα προτασιακής λογικής μπορούσε πραγματικά να ανταγωνιστεί τα καλύτερα εξειδικευμένα συστήματα επίλυσης του προβλήματος δράσης. Η επιτυχία του εργαλείου επίλυσης SATPLAN οφειλόταν στους πιο κάτω λόγους. Ένας λόγος ήταν η χρήση λογικών αναπαραστάσεων που είχε ως συνέπεια καλύτερες υπολογιστικές ιδιότητες. Επίσης κάνει χρήση της προτασιακής λογικής το οποίο είναι πολύ σημαντικό. Αποδοτικότητα προσφέρουν και οι διαφοροποιημένες δηλωτικές προτάσεις, που είναι σημασιολογικά ισοδύναμες.

Αρκετοί είναι οι ερευνητές όπου κάθε χρόνο προσπαθούν να δημιουργήσουν ταχύτερα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής. Έχουν εφαρμόσει κοινές τυποποιημένες αναπαραστάσεις οι οποίες επιτρέπουν την σύγκριση των αλγόριθμών με την χρήση ίδιων αποτελεσμάτων. Αυτό έχει ως αποτέλεσμα ένα εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογική να είναι πιο γρήγορο από

οποιανδήποτε εξειδικευμένη μηχανή επίλυσης προβλημάτων προγραμματισμού δράσης.

Μια προσέγγιση που έχει κοινά χαρακτηριστικά με τη μέθοδο επίλυσης SATPLAN είναι το Graphlan σύστημα που αναπτύσσεται ανεξάρτητα από τους Blum και Furst το 1995[2]. Το Graphplan κατέρριψε όλα τα προηγούμενα ρεκόρ σε θέμα ταχύτητας και έγινε ένα από τα πιο δημοφιλή πλαίσια εργασίας του προβλήματος προγραμματισμού δράσης. Μέσα από τις συγκρίσεις του αλγόριθμου του SATPLAN και του GRAPHPLAN κατέληξαν ότι και τα δύο συστήματα είναι εξίσου αποδοτικά. Ο SATPLAN είναι ταχύτερος σε κάποια πεδία ενώ σε κάποια άλλα ταχύτερος ήταν ο Graphplan.

Η επόμενη προσέγγιση ήταν η επίλυσή των προβλημάτων προγραμματισμού δράσης με την χρήση των ψευδοδυαδικών αλγορίθμων. Με αυτή την μέθοδο οι ερευνητές στοχεύουν να βελτιώσουν σε σημαντικό βαθμό τις λύσεις. Στους ψευδοδυαδικούς αλγόριθμους ενσωματώνονται και οι αλγόριθμοι βελτιστοποίησης. Στόχος των αλγορίθμων βελτιστοποίησης είναι η εύρεση και η ανάθεση τιμών στις μεταβλητές που θα ικανοποιεί όλους τους περιορισμούς αλλά θα επιφέρει και αποδοτικότερη λύση.

ΚΕΦΑΛΑΙΟ 2

2. Το πρόβλημα προγραμματισμού δράσης και βασικοί Αλγόριθμοι

2.1. Γνωσιολογικό Υπόβαθρο

2.1.1. Εισαγωγή

2.1.2. Αναπαράσταση προβλήματος προγραμματισμού δράσης

2.2. Μοντελοποίηση προβλημάτων προγραμματισμού δράσης σε γλώσσα Strips

2.3. Αλγόριθμοι του προγραμματισμού δράσης

2.3.1. Προς τα εμπρός αναζήτηση στο χώρο (Forward space search)

2.3.2. Προς τα πίσω αναζήτηση στο χώρο(Backward state –space search)

2.3.3. Ευρετικές συναρτήσεις για αναζήτηση στο χώρο (Heuristics for state-space search).

2.3.4. Μελέτη του Partial Order Planner

2.3.5. Ευρετικές συναρτήσεις για Partial Order Planner

2.1 Γνωσιολογικό Υπόβαθρο

2.1.1 Εισαγωγή

Η επίλυση του προβλήματος προγραμματισμού δράσης αποτελεί ένα ερευνητικό κομμάτι της τεχνητής νοημοσύνης. Στοχεύει την επίτευξη της δημιουργίας αυτόνομου έλεγχου σε περίπλοκα συστήματα. Ουσιαστικά ο στόχος του προβλήματος προγραμματισμού δράσης είναι να εντοπίσει μια ακολουθία από δράσεις. Με την χρήση της ακολουθίας αυτής το πρόβλημα προγραμματισμού δράσης να οδηγείται από την αρχική κατάσταση του, στην λύση του. Αρκετοί αλγόριθμοι του προβλήματος προγραμματισμού δράσης έχουν εφαρμοστεί επιτυχώς σε τομείς όπως η ρομποτική, σε συλλογή πληροφοριών, σε έλεγχο εκτόξευσης πυραύλων κ.α.[14]

Το πιο σημαντικό στην έρευνα του προβλήματος προγραμματισμού δράσης είναι η δυνατότητα ταξινόμησης των προβλημάτων με βάση την πολυπλοκότητα της λύσης τους. Η ταξινόμηση του ανάλογα με την πολυπλοκότητα της λύσης του επιτρέπει την επιλογή του σωστού εργαλείου για την επίλυσή του. Οι δύο, πιο σημαντικές κατηγορίες προβλημάτων είναι το πρόβλημα δημιουργίας σχεδίου δράσης και το πρόβλημα της ύπαρξης σχεδίου δράσης. Το πρόβλημα δημιουργίας σχεδίου δράσης, δημιουργεί μια σειρά από πράξεις, για να φτάσει το στόχο. Το πρόβλημα της ύπαρξης σχεδίου δράσης προσδιορίζει κατά πόσο υπάρχει μια τέτοια ακολουθία που οδηγεί στο στόχο.

Κατά την επίλυση τέτοιων προβλημάτων με βασικούς αλγορίθμους όπως είναι οι Depth first, A* κ.α. παρατηρήθηκε ότι υπήρχαν αρκετές δυσκολίες. Αυτό συνέβαινε επειδή τα εργαλεία κατακλύζονταν με αχρείαστες και περιττές δράσεις. Τότε ο αλγόριθμος αναζήτησης αναγκαζόταν να ελέγχει το αποτέλεσμα τις κάθε δράσης για να βρει ποία ικανοποιεί το στόχο.

Ένα παράδειγμα που υποδηλώνει τις δυσκολίες της επίλυσης του προβλήματος προγραμματισμού δράσης με τους προαναφερθέντες αλγόριθμους

είναι το ακόλουθο: Έστω η προσπάθεια αγοράς ενός κινητού τηλεφώνου μέσω μιας διαδικτυακής σελίδας. Κάθε κινητό έχει ένα μοναδικό αριθμό κατασκευής που αντιστοιχεί σε μια και μόνο αγορά συνεπώς και σε μια πράξη. Εάν τώρα η πράξη αυτή λαμβάνει μέρος σε ένα σύνολό από 5 δισεκατομμύρια πράξεις ο αλγόριθμος αναζήτησης πρέπει να ελέγξει το αποτέλεσμα όλων αυτών το πράξεων μέχρι να βρει αυτό που ικανοποιεί τη αγορά ενός κινητού με ένα συγκεκριμένο αριθμό κατασκευής.

Όλες αυτές οι πράξεις μπορούσαν να αποφευχθούν εάν ο αλγόριθμός λειτουργούσε ανάποδα. Αν, δηλαδή ήταν γνωστός ο αριθμός κατασκευής του κινητού τηλεφώνου, θα έψαχνε να βρει κατά πόσο το συγκεκριμένο κινητό τηλέφωνο ήταν διαθέσιμο, και αναλόγως της απάντησης θα ενεργοποιούσε την πράξη της αγοράς.

Επίσης μια ακόμη δυσκολία, αποτέλεσε η εύρεση μιας ευρετικής συνάρτησης (heuristic function) που ελαχιστοποιεί το ψάξιμο των πιθανών λύσεων του προβλήματος. Με την χρήση της ευρετικής συνάρτησης τώρα δημιουργήθηκαν επιπλέον δυσκολίες, μια από αυτές είναι η έλλειψη της αυτονομίας που έχει η μέθοδος επίλυσης. Η ευρετική συνάρτηση για κάθε νέο πρόβλημα πρέπει να δίνεται πλέον από τον χρήστη. Αν η μεθόδου επίλυσης έχει πρόσβαση σε μια σαφή εκπροσώπηση του στόχου σαν μια σύζευξη των υποστόχων του, τότε θα μπορεί να χρησιμοποιεί ένα και μόνο ανεξάρτητο πεδίο ευρετικών συναρτήσεων που είναι ο αριθμός ανικανοποίητων συζεύξεων.

2.1.2 Αναπαράσταση προβλήματος προγραμματισμού δράσης

Η αναπαράσταση ενός προβλήματος προγραμματισμού δράσης και η διάσπαση του σε καταστάσεις, πράξεις και στόχους βοηθά τους αλγορίθμους να εκμεταλλεύονται πλήρως την λογική δομή του προβλήματος. Σημαντικό είναι η εύρεση μιας γλώσσας η οποία μπορεί να αναπαραστήσει πλήρως ένα σύνολο από

προβλήματα, αλλά να είναι επίσης δομημένη, έτσι ώστε να επιτρέπει σε αποδοτικούς αλγορίθμους να την διαχειρίζονται

Η βασική γλώσσα αναπαράστασης ενός κλασσικού σχεδιαστή (classical planner)είναι γνωστή σαν STRIPS (Stanford Research Institute Problem Solver) γλώσσα. Η γλώσσα STRIPS είναι ένα μέρος των εργαλείων που επιλύουν το πρόβλημα του προγραμματισμού δράσης μέσω αναζήτησης στις αναπαραστάσεις του κόσμου για την εύρεση ενός μοντέλου που υλοποιεί το στόχο. Για κάθε μοντέλο του κόσμου θεωρείται ότι υπάρχει ένα σύνολο από εφαρμόσιμους τελεστές, όπου καθένας από αυτούς μετατρέπει το μοντέλο του κόσμου σε κάποιο άλλο μοντέλο κόσμου.

Ο σκοπός του εργαλείου που επιλύει το πρόβλημα (problem solver) είναι να βρει μια σύνθεση από τελεστές οι οποίοι να μετατρέπουν την αρχική κατάσταση του μοντέλου του κόσμου, στην επιθυμητή κατάσταση, αυτή δηλαδή που ικανοποιεί τη συνθήκη του στόχου.

Η γλώσσα STRIPS παρουσιάζει ένα μοντέλο του κόσμου με ένα σύνολο από καλά ορισμένους τύπους(well formed formulas). [13,16]

Η αναπαράσταση ενός κόσμου είναι η εξής: έστω υπάρχει ένα άνθρωπος ο οποίος στέκεται σε μια τοποθεσία a και υπάρχουν και δύο αυτοκίνητα B και C τοποθετημένα σε τοποθεσίες b και c αντίστοιχα. Το σύνολο των καλά ορισμένων τύπων θα ήταν το ακόλουθο: η τοποθεσία του ανθρώπου θα ερμηνευόταν σαν ATH(a) και των δυο αυτοκινήτων θα ερμηνευόταν σαν AT(B,b) και AT(C,c).

Στη γλώσσα STRIPS όταν ένας τελεστής εφαρμοστεί σε ένα μοντέλο κόσμου θα έχουν οριστεί ήδη οι παράμετροι του. Ένας τελεστής χαρακτηρίζεται με δύο κύριους τρόπους. Έχει την περιγραφή των αποτελεσμάτων που θα πραγματοποιηθούν εάν εκτελεστεί και την περιγραφή των συνθηκών κάτω από τις οποίες είναι εφαρμόσιμος. Τα αποτελέσματα του είναι απλά ορισμένα με μια

λίστα από καλά ορισμένους τύπους οι οποίοι ισχύουν και πρέπει να προστεθούν στο μοντέλο, και από μια λίστα καλά ορισμένων τύπων που πρέπει να αφαιρεθούν επειδή δεν θα ισχύουν πλέον. Εκτός από τα αποτελέσματα και τις συνθήκες σε καλά ορισμένους τύπους γράφονται και οι στόχοι.

Ουσιαστικά ένα πρόβλημα που είναι γραμμένο σε γλώσσα STRIPS αποτελείται από τρεις οντότητες. Αρχικά υπάρχει η αναπαράσταση των καταστάσεων όπου είναι το σπάσιμο του κόσμου σε λογικές συνθήκες και η κατάσταση αναπαριστάται ως ένας συνδυασμός θετικών όρων (literals). Ακολούθως είναι η αναπαράσταση των στόχων που υποδηλώνει ότι, ένας στόχος είναι μια μερική κατάσταση η οποία αναπαριστάται σαν συνδυασμός θετικών όρων. Η αναπαράσταση των πράξεων ονομάζεται σχήμα δράσεων(action schema) και αποτελείται από τρία μέρη, την πράξη, τις προϋποθέσεις οι οποίες πρέπει να πληρούνται πριν την εκτέλεση της πράξης και συμπεριλαμβάνονται στους παραμέτρους της και το αποτέλεσμα το οποίο περιλαμβάνει ποια στοιχεία έχουν πάρει την τιμή αληθής ή ψευδής για την επόμενη κατάσταση. Το πρόβλημα επιλύεται όταν παραχθεί ένα μοντέλο το οποίο ικανοποιεί το στόχο.

2.2 Μοντελοποίηση προβλημάτων προγραμματισμού δράσης σε γλώσσα Strips

Εδώ θα μελετηθεί πως ένα πρόβλημα προγραμματισμού δράσης μπορεί εύκολα να αναπαρασταθεί σε γλώσσα STRIPS. Το ακόλουθο παράδειγμα παρουσιάζει τη μεταφορά του προβλήματος προγραμματισμού δράσεων στις τρείς οντότητες της γλώσσα STRIPS .[9, 13.16]

Παράδειγμα 2.2.1

Έστω ότι σε ένα δωμάτιο υπάρχει ένα παιδί και σε ένα ψηλό ράφι μερικά γλυκά. Επίσης στο δωμάτιο υπάρχει και μια καρέκλα που θα βοηθήσει το παιδί να φτάσει τα γλυκά. Η αρχική θέση του παιδιού είναι το Α τα γλυκά στο Β και η

καρέκλα στο C. Το αρχικό ύψος του παιδιού και της καρέκλας είναι χαμηλό ενώ των γλυκών υψηλό. Οι πράξεις που είναι διαθέσιμες στο παιδί είναι να μετακινείται από μια θέση σε μια άλλη να μετακινεί πράγματα, να σκαρφαλώνει και να παίρνει αντικείμενα.

Αρχική κατάσταση

At(Child,a)

At(Jar,b)

At(Chair,c)

Height(Child,Low)

Height(Chair,Low)

Height(Jar,High)

Move(Chair)

Climb(Chair)

Grasp(Jar)

Open(Jar)

Grasp(Candy)

Στόχος

Have(Child, Candy)

Τελεστές

Go(x,y)

Precond: At(Child,x) $\wedge$ Height(Child,Low)

Effect: At(Child,y) $\wedge$ ~At(Child,x)

Move(b,x,y)

Precond: At(Child,x) $\wedge$ Height(Child,Low) $\wedge$ At(b,x) $\wedge$ Move(b) $\wedge$ Height(b,Low)

Effect: At(b,y) $\wedge$ At(Child,y) $\wedge$ ~At(b,x) $\wedge$ ~ At(Child,x)

ClimbUp(b)

Precond: At(Child,x) ∧ Height(Child,Low) ∧ At(b,x) ∧Climb(x) ∧ Height(b,Low)

Effect: On(Child,b) ∧ ~ Height(Child,Low) ∧ Height(Child,High)

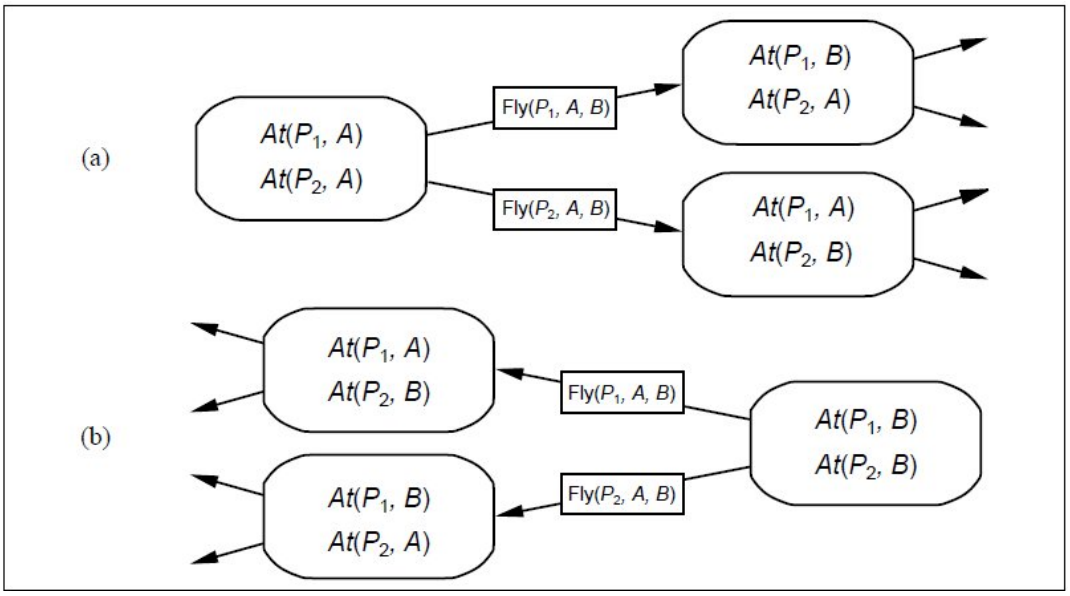
Get(b)

Precond: At(Child,x) ∧ Height(Child,h) ∧ At(b,x) ∧ Grasb(b) ∧ Height(b,h)

Effect: Have(Child,b)

2.3 Αλγόριθμοι του για προβλήματα προγραμματισμού δράσης

Οι περιγραφές των πράξεων σε ένα πρόβλημα προγραμματισμού δράσης ορίζουν τις προϋποθέσεις και τα αποτελέσματα έτσι είναι πιθανόν το ψάξιμο για την λύση να γίνει προς οποιανδήποτε κατεύθυνση, είτε από την αρχική κατάσταση προς το στόχο, είτε από το στόχο πίσω στην αρχική κατάσταση.



Εικόνα 1: Οι δυο προσεγγίσεις αναζήτησης στο χώρο

a) Προς τα εμπρός αναζήτηση στο χώρο

b) Προς τα πίσω αναζήτηση στο χώρο

2.3.1 Προς τα εμπρός αναζήτηση στο χώρο (Forward Space Search)

Στην αναζήτηση προς τα εμπρός στο χώρο(Forward Space Search), το πρόβλημα ξεκινά από την αρχική κατάσταση και με μια αλληλουχία πράξεων, βρίσκεται η σωστή ακολουθία που καταλήγει στο στόχο.[14,15]

Η διατύπωση του προβλήματος προγραμματισμού δράσης σε προβλήματα αναζήτησης στο χώρο (State - Space Search) αποτελείται από την αρχική κατάσταση, το σύνολο των πράξεων, την συνάρτηση η οποία ελέγχει πότε μια κατάσταση ικανοποιεί το στόχο και το κόστος του κάθε βήματος που ως συνήθως είναι 1. Η απουσία των συναρτησιακών συμβόλων καθιστά το πρόβλημα πεπερασμένο.

Η αναζήτηση προς τα εμπρός στο χώρο(Forward Space Search) δεν ήταν αποδοτική για το λόγο ότι εξέταζε όλες τις περιπτώσεις. Σε αυτή την αναζήτηση, σε κάθε κατάσταση λαμβάνονται υπ' όψιν όλες οι εφαρμοζόμενες πράξεις. Το πρόβλημα φτάνει γρήγορα σε τέλμα εάν δεν υπάρχει μια καλή ευρετική συνάρτηση (heuristic function). Αυτό είναι απολύτως λογικό για το λόγο ότι, εάν για παράδειγμα σε μια πιθανή λύση ενός προβλήματος, κάθε κατάσταση είχε γύρω στις 1000 πράξεις και όλο το πρόβλημα αποτελείτο από n καταστάσεις, τότε το βάθος του δένδρου αναζήτησης θα ήταν της τάξεως 1000^n. Έτσι χωρίς την χρήση μιας καλής ευρετικής συνάρτησης (heuristic function), το ψάξιμο μέσα από το δένδρο, για να φτάσουμε στο στόχο, δεν θα ήταν αποδοτικό.

2.3.2Προς τα πίσω αναζήτηση στο χώρο(Backward State – Space Search)

Στην αναζήτηση προς τα πίσω στο χώρο το πρόβλημα ξεκινά από την τελική κατάσταση, το στόχο δηλαδή, και με μια αλληλουχία πράξεων προσπαθεί να φτάσει στην αρχική κατάσταση.[14,15]

Η αναζήτηση προς τα πίσω (Backward State Space Search) είναι δύσκολό να υλοποιηθεί όταν η κατάσταση του στόχου περιγράφεται με ένα σύνολο από περιορισμούς και όχι ρητά. Επίσης δεν είναι πάντοτε εμφανή η δημιουργία των προηγούμενων καταστάσεων που οδηγούν σε μια κατάσταση στόχου.

Το κύριο πλεονέκτημα της αναζήτησης είναι ότι αποφεύγει αχρείαστες πράξεις και εξετάζει μόνο αυτές οι οποίες σχετίζονται με το στόχο, δηλαδή πετυχαίνουν ένα μέρος του στόχου. Αρκετές φορές όμως μη συσχετιζόμενες πράξεις μπορεί να οδηγήσουν στη λύση όμως αν ο αλγόριθμος εξέταζε και αυτές τις πράξεις δεν θα ήταν πλέον αποδοτικός.

2.3.3 Ευρετικές συναρτήσεις για αναζήτηση στο χώρο

Όπως έχει αναφερθεί πιο πάνω οι δύο αλγόριθμοι προς τα πίσω αναζήτηση στο χώρο και προς τα εμπρός αναζήτηση στο χώρο δεν είναι αποδοτικοί εάν δεν έχουν μια καλή ευρετική συνάρτηση. Ρόλος της ευρετικής συνάρτησης είναι ο υπολογισμός της απόστασης από την παρούσα κατάσταση στο στόχο.

Στη βασική γλώσσα STRIPS είναι γνωστό ότι το κόστος κάθε πράξης είναι 1, έτσι εδώ η απόσταση είναι ο συνολικός αριθμός των πράξεων. Βασική ιδέα είναι να βρεθούν τα αποτελέσματα που δίδει η κάθε πράξη, οι στόχοι οι οποίοι πρέπει να επιτευχθούν και να υπολογιστεί πόσες πράξεις πρέπει να λάβουν χώρα ούτως ώστε να επιτευχθεί ο στόχος. Ο εντοπισμός όμως του ακριβής αριθμού καταστάσεων που χρειάζεται να επιλυθεί ένα πρόβλημα αποτελεί πρόβλημα NP -Hard, αλλά μπορούν να βρεθούν αρκετές προσεγγίσεις χωρίς μεγάλους υπολογισμούς.

Υπάρχουν δύο προσεγγίσεις, η πρώτη είναι η παραγωγή ενός προβλήματος όπου οι περιορισμοί δεν είναι τόσο απόλυτοι (relaxed problem) και η δεύτερη προσέγγιση η ονομαζόμενη ανεξαρτησία υποστόχων (sub goal independence). Η τελευταία προσέγγιση στηρίζεται στη λογική ότι το κόστος επίλυσης των συνδυασμένων στόχων είναι το άθροισμα του κόστους όταν ο κάθε υποστόχος επιλύεται ανεξάρτητα. Η προαναφερθείσα προσέγγιση μπορεί να φέρει καλά και κακά αποτελέσματα. Αν για παράδειγμα μια δράση σε ένα υποσχέδιο αναιρεί ένα στόχο που επιτεύχθηκε από κάποιο άλλο υποσχέδιο τότε η προσέγγιση μπορεί να φέρει καλά αποτελέσματα. Εάν τώρα υπάρχουν αχρείαστες ενέργειες σε υποσχέδια τότε φέρει κακά αποτελέσματα. Αχρείαστες ενέργειες είναι αυτές οι οποίες μπορούν να αντικατασταθούν από μια ενιαία πράξη.

2.3.4 Μελέτη του Partial Order Planner

Οι αλγόριθμοι αναζήτησης προς τα εμπρός και αναζήτησης προς τα πίσω συνήθως χρησιμοποιούνται σε αυστηρές γραμμικές ακολουθίες οι οποίες συνδέονται στην αρχή ή στον στόχο. Αυτό σημαίνει ότι δεν μπορεί να επωφεληθεί από την διάσπαση προβλημάτων σε υποπροβλήματα. Έτσι δεν μπορούν να δουλεύουν πάνω σε ένα υποπρόβλημα ξεχωριστά αλλά πρέπει να παίρνουν αποφάσεις για το πώς θα εκτελείται η ακολουθία των δράσεων για όλα τα υποπροβλήματα.[14,15]

Οποιοσδήποτε αλγόριθμος ο οποίος μπορεί να συνδυάσει δύο πράξεις σε ένα σχέδιο χωρίς να προσδιορίσει ποία πράξη έρχεται πρώτη ονομάζεται Partial Order Planner και η λύση του αναπαριστάται σαν ένα γράφημα από δράσεις και όχι μια σειρά από δράσεις.

Η λύση του Partial Order Planner αναφέρεται σε διάφορα πιθανά σχέδια που μπορούν όλα να επιτευχθούν. Κάθε πιθανό σχέδιο ονομάζεται linearization.

Κάθε πιθανό σχέδιο λύσης στο Partial Order Planner αποτελείται από τέσσερα βασικά μέρη. Πρώτο μέρος είναι το σύνολο των πράξεων που προσδιορίζουν τα βήματα του σχεδίου. Ένα άδειο σχέδιο περιλαμβάνει μόνο την αρχή και το τέλος. Η αρχή υποδηλώνει ότι δεν χρειάζονται προαπαιτούμενες πράξεις και έχει ως αποτέλεσμα της πράξης, όλους τους όρους (literals) στην αρχική κατάσταση του προβλήματος. Το τέλος δεν έχει κάποιο αποτέλεσμα αλλά έχει ως προαπαιτούμενα τα literals που υποδηλώνουν το στόχο του προβλήματος μας.

Δεύτερο μέρος είναι το σύνολο από μια σειρά περιορισμών διατομής(ordering constraints). Κάθε σειρά περιορισμών έχει την μορφή A < B και διαβάζεται ως εξής: η δράση A προηγείται της δράσης B. Με λίγα λόγια η δράση A κατά την εκτέλεση πρέπει να προηγηθεί της B, αλλά η B δεν είναι υποχρεωτικό να εκτελεστεί ακριβώς μετά από την A.

Τρίτο μέρος είναι το σύνολο από συνδέσεις αιτιότητας(causal links). Η σύνδεση αιτιότητας είναι μια δράση μεταξύ δύο δράσεων που πρέπει να παραμένει αληθής, μέχρι την εκτέλεση δυο δράσεων. Η γραφή των συνδέσεων

αιτιότητας είναι η ακόλουθη A $\xrightarrow{p}$ B και ερμηνεύεται ότι η δράση A υλοποιεί το p για την B. Η δράση p εκτελείται στην A και είναι προϋπόθεση της B.

Τέλος το τέταρτο μέρος είναι το σύνολο των ανοιχτών προϋποθέσεων (open precondition). Οι ανοιχτές προϋποθέσεις είναι αυτές οι οποίες δεν έχουν υλοποιηθεί από κάποια δράση.

Ένα σχέδιο λέγεται συνεπές (consistent) όταν σε αυτό δεν παρουσιάζονται κύκλοι κατά την εκτέλεση. Επίσης δεν υπάρχει σύγκρουση μεταξύ των συνδέσεων αιτιότητας. Ένα συνεπές πλάνο (consistent plan) που δεν έχει ανοιχτές προϋποθέσεις αποτελεί λύση.

Έχουμε το αρχικό πλάνο που δηλώνει σαφώς την αρχική και τελική κατάσταση χωρίς την ύπαρξη συνδέσεων αιτιότητας και όλοι οι περιορισμοί που πρέπει να υλοποιηθούν είναι στο τέλος με την μορφή των ανοιχτών προϋποθέσεων. Στην συνέχεια διαλέγεται μια ανοιχτή προϋπόθεση για μια δράση και παράγει ένα πλάνο για κάθε πιθανή επιλογή μιας δράσης. Στο τέλος ελέγχεται εάν το πλάνο που δημιουργείται αποτελεί λύση για το αρχικό πρόβλημα.

2.3.5 Ευρετικές συναρτήσεις για Partial Order Planner

Ο αλγόριθμος του Partial Order Planner και η δυνατότητα του να διασπά τα προβλήματα σε επιμέρους υποπροβλήματα αποτελεί το μεγαλύτερο του πλεονέκτημα. Μειονεκτεί όμως, γιατί δεν μπορεί να παραστήσει απευθείας τις καταστάσεις πράγμα που καθιστά δύσκολο τον υπολογισμό της απόστασης που βρίσκεται ο Partial Order Planner από την υλοποίηση του στόχου. Προς το παρών ο υπολογισμός και η εύρεση μια καλής και ακριβής ευρετικής συνάρτησης δεν έχει βρεθεί. Το πιο προφανές είναι η ευρετική συνάρτηση να μετράει τον αριθμό των ξεχωριστών προϋποθέσεων και να βελτιώνεται στην συνέχεια με την αφαίρεση όλων των ανοιχτών προϋποθέσεων που ταιριάζουν με κάποιο όρο κατά την αρχική κατάσταση. Με αυτό το τρόπο υπερεκτιμάται το κόστος όταν υπάρχουν δράσεις που πετυχαίνουν πολλαπλούς στόχους και υποτιμάται το κόστος όταν υπάρχουν αρνητικές αλληλεπιδράσεις μεταξύ των καταστάσεων του σχεδίου.

Η ευρετική συνάρτηση χρησιμοποιείται για να επιλέξει ποιο σχέδιο θα τελειοποιήσει. Δεδομένης αυτής της επιλογής, ο αλγόριθμος δημιουργεί διαδόχους βασιζόμενος στην επιλογή μιας ανοικτής προϋπόθεσης. Η επιλογή αυτή έχει μεγάλη επίπτωση στην αποδοτικότητα, όπως και στην περίπτωση επιλογής της μεταβλητής στους αλγορίθμους ικανοποίησης περιορισμών. Η πιο περιορισμένη μεταβλητή για την ευρετική συνάρτηση από το πρόβλημα ικανοποίησης περιορισμών μπορεί να υιοθετηθεί από τους αλγόριθμους του προγραμματισμού δράσης και να αποδώσει και σε καλό βαθμό.

Η ιδέα είναι να επιλεχθεί η ανοικτή προϋπόθεση που μπορεί να ικανοποιηθεί με τον λιγότερο αριθμό διαφορετικών τρόπων. Υπάρχουν δύο ειδικές περιπτώσεις της ευρετικής συνάρτησης. Πρώτον, εάν μια ανοικτή προϋπόθεση δεν μπορεί να επιτευχθεί από καμία πράξη, η ευρετική θα επιλέξει την ανοικτή αυτή προϋπόθεση. Η ιδέα αυτή είναι πολύ βοηθητική, διότι η έγκαιρη ανίχνευση ότι το πρόβλημα δε μπορεί να επιλυθεί, θα γλιτώσει τον αλγόριθμό από επιπλέον και αχρείαστη δουλεία. Δεύτερον, αν μια ανοιχτή κατάσταση μπορεί να επιτευχθεί μόνο με έναν τρόπο, τότε θα πρέπει να επιλέγεται διότι η απόφαση είναι αναπόφευκτη και μπορεί να παράγει επιπρόσθετους περιορισμούς που αφορούν άλλες επιλογές που ακόμη να εκτελεσθούν. Παρά το γεγονός ότι ο πλήρης υπολογισμός του αριθμού των τρόπων που ικανοποιούν κάθε ανοικτή προϋπόθεση, είναι ακριβός και δεν αξίζει πάντα τον κόπο, πειραματικές μελέτες δείχνουν ότι ο χειρισμός των δύο ειδικών περιπτώσεων παρέχουν πολύ σημαντικές επιταχύνσεις στο χρόνο επίλυσης.

ΚΕΦΑΛΑΙΟ 3

3.Γράφημα δράσεων (Planning Graphs)

3.1 Εισαγωγή

Αρκετές από τις ευρετικές συναρτήσεις οι οποίες έχουν προταθεί έχουν αποδειχθεί να είναι ανακριβείς. Σε αυτό ήρθε να βοηθήσει μια ειδική δομή, η λεγόμενη γράφημα δράσεων. Η δομή αυτή χρησιμοποιείται για να δίδει καλύτερες εκτιμήσεις για τις ευρετικές συναρτήσεις. Από την ειδική αυτή δομή, μπορούμε να εξάγουμε την λύση του προβλήματος με την χρήση του αλγορίθμου GraphPlan. [2, 14,]

Η ειδική δομή του γραφήματος δράσεων(planning graph) κωδικοποιεί τα προβλήματα προγραμματισμού δράσης με τέτοιο τρόπο, ώστε αρκετοί χρήσιμοι περιορισμοί που ενυπάρχουν στο πρόβλημα, να βοηθούν στη μείωση της αναζήτησης που χρειάζεται για να βρεθεί η λύση. Επίσης, η ειδική αυτή δομή κατασκευάζεται γρήγορα, έχει πολυωνυμικό μέγεθος αλλά δημιουργείται και σε πολυωνυμικό χρόνο.

Επίσης η δομή γραφήματος δράσεων προσφέρει ένα μέσο οργάνωσης και διατήρησης των πληροφοριών αναζήτησης. Σημαντικό ρόλο έχει και στην επίλυση των προβλημάτων προγραμματισμού δράσης, έστω και αν η πολυπλοκότητα του προγραμματισμού δράσεων σε γλώσσα STRIPS είναι PSPACE – hard.

Συγκεκριμένα μια δομή γραφήματος δράσεων περιέχει μια σειρά από επίπεδα τα οποία αντιστοιχούν σε χρονικά βήματα στο πλάνο. Έχει ως αρχική κατάσταση το επίπεδο 0. Ακολούθως κάθε επίπεδο περιέχει ένα σύνολο από δράσεις και ένα σύνολο από όρους(literals), οι οποίοι θα μπορούσαν να είναι αληθείς(True) στο συγκεκριμένο χρονικό σημείο. Οι όροι παίρνουν τις ανάλογες τιμές με βάση τις δράσεις που προηγήθηκαν στο προηγούμενο επίπεδο (προηγούμενο χρονικό βήμα).

Τα επίπεδα στο γράφημα δράσεων αποτελούνται από τέσσερα μέρη. Το πρώτο μέρος είναι οι προτάσεις που είναι αληθείς. Το δεύτερο μέρος από πιθανές δράσεις στη δεδομένη χρονική στιγμή. Το τρίτο μέρος από πιθανές προτάσεις που θα γίνουν αληθείς και το τελευταίο από πιθανές δράσεις που θα εκτελεστούν στην επόμενη χρονική στιγμή.

3.1.1 Παράδειγμα Γραφήματος Δράσεων

Init (Have (Diner))

Goal (Have (Diner) ∧ Eaten (Diner))

Action (Eat (Diner)

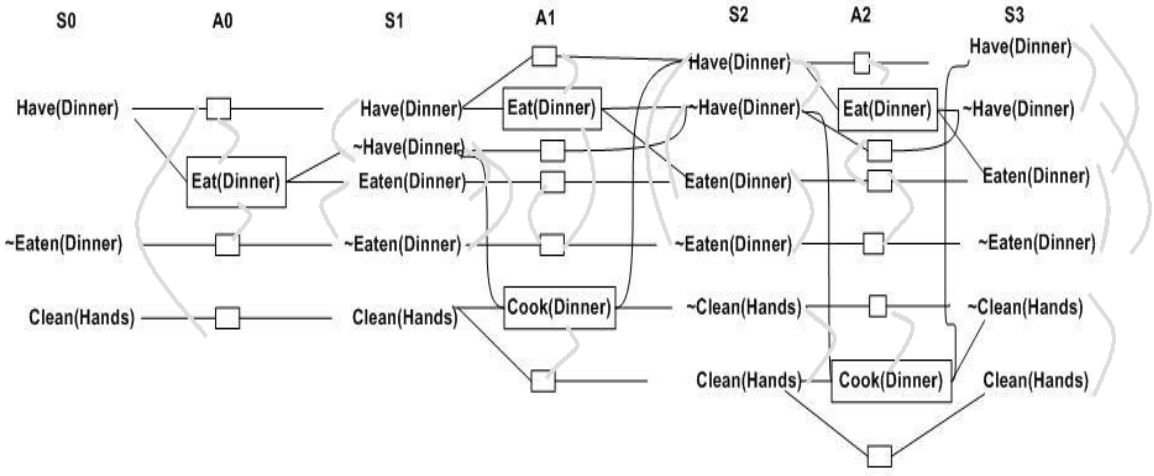
PRECONDITION: Have (Diner)

EFFECT: ~Have (Diner) ∧ Eaten (Diner))

Action (Cook (Diner))

PRECONDITION: Clean (Hands) ∧ ~Have (Diner)

EFFECT: Have (Diner)



Σχήμα 1 «Planning Graph»

Όπως παρουσιάζεται στην εικόνα 1 στο επίπεδο A0 περιέχονται όλες οι δράσεις που μπορούν να λάβουν μέρος στην κατάσταση S0. Στην συνέχεια στο επίπεδο S1 περιέχονται όλοι οι όροι(literals) που είναι στην ουσία, τα αποτελέσματα κάθε δράσης του επιπέδου A0. Αυτή η διαδικασία ανάμεσα στο επίπεδο καταστάσεων S και το επίπεδο των δράσεων A συνεχίζεται έως ότου φτάσουμε σε ένα επίπεδο οπού τα δύο διαδοχικά επίπεδα είναι τα ίδια.

3.2 Αλγόριθμος εξαγωγής λύσης από γραφήματα δράσεων

Ο αλγόριθμος εξαγωγής λύσης από γραφήματα δράσεων (GraphPlan) αποτελείται από δύο εναλλασσόμενες φάσεις. Την προς τα εμπρός φάση και την προς τα πίσω φάση. Στην προς τα εμπρός φάση η ειδική δομή του γραφήματος δράσεων επεκτείνεται σταδιακά σε αντίθεση με την προς τα πίσω φάση, που αναζητούμε στην ειδική δομή γραφήματος δράσεων την εξαγωγή ενός έγκυρου πλάνου(valid plan).[14,15]

Ο αλγόριθμός της εξαγωγής λύσης από γραφήματα δράσεων όπως παρουσιάζεται στο σχήμα 2 ξεκινά με την δημιουργία ενός γραφήματος δράσεων, που έχει ένα αρχικό επίπεδο. Στο επίπεδο αυτό περιλαμβάνονται όλες οι αρχικές προϋποθέσεις. Η εκτέλεση του αλγορίθμου γίνεται σε στάδια. Στο στάδιο k ο αλγόριθμός παίρνει το γράφημα δράσεων που δημιουργήθηκε από το προηγούμενο στάδιο, δηλαδή στο στάδιο k-1, και το επεκτείνει κατά ένα χρονικό βήμα. Ακολούθως ο αλγόριθμος ψάχνει το αναβαθμισμένο γράφημα να βρει πλάνο που ισχύει με μήκος k. Στην συγκεκριμένη στιγμή που ψάχνει ο αλγόριθμος είτε θα βρει ένα πλάνο το οποίο να ισχύει και θα το κρατήσει, είτε αποφασίζει ότι οι στόχοι δεν είναι όλοι εφαρμόσιμοι στην χρονική στιγμή k και προχωράει στο επόμενο στάδιο.

```
function GRAPHPLAN(problem) returns solution or failure

  graph ← INITIAL-PLANNING-GRAPH(problem)
  goals ← GOALS[problem]
  loop do
      if goals all non-mutex in last level of graph then do
          solution ← EXTRACT-SOLUTION(graph, goals, LENGTH(graph))
          if solution ≠ failure then return solution
          else if NO-SOLUTION-POSSIBLE(graph) then return failure
      graph ← EXPAND-GRAPH(graph, problem)
```

Σχήμα 2 «Αλγόριθμός GraphPlan»

3.3 Αλγόριθμος SatPlan

Η ιδέα της κωδικοποίησης του προβλήματος προγραμματισμού δράσης σαν πρόβλημα ικανοποιησιμότητας προτασιακής λογικής προτάθηκε αρχικά το 1992. Αρχικά η ιδέα αυτή χρησιμοποιήθηκε για να δημιουργεί ενδιαφέροντα προβλήματα ικανοποιήσιμοτητας προτασιακής λογικής, αλλά δε φάνηκε να είναι αποδοτική προσέγγιση για τα προβλήματα προγραμματισμού δράσης.

Στην συνέχεια την εμφάνιση του έκανε ο SatPlan όπου ανταγωνιζόταν τη συγκεκριμένη περίοδο τη τεχνολογία για επίλυση προβλημάτων προγραμματισμού δράσης. Αργότερα την εμφάνιση της έκανε η ευρετική συνάρτηση, μια μέθοδος η οποία φάνηκε να είναι καλύτερη στην επίλυση του προβλήματος προγραμματισμού δράσης σαν πρόβλημα ικανοποιησιμότητας προτασιακής λογικής. Στην συνέχεια όμως αποδείχτηκε πως το εργαλείο SatPlan ήταν πιο αποδοτικό.

3.3.1 Ιστορική αναδρομή στην υλοποίηση του SatPlan

Το αρχικό SatPlan σύστημα ήταν στην πραγματικότητα ένα σύνολο συμβάσεων για την κωδικοποίηση προβλημάτων γραμμικού περιορισμού της μορφής STRIPS, σε προτασιακό σχήμα δράσεων το οποίο προτάθηκε από τους Kautz και Selman το 1992.[1,6,7]

Η βασική ανάπτυξη της θεωρίας αυτής ήταν η εισαγωγή των παράλληλων κωδικοποιήσεων. Αυτό σήμαινε ότι πολλές πράξεις οι οποίες δεν παρεμβάλλονταν μεταξύ τους, μπορούσαν να συμβούν στο ίδιο χρονικό πλαίσιο. Η πρώτη εφαρμογή του SatPlan ήταν στο MEDIC σύστημα το 1997 και είχε σαν είσοδο το συμβολισμό STRIPS. Ακολούθησε μετά το BLACKBOX το οποίο υλοποιούσε επιπλέον mutex propagation που είναι μια μορφή τοπικής-συνέπειας. Οι επιδόσεις του BLACKBOX ήταν εξαίρετες.

Μετέπειτα, εμφανίστηκε το Satplan2004 σύστημα όπου ήταν το ταχύτερο και πιο ισχυρό σύστημα επίλυσης προβλημάτων προγραμματισμού δράσεων σε πέντε διαφορετικά πεδία ορισμού και δεύτερο καλύτερο σύστημα σε δύο πεδία ορισμού. Η επόμενη αναβαθμισμένη έκδοση ήταν η SatPlan-2006. Η έκδοση SatPlan-2006 παίρνει σαν είσοδο το υποσύνολο STRIPS της PDDL(Planning Domain Definition Language) γλώσσας. Σκοπός του είναι να βρίσκει λύσεις με βάση το μικρότερο παράλληλο μήκος. Έχει την λειτουργικότητα αυτή γιατί πρέπει να υπολογίζει το μικρότερο σύνολο χρονικών βημάτων μέσα από τις πολλές πράξεις που μπορεί να εμφανίζονται παράλληλα σε κάθε χρονικό διάστημα.

3.3.2 Λειτουργία του SatPlan

Ο SatPlan λειτουργεί αρχικά με την κατασκευή ενός GraphPlan μέχρι ένα μήκος k. Ο GraphPlan με βάση το πρόβλημα το οποίο του δίνεται κατασκευάζει μια δομή, το γράφημα δράσεων. Κάθε σχέδιο είναι ένα είδος ροής των μεταβλητών που παίρνουν τιμή true μέσα στο γράφημα. Στην συνέχεια ο SatPlan μεταφράζει τους περιορισμούς από το γράφο σε ένα σύνολο από προτάσεις(clauses). Σε κάθε συγκεκριμένο στιγμιότυπο μιας δράσης ή ενός γεγονότος, το σύνολό των προτάσεων αυτών αποτελεί μια δήλωση[6,7,14].

Ακολούθως, χρησιμοποιεί ένα εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής. Προσπαθεί να βρει μια ανάθεση τιμών που θα επιλύει το πρόβλημα. Εάν το αποτέλεσμα τώρα δεν είναι ικανοποιήσιμο αυξάνεται η σταθερά k κατά 1 και επαναλαμβάνεται η διαδικασία αλλιώς μεταφράζεται η λύση του προβλήματος ικανοποιησιμότητας προτασιακής λογικής στη λύση του αρχικού προβλήματος. Τέλος επεξεργάζεται την λύση αφαιρώντας αχρείαστες πράξεις, απαραίτητο βήμα για το λόγο ότι η μετάφραση του προβλήματος ικανοποιησιμότητας προτασιακής λογικής από το γράφημα δράσεων δεν εγγυάται ότι κάθε πράξη η οποία έχει την τιμή true είναι όντως απαραίτητη για την επίτευξη των στόχων του αρχικού προβλήματος.Ο αλγόριθμος παρουσιάζεται στο σχήμα 3.

```
function SATPLAN(problem, T_max) returns solution or failure
  inputs: problem, a planning problem
          T_max, an upper limit for plan length

  for T = 0 to T_max do
    cnf, mapping ← TRANSLATE-TO-SAT(problem, T)
    assignment ← SAT-SOLVER(cnf)
    if assignment is not null then
      return EXTRACT-SOLUTION(assignment, mapping)
  return failure
```

Σχήμα 3 «Αλγόριθμος SatPlan»

3.3.2.1 Προσέγγιση SAT MAX PLAN

Η προσέγγιση του SatPlan για επίλυση και βελτιστοποίηση των STRIPS προβλημάτων είναι μια από τις πιο επιτυχείς προσεγγίσεις. Η αποδοτικότητα της στηρίζεται στο συνδυασμό της απόδοσηw του state-of-the-art των προτασιακών Sat Solvers με την χρήση των περιορισμών που έχουν εισαχθεί από το πρόβλημα του γραφήματος δράσεων.[1]

Μια νέα έρευνα [1] βασίζεται στην μελέτη και έρευνα του σχεδιασμού ενός ικανοποιήσιμου πλαισίου όπου θεωρεί το πρόβλημα του προγραμματισμού

δράσεων σαν ένα πρόβλημα ικανοποίησης περιορισμών. Βασική ιδέα είναι η ανάθεση τιμών στις νέες μεταβλητές να παράγεται από μεταβλητές οι οποίες είναι ήδη γνωστές ή έχουν υιοθετηθεί από άλλες μεταβλητές. Περισσότερες τιμές για τις μεταβλητές εξάγονται από ισχυρότερες μορφές διάχυσης(propagation). Αυτό το γεγονός βοηθά στην αναζήτηση γιατί γίνεται ένα κλάδεμα στο χώρο αναζήτησης σε σύγκριση με ασθενέστερες μορφές. Έχει παρατηρηθεί ότι εάν το υπολογιστικό κόστος της διάχυσης είναι χαμηλό τότε η μείωση του χώρου αναζήτησης μεταφράζεται σε καλύτερους χρόνους εκτέλεσης.

Κάθε ένα εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής χρησιμοποιεί την καθιερωμένη μέθοδο της διάχυσης. Το μοντέλο του προβλήματος προγραμματισμού δράσης θεωρείται πιο ισχυρό εάν μπορεί να διαχέει περισσότερες τιμές για τις μεταβλητές. Επιπλέον μια κωδικοποίηση είναι πιο συμπαγής εάν επιτυγχάνει την ίδια διάχυση τιμών αλλά με ένα υποσύνολο από όρους.

Το εργαλείο BLACKBOX χρησιμοποιεί κωδικοποίηση η οποία είναι πιο ισχυρή από την κωδικοποίηση που χρησιμοποιεί ο SATPLAN06.Ένα νέο εργαλείο τώρα το SATMAXPLAN εκτελεί περισσότερη διάχυση από όλα τα άλλα μοντέλα. Αυτό το πετυχαίνει με σύνολα από όρους που δεν περιέχουν κανένα πλεονασμό. Ο SATMAXPLAN σε συνδυασμό με ένα καινούργιο εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής τον precosat μπορεί να επιλύσει περισσότερα προβλήματα και παρουσιάζει μια αξιοσημείωτη πρόοδο.

ΚΕΦΑΛΑΙΟ 4

4. Ψευδοδυαδικοί περιορισμοί και Προτασιακή Ικανοποιήσιμοτητα

4.1 Ψευδοδυαδικοί περιορισμοί πληθικότητας

4.1.1 Εισαγωγή

Θέτοντας ένα ευρύτερο ορισμό, οι ψευδοδυαδικές συναρτήσεις είναι συναρτήσεις, στις οποίες οι Boolean τιμές αντιστοιχούν σε πραγματικούς αριθμούς. Ο ορισμός τους πηγάζει από το γεγονός ότι αν και δεν είναι Boolean συναρτήσεις διατηρούν σε μεγάλο βαθμό ομοιότητα με αυτές. Η μελέτη τους άρχισε στα μέσα του 1960 στο πλαίσιο της μελέτης των τελεστών αλλά και του 0-1 προγραμματισμού.

Αρκετά προβλήματα σε διάφορους τομείς μπορούν να εκφραστούν ως προβλήματα βελτιστοποίησης της τιμής μιας ψευδοδυαδικής συνάρτησης, για αυτό το λόγο οι ψευδοδυαδικές συναρτήσεις αποτελούν ένα ενδιαφέρον θέμα προς μελέτη. Οι ψευδοδυαδικοί περιορισμοί είναι πιο εκφραστικοί και μπορούν να δείξουν με ένα πιο φυσικό τρόπο τους περιορισμούς σε σύγκριση με τα clauses. Επίσης ο φορμαλισμός που ακολουθούν δεν διαφέρει σε μεγάλο βαθμό από το φορμαλισμό που χρησιμοποιείται στα SAT προβλήματα και έτσι επωφελείται από τις εξελίξεις που σημειώνονται για επίλυση των SAT προβλημάτων. Ακόμη ένα πλεονέκτημα είναι ότι τα εργαλεία επίλυσης ψευδοδυαδικών προβλημάτων επωφελούνται από την μεγάλη γνώση στο τομέα του γραμμικού ακέραιου προγραμματισμού(Integer Linear Programming) και πιο συγκεκριμένα του 0-1 ακέραιου προγραμματισμού. Μέσα από δυνατούς συμπερασματικούς κανόνες τα προβλήματα μπορούν να λύνονται σε πολυωνυμικό χρόνο, όταν κωδικοποιούνται με ψευδοδυαδικούς περιορισμούς. Αν η κωδικοποίηση των προβλημάτων γίνεται με διάζευξη όρων η ανάλυση τους απαιτεί εκθετικό αριθμό βημάτων.[12]

Εν ολίγοις οι ψευδοδυαδικοί περιορισμοί είναι μια υποσχόμενη μελέτη σε σχέση με δύο θέματα. Το πρώτο αφορά την εκφραστικότητα του φορμαλισμού

για την αναπαράσταση ενός προβλήματος. Το δεύτερο αφορά την δυσκολία του να λυθεί το πρόβλημα με την μορφή του συγκεκριμένου φορμαλισμού.

4.2 Γνωσιολογικό υπόβαθρο.

4.2.1Γραμμικοί Pseudo Boolean περιορισμοί (Linear Pseudo-Boolean Constraints)

Ένας γραμμικός Pseudo Boolean περιορισμός (LPB Constraint) έχει την μορφή

$$\sum_{j}^{n} a_j l_j \otimes b$$

όπου το αj και b είναι ακέραιες μεταβλητές, lj είναι οι συντελεστές και το σύμβολο ⊗ αντιστοιχεί στους σχεσιακούς τελεστές(=, <, >, ≤, ≥). Το δεξιό μέρος του περιορισμού ονομάζεται βαθμός του περιορισμού. Ο περιορισμός θεωρείται ότι είναι ικανοποιήσιμος όταν το δεξί και αριστερό μέρος του περιορισμού πάρουν τιμές ακεραίων τέτοιες ώστε να ικανοποιούν τον σχεσιακό τελεστή.

Παράδειγμα:

$4x1+ 2x2 + 6(\neg x3) \geq 8$

Ο πιο πάνω περιορισμός μπορεί να ικανοποιηθεί εάν οι μεταβλητές πάρουν τις ακόλουθες τιμές:

α)x1 = 1 ,x2 = 0, x3 = 0
β)x1 = 1, x2 = 1, x3 = 0
γ)x1 = 0, x2 = 1, x3 = 0

4.2.2 Περιορισμοί πληθικότητας

Οι περιορισμοί πληθικότητας είναι περιορισμοί που εφαρμόζονται πάνω στον αριθμό των όρων που έχουν τιμή αληθείας True σε ένα δεδομένο σύνολο από όρους. Υπάρχουν τρεις διαφορετικοί τύποι περιορισμών πληθικότητας, οι ελάχιστοι (at least), οι μέγιστοι (at most) και οι ακριβείς (exactly). Οι ελάχιστοι (at least) περιορισμοί έχουν την ακόλουθη μορφή atleast(k,{x1,x2,...,xn}). Ο περιορισμός αυτός θεωρείται ικανοποιήσιμος εάν και μόνο εάν το λιγότερο k όροι από το σύνολο x1,x2,...,xn πάρουν τιμή αληθείας True. Ο περιορισμός σε ψευδοδυαδική μορφή είναι ο ακόλουθός $x1+,x2+...+xn \geq k$.

Οι μέγιστοι (at most) περιορισμοί έχουν την ακόλουθη μορφή atmost(k,{x1,x2,...,xn}). Ο περιορισμός αυτός θεωρείται ικανοποιήσιμος εάν και μόνο εάν ο μέγιστος αριθμός k όρων από το σύνολο x1,x2,...,xn πάρουν τιμή αληθείας True. Ο περιορισμός σε ψευδοδυαδική μορφή είναι ο ακόλουθός $x1+,x2+...+xn \leq k$.

Τέλος ο ακριβής περιορισμός έχει την ακόλουθη μορφή (k,{x1,x2,...,xn}). Ο περιορισμός αυτός θεωρείται ικανοποιήσιμος εάν και μόνο εάν, ακριβής αριθμός k όρων από το σύνολο x1,x2,...,xn πάρουν τιμή αληθείας True. Ένας ψευδοδυαδικός περιορισμός όπου όλοι οι συντελεστές του είναι ίσοι είναι ουσιαστικά ένας πληθικός περιορισμός. Με λίγα λόγια η μορφή του περιορισμού

$$\sum_{j=1}^{n} ax_j \geq b$$

είναι ο αντίστοιχος πληθικός περιορισμός atleast($\lceil b/a \rceil$,{x1,x2,...,xn}).

Οι περιορισμοί πληθικότητας είναι ουσιαστικά ένα ξεχωριστό είδος ψευδοδυαδικών περιορισμών. Ο πληθικός περιορισμός atleast(k,{x1,x2,...,xn}) μπορεί να μεταφραστεί σε ένα ισοδύναμο συνδυασμό από διαζεύξεις όρων.

Χωρίς όμως μια βοηθητική μεταβλητή το μέγεθός της μετάφρασης θα είναι εκθετικό. Στόχος της μετατροπής αυτής είναι να εκφράσει ότι δεν μπορούν περισσότεροι από n – k όροι να πάρουν την τιμή false. Αυτό εξ' υπακούει στο γεγονός ότι από την στιγμή που υπάρχουν επιλεγμένοι n – k + 1 όροι, έστω ένας από αυτούς πρέπει να είναι true. Με αυτό το τρόπο ο περιορισμός atleast(k,{x1,x2,...,xn}) είναι ισοδύναμος με το συνδυασμό όλων των πιθανών clauses που λαμβάνονται όταν διαλέγουν n – k + 1 όρους μέσα από το σύνολο {x1,x2,...,xn}.

4.3 Προβλήματα αποφάσεων και προβλήματα βελτιστοποίησης

Τα προβλήματα αποφάσεων είναι τα προβλήματα στα οποία αναζητούμε να δώσουμε μόνο μια απάντηση για κάθε ένα από αυτά. Με βάση μια φόρμουλα, που είναι ο συνδυασμός Pseudo Boolean περιορισμών, καλείται η φόρμουλα αυτή να αναγνωρίσει κατά πόσο υπάρχει μια ανάθεση τιμών τέτοια ώστε το πρόβλημα να θεωρείται ικανοποιήσιμο. Η μόνη απάντηση που δίδεται είναι εάν το πρόβλημα είναι ικανοποιήσιμο ή το αντίθετο. Τα προβλήματα αποφάσεων έχουν την ονομασία Pseudo Boolean Solving (PBS). Το να βρεθεί πότε ικανοποιούνται οι ψευδοδυαδικοί περιορισμοί είναι πολύ κοντά στην ιδέα του προβλήματος του SAT, που ασχολείται με την ικανοποιησιμότητα των διαζευκτικών προτάσεων.

Ακόμη ένα θέμα το οποίο είναι υπό μελέτη είναι το πρόβλημα της βελτιστοποίησης. Το πρόβλημα της βελτιστοποίησης προσπαθεί να βρει εάν το πρόβλημα είναι επιλύσιμο και αν ναι προσπαθεί να βρει την καλύτερη δυνατή λύση. Η απάντηση που αναμένεται να δώσει το εργαλείο επίλυσης είναι ή ότι το πρόβλημα δεν είναι επιλύσιμο ή ότι είναι επιλύσιμο και δεν μπορεί να βρεθεί καλύτερη λύση από αυτή την οποία ήδη βρέθηκε. Οι καλύτερες λύσεις ονομάζονται βέλτιστες (optimum). Ένα πρόβλημα μπορεί να έχει αρκετές βέλτιστες λύσεις. Παρουσιάζονται πολλές λύσεις λόγω του ότι το εργαλείο

επίλυσης μερικές φορές δεν έχει το χρόνο να βρει την βέλτιστη λύση με βάση τούς πόρους τους οποίους διαθέτει, και δίνει την καλύτερη λύση που βρήκε μέχρι την δεδομένη στιγμή. Αν και η λύση δεν είναι η βέλτιστη αποτελεί όριο της βέλτιστης λύσης.

Λόγω του ότι υπάρχουν αρκετές λύσεις στο πρόβλημα της βελτιστοποίησης πρέπει με κάποιο τρόπο να εντοπίζεται ποία λύση είναι καλύτερη από κάποια άλλη. Αυτό μπορεί να επιτευχθεί με μια συνάρτηση την ονομαζόμενη αντικειμενική συνάρτηση(objective function) η οποία αντιστοιχεί κάθε λύση με ένα ακέραιο. Η συνάρτηση αυτή μπορεί να ελαχιστοποιήσει την τιμή της. Σε αυτή την περίπτωση ονομάζεται συνάρτηση κόστους (cost function) και η λύση S είναι προτιμότερη από την S' όταν ικανοποιεί το ακόλουθο ObjectiveFunction(S) <ObjectiveFunction(S'). Επίσης η συνάρτηση αυτή μπορεί να μεγιστοποιηθεί. Σε αυτή την περίπτωση ονομάζεται συνάρτηση ωφέλειας (utility function)και η λύση S είναι προτιμότερη από την S' όταν ικανοποιεί το ακόλουθο ObjectiveFunction(S)>ObjectiveFunction(S'). Όλα τα προβλήματα μπορούν να οριστούν με την ελαχιστοποίηση της βέλτιστης συνάρτησης εφόσον η μεγιστοποίηση της βέλτιστης συνάρτησης είναι ισοδύναμη με την άρνηση της ελάχιστης. Με αυτό το τρόπο καταλήγουμε στο γενικό ορισμό της βελτιστοποίησης ενός Pseudo Boolean προβλήματος:

$$\text{minimize Cost}\left(x_{,1}, x_2 ..., x_n\right)$$

$$\text{subject to } \wedge_i \sum_j a_{i,j}\ x_j \geq b_i$$

Η συνάρτηση κόστους μπορεί να οριστεί και σαν γραμμική συνάρτηση αλλά και σαν μη γραμμική.

4.3.1 Εκφραστική δύναμη της πληθικότητας και των ψευδοδυαδικών περιορισμών.

Όπως έχει ήδη αναφερθεί οι ψευδοδυαδικοί περιορισμοί είναι πιο εκφραστικοί από ότι οι διαζευκτικές προτάσεις. Ένας ψευδοδυαδικός

περιορισμός μπορεί να αντικαταστήσει ένα εκθετικό αριθμό από διαζευκτικές προτάσεις. Η μετάφρασή ενός ψευδοδυαδικού περιορισμού σε ένα ισοδύναμο σύνολο από διαζευκτικές προτάσεις μπορεί στη χειρότερή περίπτωση να δώσει εκθετική πολυπλοκότητα. Ένα παράδειγμα που δημιουργεί εκθετική πολυπλοκότητα είναι η μετατροπή του περιορισμού atleast(k, $\{x_{,1}, x_2 ..., x_n\}$). Ο περιορισμός αυτός για να μεταφραστεί χρειάζεται συνδυασμό $\binom{n}{n-k+1}$ διαζευκτικών όρων Κωδικοποιώντας το περιορισμό αυτό σε διαζευκτικές προτάσεις, απαιτείται η εισαγωγή καινούργιων μεταβλητών, με αποτέλεσμα να καταλήγει σε μια κωδικοποίηση πολυωνυμικού χρόνου.

Στους ψευδοδυαδικούς περιορισμούς οι συντελεστές οι οποίοι χρησιμοποιούνται είναι αυθαίρετου μεγέθους ακόμη και σε συγκεκριμένα προβλήματα. Έτσι όταν ένα πρόβλημα αναλυθεί για την πολυπλοκότητά του, εισάγει και τον αριθμό των συντελεστών του. Αυτό γίνεται γιατί οι συντελεστές δεν μπορούν να έχουν ένα σταθερό μέγεθος. Για αυτό το λόγο πρέπει το εργαλείο επίλυσης να μην χρησιμοποιεί αριθμητική ακρίβεια γεγονός που κοστίζει σε αποδοτικότητα. Αν το εργαλείο επίλυσης χρησιμοποιεί αριθμητική ακρίβεια μπορεί να έχει υπερχείλιση ακεραίων σε μερικές περιπτώσεις. Η υπερχείλιση μπορεί να συμβεί στο αρχικό στάδιο, τη μεταγλώττιση ή κατά την εκτέλεση όταν εισάγονται καινούργιοι περιορισμοί από περιορισμούς που ήδη υπάρχουν. Αν η υπερχείλιση παρουσιαστεί κατά την διάρκεια της μεταγλώττισης είναι πιο εύκολο να εντοπιστεί και να διακοπεί η εκτέλεση του αλγορίθμου. Αν τώρα παρουσιαστεί κατά την εισαγωγή καινούργιων περιορισμών δεν μπορεί εύκολα να επιλυθεί για το λόγο ότι έχει αντίκτυπο στην ορθότητα εκτέλεσης του αλγορίθμου.

Η πολυπλοκότητα των ψευδοδυαδικών περιορισμών είναι ένα NP - complete πρόβλημα. Ο έλεγχος αν μια ανάθεση τιμών ικανοποιεί το ψευδοδυαδικό πρόβλημα μπορεί να γίνει σε πολυωνυμικό χρόνο αναλόγως βέβαια του μεγέθους του προβλήματος και κατ' επέκταση και το μέγεθος των

συντελεστών. Έτσι η ικανοποίηση ψευδοδυαδικών περιορισμών αποτελεί πρόβλημα NP. Επίσης είναι ένα NP- complete πρόβλημα εφόσον περιέχει το της επίλυσης προβλημάτων ικανοποίησης προτασιακής λογικής που είναι και αυτό με τη σειρά του NP – complete.

4.4 Προτασιακή Ικανοποιησιμότητα (Propositional Satisfiability)

Τα τελευταία χρόνια η χρήση των ειδικών εργαλείων(SAT solvers)που επιλύουν πρακτικά προβλήματα έχουν εξελιχθεί σε μεγάλη κλίμακα και έχουν εφαρμοστεί σε λογισμικά μοντέλα ελέγχου.

Θέμα εντατικής έρευνας εδώ και δεκαετίες αποτελεί το πρόβλημα της προτασιακής ικανοποιησιμότητας. Η έρευνα ασχολείται με ένα δοθέντα σύνολο από δυαδικές μεταβλητές και ένα σύνολο από περιορισμούς που είναι διατυπωμένοι σε κανονική διαζευκτική μορφή (Conjunctive Normal Form). Στόχος είναι η εύρεση μιας ανάθεσης τιμών που ικανοποιεί όλους τους περιορισμούς ή την απόδειξη αδυναμίας εύρεσης μιας τέτοιας ανάθεσης. Η μορφή των περιορισμών όμως σε κανονική διαζευκτική μορφή αποτέλεσε εμπόδιο για την αναπαράσταση προβλημάτων με περιορισμούς «counting constraints». Αυτό συνέβηκε επειδή τέτοιοι περιορισμοί δεν μπορούν να αναπαρασταθούν ικανοποιητικά. Έτσι οι ερευνητές επέκτειναν τα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής με σκοπό να μπορούν να επιλύουν προβλήματα με μορφή ψευδοδυαδικών περιορισμών που έχουν την ειδικότητα να αναπαριστούν εύκολα «counting constraints». Οι ψευδοδυαδικοί περιορισμοί είναι πιο συγκεκριμένοι και έχουν την δυνατότητα να αντικαταστήσουν ένα εκθετικό αριθμό από κανονικά διαζευκτικούς περιορισμούς. Τα προβλήματα αυτά στο παρελθόν αντιμετωπίζονταν σαν περιπτώσεις ILP(integer linear programming).

Αναλυτικά μια κανονική διαζευκτική πρόταση περιέχει δυαδικές μεταβλητές $x1,..,xn$. Οι μεταβλητές περιέχουν την σύζευξη (AND) m clauses $\omega 1,\ldots,\omega m$ όπου το καθένα αποτελείται από τη διάζευξη (OR) k όρων (literals). Ο όρος (literal) είναι η εμφάνιση ή το συμπλήρωμα μιας δυαδικής μεταβλητής. Η τιμή που αποδίδεται σε μια μεταβλητή είναι 0 ή 1. Αν η τιμή που αποδίδεται στην μεταβλητή δεν είναι 0 ή 1 δεν γίνεται ανάθεση. Ο όρος παίρνει τιμές 0 ή 1 αν αποτιμάται σε ψευδές ή αληθές αντίστοιχά κάτω από την συσχετιζόμενη με εκείνο μεταβλητή. Είναι ελεύθερο αν δεν έχει γίνει κάποια ανάθεση στην συσχετιζόμενη μεταβλητή. Ένα clause τώρα αποτιμάται σαν αληθή και είναι ικανοποιήσιμο εάν έστω και ένας όρος του πάρει την τιμή 1, δηλαδή είναι αληθές. Αν τώρα όλα τα στοιχεία που αποτελούν το clause είναι ψευδή τότε δεν είναι ικανοποιήσιμο. Σε περίπτωση που οι όροι είναι όλοι ψευδή εκτός από ένα τότε η διαζευκτική πρόταση ονομάζεται unit και αν ισχύει το αντίθετο τότε ονομάζεται άλυτο. Μια πρόταση τώρα είναι ικανοποιήσιμη εάν όλες οι διαζευκτικές προτάσεις της είναι ικανοποιήσιμες και μη ικανοποιήσιμη όταν έστω και μια διαζευκτική πρόταση της δεν ικανοποιείται.[3,12]

Οι περισσότεροι και πιο πρόσφατοι SAT Solvers στηρίζονται στον αλγόριθμό Davis – Putnam –Logemann – Loveland (DPLL). Στην αρχή της εκτέλεσης του αλγορίθμου οι μεταβλητές δεν έχουν κάποια τιμή έτσι προσπαθεί να βρει μια ανάθεση τιμών. Μετά την κάθε ανάθεση σε μια μεταβλητή ελέγχονται οι συνέπειες στις υπόλοιπες μεταβλητές. Ο έλεγχος αυτός γίνεται εφικτός υποχρεώνοντας την ανάθεση της μεταβλητής να ισχύει. Η μεταβλητή εδώ αναπαριστά ένα όρο που δεν έχει τιμή σε ένα άλυτο clause δηλαδή ένα clause που όλοι οι όροι του έχουν την τιμή 0. Ο προαναφερθέντας έλεγχος είναι γνωστός και ως κανόνας unit clause και η επαναλαμβανόμενη εφαρμογή του είναι γνωστή ως Boolean constraint propagation (BCP).

Αν ο αλγόριθμος δεν εντοπίσει κάποια σύγκρουση συνεχίζει με την δημιουργία μιας νέας ανάθεσης τιμών σε μια άλλη μεταβλητή που δεν είχε μέχρι

τώρα τιμή. Εάν όμως παρουσιαστεί σύγκρουση τότε ο αλγόριθμος επιστρέφει προς τα πίσω και αφαιρεί από την μεταβλητή την δοθείσα τιμή και συνεχίζει με την εύρεση καινούργιας τιμής. Η γενική μορφή του ψευδοδυαδικού αλγορίθμου παρουσιάζεται στο σχήμα 4.

```
1: function GENERIC_PB_SOLVER(P)
2:     while true do
3:         if Solution_Found(P) then
4:             return SATISFIABLE
5:         else
6:             Decide(P)
7:         end if
8:         while Deduce(P)=CONFLICT do
9:             if Diagnose(P)=CONFLICT then
10:                return UNSATISFIABLE
11:            end if
12:        end while
13:    end while
14: end function
```

Σχήμα 4 «Γενικός Αλγόριθμος PB Solver»

Ο κανόνας του Unit Clause εφαρμόζεται στα προτασιακά clauses. Στους αλγορίθμους επίλυσης προβλημάτων προτασιακής λογικής δηλώνεται πως εάν έχουμε ένα Unit Clause με ένα όρο χωρίς τιμή και όλους τους άλλους όρους με την τιμή 0, ο όρος που δεν έχει τιμή θα πρέπει να πάρει την τιμή 1. Το εργαλείο επίλυσης ψευδοδυαδικών προβλημάτων μπορεί να υπολογίσει την χαλαρότητα του κάθε περιορισμού και να εντοπίσει unit clauses. Ο εντοπισμός των unit clauses είναι αναγκαίο να επιτυγχάνεται σε γρήγορο βαθμό έτσι ώστε αν δεν πραγματοποιείται να γίνεται μια οπισθοδρόμηση στο δένδρο αναζήτησης. Για αυτό το λόγο η χρήση αποδοτικών δομών είναι αναγκαία για το χειρισμό των περιορισμών του προβλήματος. Ειδικές δομές έχουν υλοποιηθεί στα εργαλεία επίλυσης προβλημάτων SAT για να διαχειρίζονται ψευδοδυαδικούς περιορισμούς όπου και αυτοί με την σειρά τους διαχειρίζονται τα προτασιακά clauses.

4.4.1 Διάγνωση Συγκρούσεων

Ο αλγόριθμος Boolean constraint propagation (BCP) εμπλουτίστηκε με τη διάγνωση συγκρούσεων (Conflict Diagnosis) δημιουργώντας ένα από τα κύρια πλεονεκτήματα των CNF- SAT Solvers.[3,8,12]

Η γενική ιδέα της ανάλυσης συγκρούσεων είναι η ανάλυση των αλυσιδωτών επιπτώσεων που είναι δυνατόν να υπάρξουν μετά την αρχικοποίηση κάποιας μεταβλητής. Αυτό έχει σαν αποτέλεσμα ένας ή περισσότεροι όροι να είναι μη ικανοποιήσιμοι. Με την ανάλυση αυτή προσδιορίζεται ένα υποσύνολό από μεταβλητές που προκαλούν σύγκρουση με την ανάθεση τιμών που έχουν λάβει. Οι αναθέσεις των τιμών που δημιουργούν συγκρούσεις μετατρέπονται σε ένα όρο με την ονομασία conflict-induced. Ο όρος αυτός προστίθεται στην βάση δεδομένων των όρων και εμποδίζει την επανάληψη της ίδιας σύγκρουσης γεγονός που μπορεί να θεωρηθεί και ως μάθηση.

Με αυτό τον τρόπο έχουμε το πλεονέκτημα να μη κάνουμε χρονολογική «οπισθοχώρηση».Δηλαδή δεν πάμε στο προηγούμενο βήμα για έλεγχο των αναθέσεων αλλά πηγαίνουμε πίσω στο συγκεκριμένο επίπεδο όπου η μεταβλητή παρουσίασε σύγκρουση. Έτσι αποφεύγουμε την άσκοπη αναζήτηση στο χώρο.

Αρκετές φορές ένας SAT solver μπορεί να κολλήσει κατά την προσπάθεια του να βρει λύση λόγω των αποφάσεων που έχουν ήδη γίνει. Μελέτες όμως έχουν ανακαλύψει ότι με μια επανεκκίνηση μπορεί ο SAT Solver να ξεκολλήσει από τέτοιες περιοχές του χώρου αναζήτησης. Αυτό είναι εφικτό με την περιοδική επαναφορά όλων των αποφάσεων και την τυχαία επιλογή μιας καινούργιας ανάθεσης. Επίσης διασφαλίζεται ότι κάθε φορά η αναζήτηση λαμβάνει μέρος σε ένα άλλο υπόδεντρο.

4.5 Ψευδοδυαδικοί περιορισμοί

Σε ένα πρόβλημα προτασιακής λογικής μπορούν να εισαχθούν και ψευδοδυαδικοί περιορισμοί. Οι ψευδοδυαδικοί περιορισμοί έχουν την μορφή που φαίνεται στην σχήμα 5 όπου τα a_i,b ανήκουν στο σύνολο Z και x_i είναι τα στοιχεία των δυαδικών μεταβλητών. Γενικά όμως κάθε όρος που είναι γραμμένος σε κανονική διαζευκτική μορφή μπορεί να γραφεί και ως ψευδοδυαδικός περιορισμός. Η κανονικοποιημένη μορφή ενός ψευδοδυαδικού περιορισμού διαθέτει μονό θετικούς συντελεστές και διευκολύνει την αποτελεσματικότητα των αλγορίθμων επίλυσης προβλημάτων προτασιακής λογικής. Η κανονικοποιημένη μορφή των ψευδοδυαδικών περιορισμών μπορεί να δημιουργηθεί με την χρήση των πιο κάτω σχέσεων:[3,4,18]

- $\neg x = (1 - x)$
- $(Ax = b) \Leftrightarrow (Ax \leq b) \wedge (Ax \geq b)$
- $(Ax \geq b) \Leftrightarrow (\neg Ax \leq b)$

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \leq b$$

Σχήμα 5 «Ψευδοδυαδικοί περιορισμοί»

4.5.1 Το πρόβλημα της βελτιστοποίησης

Υπάρχουν δυο διαφορετικές προσεγγίσεις για την επίλυση βέλτιστων προβλημάτων με ένα εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής, η γραμμική αναζήτηση και η δυαδική αναζήτηση.

4.5.1.1 Αλγόριθμός γραμμικής αναζήτησης

Στην γραμμική αναζήτηση η συνάρτηση βελτιστοποίησης $o(x_1, x_2, ...x_n)$ ελαχιστοποιείται και μετατρέπεται σε ένα περιορισμό της μορφής $o(x_1, x_2, ...x_n) \leq$ B όπου το B είναι μια ακέραια τιμή και ενημερώνεται σε κάθε επανάληψη του αλγορίθμου. Αρχικά η μεταβλητή B παίρνει την μέγιστη τιμή της συνάρτησης βελτιστοποίησης γεγονός που δεν επηρεάζει και την λύση του προβλήματος.

1: $B \leftarrow$ maximum value of the objective function
2: enforce constraint $o(x_1, \ldots, x_n) \leq B$
3: **while** formula is satisfiable **do**
4: $\quad B \leftarrow o(solution) - 1$
5: $\quad$ enforce constraint $o(x_1, \ldots, x_n) \leq B$
6: **end while**

Σχήμα 7 «Αλγόριθμός γραμμικής αναζήτησης»

Σε κάθε βήμα βρίσκεται μια λύση S και η μεταβλητή B παίρνει μια νέα τιμή που υπολογίζεται o(S) – 1 έτσι ώστε να προσπαθήσει να βρει μια καλύτερη λύση από την S. Η διαδικασία αυτή τελειώνει όταν η φόρμουλα δεν ικανοποιείται. Σε αυτή την περίπτωση εάν έχει βρεθεί προηγούμενη λύση, η λύση αυτή αποτελεί και την βέλτιστη. Στο σχήμα 7 παρουσιάζεται ο αλγόριθμος γραμμικής αναζήτησης. Ο αλγόριθμος μπορεί να κινείται αργά προς την εύρεση της βέλτιστης λύσης αλλά μπορεί να ανακαλύπτει σε πιο γρήγορα βαθμό τις πρώτες λύσεις σε αντίθεση με τον αλγόριθμό δυαδικής αναζήτησης.

4.5.1.2 Αλγόριθμός δυαδικής αναζήτησης

Στη δυαδική αναζήτηση η συνάρτηση βελτιστοποίησης $o(x_1, x_2, ...x_n)$ συσχετίζεται με δύο περιορισμούς τους L $\leq o(x_1, x_2, ...x_n)$ και $o(x_1, x_2, ...x_n) \leq$ U

όπου οι μεταβλητές L και U είναι ακέραιοι αριθμοί που έχουν την ελάχιστη και την μέγιστη τιμή της συνάρτησης βελτιστοποίησης αντίστοιχά.

```
1: L ← minimum value of the objective function
2: U ← maximum value of the objective function
3: while L ≤ U do
4:     M ← (L + U)/2
5:     enforce constraints L ≤ o(x_1, ..., x_n) ≤ M
6:     if formula is satisfiable then
7:         U ← o(solution) − 1
8:     else
9:         L ← M + 1
10:    end if
11: end while
```

Σχήμα 8 «Αλγόριθμός δυαδικής αναζήτησης»

Σε κάθε βήμα υπολογίζεται η μέση τιμή του L και U η M. Μόλις βρεθεί η μεταβλητή M ο περιορισμός παίρνει την μορφή L $\leq o(x_1, x_2, ...x_n) \leq$ M. Σε κάθε βήμα που βρίσκεται μια λύση η μεταβλητή U παίρνει μια νέα τιμή που υπολογίζεται o(S) – 1.Αν τώρα δεν βρεθεί λύση αλλάζει η μεταβλητή L και παίρνει την τιμή M+1. Ο αλγόριθμός επαναλαμβάνεται έως ότου η τιμή L γίνει μεγαλύτερη από την τιμή U. Η περιγραφή του παρουσιάζεται στο σχήμα 8.

Οι ψευδοδυαδικοί περιορισμοί αποτελούν ένα ενδιαφέρον θέμα προς μελέτη. Είναι πιο εκφραστικοί και μπορούν να εκφράσουν με ένα πιο φυσικό τρόπο περιορισμούς σε σύγκριση με τις προτάσεις διαζευκτικών όρων. Επίσης ο φορμαλισμός που ακολουθούν δεν διαφέρει σε μεγάλο βαθμό από το φορμαλισμό που χρησιμοποιείται στα προβλήματα ικανοποιησιμότητας προτασιακής λογικής και έτσι ωφελείται από τις εξελίξεις που αφορούν το τομέα αυτό. Έχει αποδειχθεί ότι ο φορμαλισμός των ψευδοδυαδικών περιορισμών είναι πιο μικρός εκθετικά από ότι ο φορμαλισμός των διαζευκτικών προτάσεων. Επίσης το ίδιο ισχύει και για τις κωδικοποιήσεις των εργαλείων επίλυσης ψευδοδυαδικών περιορισμών σε σχέση με τις κωδικοποιήσεις που

χρησιμοποιούνται στα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής.

ΚΕΦΑΛΑΙΟ 5

5. Προβλήματα Προγραμματισμού Δράσης – SATMAXPLAN

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε διάφορα προβλήματα προγραμματισμού δράσεων στην γλώσσα STRIPS καθώς και τους χρόνους επίλυσης τους μέσα από το εργαλείο SATPLANMAX. Το εργαλείο SATPLANMAX συνδυάζει ένα καινούργιο εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής το Precosat. Το εργαλείο αυτό έχει την δυνατότητα να επιλύει περισσότερα προβλήματα και επιτυγχάνει να είναι κυρίαρχο στα προβλήματα προγραμματισμού δράσης μεταφρασμένα σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής. Το εργαλείο επίλυσης προβλημάτων SATPLANMAX χρησιμοποιεί μια νέα κωδικοποίηση την ονομαζόμενη GraphPlan- direct που αποτελεί μια άμεση μετάφραση της δομής του γραφήματος δράσεων σε προτασιακή λογική.[1]

5.1.1 Γνωσιολογικό υπόβαθρό

Ένα πρόβλημα γραμμένο σε γλώσσα STRIPS είναι μια τριάδα από ένα σύνολο με γεγονότα τα οποία απεικονίζουν την αρχική κατάσταση(I), τους στόχους που πρέπει να ικανοποιηθούν(G) και ένα σύνολο από δράσεις (A). Η τριάδα αυτή αναπαριστάται ως P =< I,G,A >. Κάθε δράση A έχει ένα σύνολο από προϋποθέσεις που δηλώνονται σαν pre(A). Την πρόσθεση ή την αφαίρεση του αποτελέσματος από την εκτέλεση μιας πράξης που δηλώνονται σαν add(A) και del(A) αντίστοιχα. Η δομή του γραφήματος δράσεων μπορεί να κατασκευαστεί από την περιγραφή ενός προβλήματος προγραμματισμού δράσεων με κεντρική ιδέα την παρεμβολή των δράσεων. Ένας ορισμός για αυτό είναι ο ακόλουθος: δύο δράσεις A1 και A2 παρεμβάλλονται όποτε κάποιο από τα σύνολα del(A1) ∩ add (A2) και del(A1) ∩ pre(A2) δεν είναι άδεια .

Όλα τα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής τόσο το BLACKBOX όσο και το SATPLAN χρησιμοποιούν πληροφορίες οι οποίες εξάγονται από το γράφημα δράσεων το

οποίο δημιουργείται με βάση το πρόβλημα. Ένα μέρος των πληροφοριών που εξάγονται είναι αμοιβαία αποκλειόμενα ζεύγη (mutually exclusive pairs, mutexes). Τα ζεύγη τα οποία είναι αμοιβαία αποκλειόμενα έχουν τους ακόλουθους ορισμούς. Δύο δράσεις A_1, A_2 θεωρούνται αμοιβαία αποκλειόμενες εάν στο επίπεδο l παρεμβάλλονται ή υπάρχει ένας ζεύγος από γεγονότα $f_1 \in \text{pre}(A_1)$, $f_2 \in \text{pre}(A_2)$ τέτοια ώστε f_1, f_2 να είναι αμοιβαία αποκλειόμενα στο επίπεδο l. Δύο γεγονότα f_1, f_2 θεωρούνται αμοιβαία αποκλειόμενα εάν στο επίπεδο l για κάθε ζεύγος δράσεων A_1, A_2 τέτοιο ώστε $f_1 \in \text{add}(A_1)$, $f_2 \in \text{add}(A_2)$και οι δράσεις A_1, A_2 να είναι αμοιβαία αποκλειόμενα ζεύγη 1-1.

Στο γράφημα δράσεων κάθε ένα επίπεδο αναφέρεται σε ένα διαφορετικό χρονικό σημείο και η αδράνεια υποδηλώνεται με noop δράσεις. Σε ένα μοντέλο προβλήματος ικανοποιησιμότητας προτασιακής λογικής οι χρονικές μεταβλητές αναπαριστούν τις δράσεις ή τα γεγονότα του προβλήματος. Συγκεκριμένα μια μεταβλητή της μορφής A(T) ,με το A να είναι η δράση , αντιστοιχεί στο αν η δράση A θα εκτελεστεί ή όχι την χρονική στιγμή T. Το ίδιο ισχύει και για μεταβλητές της μορφής f(T) όπου το f υποδηλώνει γεγονός. Σε όλα τα συστήματα που χρησιμοποιούν το γράφημα δράσεων μια μεταβλητή δράση ή γεγονότος εμφανίζεται στο θεώρημα μόνο εάν η μεταβλητή αυτή εμφανίζεται στο γράφημα δράσεων.

Μια νέα κωδικοποίηση είναι η Graphplan- direct. Η κωδικοποίηση αυτή είναι μια απευθείας μετάφραση από την δομή του γραφήματος δράσεων σε προτασιακή λογική. Πιο κάτω περιγράφονται οι προτάσεις που απαρτίζουν τη κωδικοποίηση Graphplan- direct.

1. Unit προτάσεις για την αρχική και τελική κατάσταση
2. $A(T) \rightarrow f(T)$, για κάθε δράση A και γεγονός f τέτοιο ώστε $F \in \text{pre}(A)$.

3. A(T) → f (T + 1), για κάθε δράση A και γεγονός f τέτοιο ώστε f ∈ add(A).
4. A(T) → ¬f (T + 1),για κάθε δράση A και γεγονός f τέτοιο ώστε f ∈ del(A).
5. f (T) → A1 (T − 1) ∨ . . . ∨ Am (T − 1), για κάθε γεγονός και όλες τις δράσεις Ai,1 ≤ i ≤ m (συμπεριλαμβανομένων και των noops) τέτοια ώστε f ∈ add(Ai).
6. ¬f (T) → A1 (T − 1) ∨. . . ∨ Am (T − 1) ∨ ¬f (T − 1), για κάθε γεγονός και όλες τις δράσεις A,1 ≤ i ≤ m τέτοιο ώστε f ∈ del(Ai).

7.1 ¬A1 (T) ∨ ¬A2 (T), για κάθε ζευγάρι από δράσεις A1 , A2 τέτοιο ώστε το σύνολο del(A1) ∩ pre(A2) να μην είναι άδειο.

7.2 ¬A1 (T) ∨ ¬A2 (T), για κάθε ζευγάρι από δράσεις A1 , A2 τέτοιο ώστε το σύνολο del(A1) ∩ add(A2) να μην είναι άδειο.

7.3 ¬A1 (T) ∨ ¬A2 (T), εάν υπάρχει ένα ζευγάρι από γεγονότα f1 ∈ pre(A1) f2 ∈ pre(A2) τέτοιο ώστε f1 , f2 αλληλοαναιρούνται στην χρονική στιγμή T.

8. ¬f1 (T) ∨ ¬f2 (T), για κάθε ζευγάρι από γεγονότα f1 , f2 τα οποία είναι mutex στη χρονική στιγμή T .

Διάφορα υψηλού επίπεδου ανάπτυξης εργαλεία επίλυσης SAT προβλημάτων όπως το Siege το Minisat και το Precosat χρησιμοποιούν Unit resolution για την διάχυση των περιορισμών (constraint propagation). Προσπαθούν να επιλύσουν όλους τους unit όρους, όροι δηλαδή που τα στοιχεία είναι όλα ψευδή εκτός από ένα, με βάση όλες τις προτάσεις που βρίσκονται στη θεωρία. Αυτή η διαδικασία επαναλαμβάνεται έως ότου να μην μπορεί να παραχθεί κάποιο άλλος unit όρος ή εάν αποδειχτεί ότι το πρόβλημα δεν είναι επιλύσιμο.

5.2 Ο ρόλος και η δύναμη των κωδικοποιήσεων

Τα εργαλεία BLACKBOX και SATMAXPLAN υποστηρίζουν διαφορετικές κωδικοποιήσεις. Το πρώτο σύστημα το όποιο ασχολήθηκε με πληροφορίες εξαγόμενες από το γράφημα δράσεων και την προτασιακή κωδικοποίηση ενός προβλήματος προγραμματισμού δράσης ήταν το εργαλείο BLACKBOX. Το εργαλείο αυτό υποστηρίζει διαφορετικές κωδικοποιήσεις, μια από αυτές είναι και η κωδικοποίηση BB-31. Το σύνολο των προτάσεων της κωδικοποίησης BB-31 είναι υποσύνολο των προτάσεων του Graphplan-direct. Συγκεκριμένα αποτελείται από τις προτάσεις 1 ,2 ,3,4,5,7.1 και 8.

Για καλύτερη κατανόηση θα εισαχθεί ο όρος UP(T). Ο όρος UP (T) δηλώνει την κλειστότητα της θεωρίας T κάτω από τη διάχυση των προτάσεων(Unit Propagation). Η έννοια τώρα της UP-περιτότητας που θα είναι χρήσιμη στη καλύτερη κατανόηση των κωδικοποιήσεων, ορίζεται ως εξής:
Η δυαδική πρόταση $l1 \vee l2$ είναι UP- περιττή με βάση μια θεωρία T εάν $l1 \vee l2 \in$ T ή εάν $lj \in (UP (T \cup \{li\})$ για κάθε $j \neq i$ και $j,i \in \{1,2\}$.

Κάθε μια κωδικοποίηση υποστηρίζει και διαφορετικό σύνολο από προτάσεις όπως φαίνεται παρακάτω:
Η BB-31 Υποστηρίζει τις προτάσεις 1 ,2 ,3,4,5,7.1,8.
Η BB-7 Υποστηρίζει τις προτάσεις 1, 2, 5, 7.1, 7.2, 7.3
Η BB-32 Υποστηρίζει τις προτάσεις 1, 2, 3, 4, 5, 7.1, 7.2, 7.3, 8
Η SATPLAN06-4 Υποστηρίζει τις προτάσεις 1, 2, 5, 7.1, 7.2, 8
Η SATPLAN06-3 Υποστηρίζει τις προτάσεις 1, 2, 5, 7.1, 7.2, 7.3, 8

Στις προτάσεις που ακολουθούν μας επιβεβαιώνεται ότι σε κωδικοποιήσεις χρησιμοποιούνται αμοιβαία αποκλειόμενες προτάσεις οι οποίες είναι UP- περιττές.

Πρόταση 1:Τα σύνολα των προτάσεων 7.3 είναι UP-περιττά με βάση οποιαδήποτε κωδικοποίηση, που περιέχει τα σύνολα των προτάσεων 2 και 8.

Από εδώ εξάγεται το συμπέρασμα ότι η κωδικοποίηση SATPLAN06-4(P) είναι πιο ισχυρή από τη κωδικοποίηση SATPLAN06-3(P) για κάθε πρόβλημα προγραμματισμού δράσης P.

Πρόταση 2:Τα σύνολα των προτάσεων 7.2 είναι UP- περιττά με βάση οποιαδήποτε κωδικοποίηση, που περιέχει τα σύνολα των προτάσεων 3 και 4.

Από εδώ και με βάση την πρόταση 1 εξάγεται το συμπέρασμα ότι οι προτάσεις 7.2 και 7.3 μπορούν να παραληφθούν. Έτσι η κωδικοποίηση BB-31 είναι πιο ισχυρή από το από την κωδικοποίηση BB-32 για κάθε πρόβλημα προγραμματισμού δράσης P.

Αφαιρώντας τις UP- περιττές προτάσεις 7.2 και 7.3 καταλήγουμε στην κωδικοποίηση του SATMAXPLAN που υποστηρίζει από το υποσύνολο των προτάσεων του Graphplan-direct τις προτάσεις 1,2,3,4,5,6,7.1 και 8.
Το σύνολο των προτάσεων 8 τώρα δεν αποτελεί πλεονασμό για οποιαδήποτε κωδικοποίηση καθιστώντας έτσι την κωδικοποίηση BB-31 την πιο ισχυρή μέχρι στιγμής. Στην κωδικοποίηση όμως τώρα, του SATMAXPLAN υπάρχει το σύνολο των προτάσεων 6 γεγονός που τη καθιστά πιο ισχυρή.

Είναι γεγονός ότι η κωδικοποίηση SATMAXPLAN χρησιμοποιεί μόνο ένα σύνολο των αμοιβαίων αποκλειόμενων δράσεων, το 7.1. Με αυτό το τρόπο είναι πιθανό μια πρόταση να συμπεριλαμβάνεται σε περισσότερα από ένα σύνολα αμοιβαίων αποκλειόμενων προτάσεων και κάθε φόρα για διαφορετικό λόγο. Ως εκ τούτου, ένα αμοιβαία αποκλειόμενο ζευγάρι δράσεων που ανήκει στο σύνολο προτάσεων 7.1, μπορεί επίσης να ανήκει και σε άλλα αμοιβαία αποκλειόμενα σύνολα που είναι UP- περιττά. Επιπλέον το μέγεθος του συνόλου των προτάσεων 7.1 του SATMAXPLAN, μπορεί να μειωθεί με την παράλειψη όλων των προτάσεων του συνόλου που ανήκουν στα σύνολα 7.2 ή 7.3.

Επιπρόσθετα, μπορούν να παραληφθούν, όλες οι προτάσεις των αμοιβαία αποκλειόμενων δράσεων πάνω σε δράσεις A_1 και A_2 με χρόνο T που περιέχουν την προσθήκη των αποτελεσμάτων p_1 και p_2 αντίστοιχα, έτσι ώστε p_1 και p_2 να είναι mutex στο χρόνο T+1. Η κωδικοποίηση αυτή ονομάζεται SATMAXPLAN(SMP).

5.3 Προβλήματα προγραμματισμού δράσης

5.3.1 Προβλήματα προγραμματισμού δράσης Trucks

Τα προβλήματα προγραμματισμού δράσης Trucks[22] ασχολείται με την μεταφορά πακέτων από ένα μέρος σε ένα άλλο με ένα φορτηγό με βάση ένα σύνολο από περιορισμούς. Το φόρτωμα του φορτηγού με τα κιβώτια πρέπει να υπακούει τα πιο κάτω:

Ένα πακέτο μπορεί να φορτωθεί και να εκφορτωθεί σε ένα μέρος του φορτηγού ένα και μόνο εάν η περιοχή που ια μεταφερθεί το πακέτο και η πόρτα του φορτηγού είναι ελεύθερες. Επιπλέον, κάποια πακέτα πρέπει να μεταφερθούν με βάση μια ημερομηνία παράδοσης,

5.3.2 Προβλήματα προγραμματισμού δράσης Pathways

Τα προβλήματα προγραμματισμού δράσης Pathways[22] αφορά το τομέα της μοριακής βιολογίας. Ένα μονοπάτι είναι μια ακολουθία από χημικές αντιδράσεις σε ένα βιολογικό οργανισμό.

Η μοντελοποίηση τμημάτων της λειτουργίας ενός μονοπατιού σε πρόβλημα προγραμματισμού δράσης πραγματοποιείται με την αναπαράσταση των χημικών αντιδράσεων σε δράσεις.

5.3.3 Προβλήματα προγραμματισμού δράσης Storage

Τα προβλήματα προγραμματισμού δράσης Storage[22] ασχολείται με την μετακίνηση ορισμένου αριθμού κιβωτίων από μερικά εμπορευματοκιβώτια σε ορισμένες αποθήκες μέσω ανυψωτήρων.

Οι περιορισμοί είναι ότι μέσα σε μια αποθήκη ένας ανελκυστήρας μπορεί να κινηθεί σύμφωνα με το χάρτη ο οποίος ενώνει διάφορες περιοχές της αποθήκης και ο οποίος δίδεται στην αρχή του κάθε προβλήματος. Η διαφορά των προβλημάτων είναι στον αριθμό των κιβωτίων, ανελκυστήρων, αποθηκών, χώρων αποθηκών και εμπορευματοκιβωτίων.

5.3.4 Προβλήματα προγραμματισμού δράσης Rovers

Τα προβλήματα προγραμματισμού δράσης Rovers[22] ασχολείται με μια συλλογή από οχήματα χωρίς πλήρωμα που χρησιμοποιούνται για την πλοήγηση στην επιφάνεια ενός πλανήτη, να βρίσκουν δείγματα και να επικοινωνούν πίσω στην γη.

5.3.5 Προβλήματα προγραμματισμού δράσης PipesWorld

Τα προβλήματα προγραμματισμού δράσης PipesWorld[22] ασχολείται με την μεταφορά παρτίδων προϊόντων πετρελαίου σε ένα δίκτυο αγωγών, με ή χωρίς περιορισμούς στην χωρητικότητα της ενδιάμεσης αποθήκευσης του πετρελαίου σε δεξαμενές.

5.4 Χρόνοι εκτέλεσης μέσω SATPLANMAX με κωδικοποιήσεις 5 και 6

Στους ακόλουθους πίνακες παρουσιάζονται οι χρόνοι των προβλημάτων προγραμματισμού δράσης Trucks, Pathways,Storage, Rovers, PipesWorld μέσα από το εργαλείο επίλυσης SATMAXPLAN με δύο διαφορετικές κωδικοποιήσεις. Όταν γίνεται χρήση της κωδικοποίησης 5 παρατηρήθηκε πως τα προβλήματα εκτελούνται με γρηγορότερους χρόνους σε αντίθεση με την κωδικοποίηση 6.

Αυτό συμβαίνει για το λόγο ότι από την κωδικοποίηση 5 αφαιρούνται οι UP-περιττές προτάσεις 7.2 και 7.3 και επιπρόσθετα παραλείπονται όλες οι προτάσεις των αμοιβαία αποκλειόμενων ζευγαριών δράσεων. Αυτό συμβαίνει γιατί χρησιμοποιείται μόνο ένα σύνολο των αμοιβαία αποκλειόμενων δράσεων, το 7.1. Το σύνολο αυτό όπως αναφέρθηκε είναι πιθανό να συμπεριλαμβάνεται σε περισσότερα από ένα σύνολα αμοιβαία αποκλειόμενων προτάσεων και κάθε φόρα για διαφορετικό λόγο. Είναι γεγονός ότι η κωδικοποίηση 6 ήταν πιο ισχυρή από τις αρχικές κωδικοποιήσεις. Από την στιγμή όμως που και στις δύο κωδικοποιήσεις δεν χρησιμοποιούνται αμοιβαία αποκλειόμενες προτάσεις, οι οποίες είναι UP – περιττές, αυτό που διαφοροποιεί τις δυο κωδικοποιήσεις και κάνει πιο ισχυρή την κωδικοποίηση 5 είναι η εισαγωγή των προτάσεων 6.

	Trucks	
	Encoding 5	**Encoding 6**
p01	0.876s	1.161s
p02	1.811s	3.631s
p03	12.512s	33.737s
p04	81.140s	363.475s
p05	206.632s	2011.680s
p06	1232.867s	Aborted
p07	97.931s	395.644s
p08	441.172s	2026.163s
p09	2337.327s	Aborted
p10	Aborted	Aborted
p11	Aborted	Aborted
p12	Aborted	Aborted
p13	Aborted	Aborted
p14	Aborted	Aborted
p15	Aborted	Aborted
p16	Aborted	Aborted
p17	Aborted	Aborted
p18	Aborted	Aborted
p19	Aborted	Aborted
p20	Aborted	Aborted

Πίνακας 5.3.5.1

	Pathways	
	Encoding 5	**Encoding 6**
p01	0.202s	0.384s
p02	0.244s	0.254s
p03	0.364s	0.396s
p04	0.360s	0.482s
p05	0.667s	0.732s
p06	2.725s	2.946s
p07	9.694s	6.468s
p08	1587.428s	Aborted
p09	Aborted	Aborted
p10	4.008s	4.747s
p11	3119.013s	Aborted
p12	Aborted	Aborted
p13	Aborted	Aborted
p14	Aborted	Aborted
p15	416.147s	391.145s
p16	Aborted	Aborted
p17	Aborted	Aborted
p18	Aborted	Aborted
p19	Aborted	Aborted
p20	Aborted	Aborted

Πίνακας 5.3.5.2

	Storage	
	Encoding 5	**Encoding 6**
p01	0.178s	0.338s
p02	0.123s	0.197s
p03	0.103s	0.111s
p04	0.207s	0.287s
p05	0.201s	0.223s
p06	0.219s	0.208s
p07	0.559s	1.021s
p08	0.368s	0.760s
p09	0.406s	0.915s
p10	6.460s	28.721s
p11	7.281s	84.721s
p12	2.080s	32.604s
p13	137.586s	114.511s
p14	12.741s	13.181s
p15	3.878s	6.620s
p16	Aborted	Aborted
p17	Aborted	Aborted
p18	Aborted	Aborted
p19	Aborted	Aborted
p20	Aborted	Aborted

Πίνακας 5.3.5.3

	Rovers	
	Encoding 5	**Encoding 6**
p01	0.170s	0.272s
p02	0.302s	0.206s
p03	0.288s	0.390s
p04	0.294s	0.303s
p05	0.374s	0.425s
p06	0.546s	0.643s
p07	0.278s	0.383s
p08	0.566s	0.567s
p09	1.051s	1.111s
p10	0.541s	0.687s
p11	1.024s	1.261s
p12	0.430s	0.527s
p13	1.452s	1.830s
p14	0.988s	1.300s
p15	1.452s	2.691s
p16	1.395s	1.700s
p17	5.312s	6.665s
p18	3.293s	4.605s
p19	15.062s	21.639s
p20	39.570s	73.787s

Πίνακας 5.3.5.4

	Pipesworld	
	Encoding 5	**Encoding 6**
p01	0.338s	0.307s
p02	0.598s	0.607s
p03	1.233s	2.111s
p04	1.273s	2.101s
p05	1.213s	2.311s
p06	1.281s	2.576s
p07	6.542s	18.533s
p08	9.058s	22.455s
p09	113.521s	243.339s
p10	113.723s	147.134s
p11	7.081s	25.998s
p12	377.985s	467.205s
p13	29.344s	101.102s
p14	Aborted	Aborted
p15	2333.634s	3258.259s
p16	Aborted	Aborted
p17	Aborted	Aborted
p18	437.526s	Aborted
p19	605.394s	Aborted
p20	Aborted	Aborted

Πίνακας 5.3.5.4

ΚΕΦΑΛΑΙΟ 6

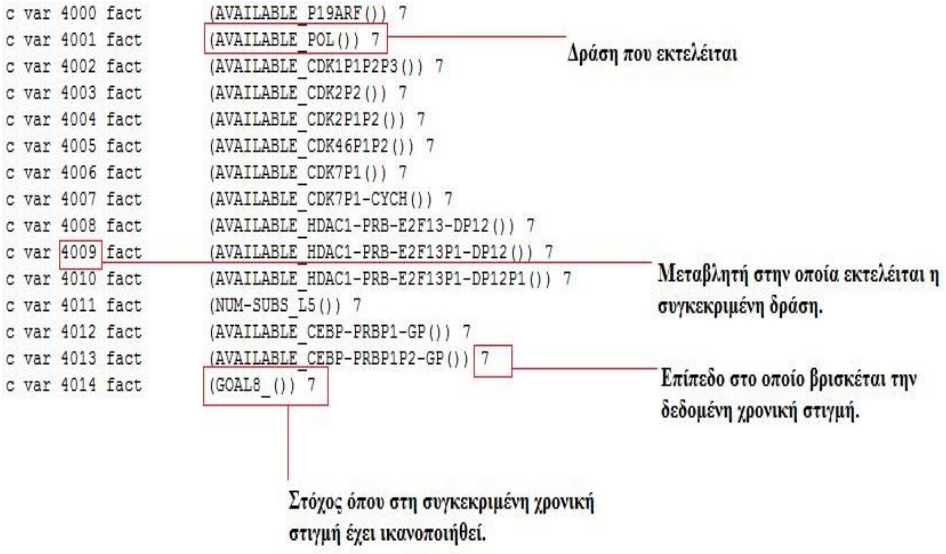
6. Εργαλεία MINISAT+ και BSOLO

6.1 Εισαγωγή

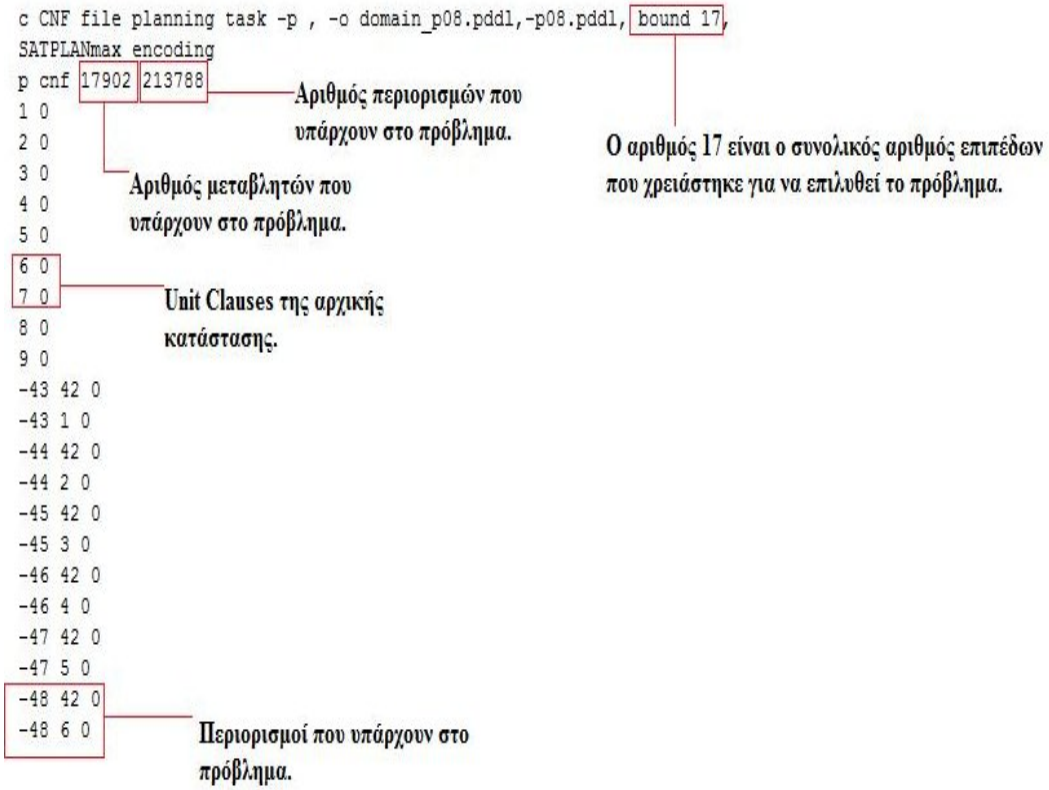
Στο κεφάλαιο αυτό θα μελετηθεί η μετατροπή των αποτελεσμάτων παραγωγής της κωδικοποίησης SATMAXPLAN στην κατάλληλη μορφή εισόδου σε ένα εργαλείο επίλυσης ψευδοδυαδικών προβλημάτων όπως είναι ο BSOLO και το MINISAT+.

Κατά την διάρκεια της εκτέλεση του SATMAXPLAN δημιουργείται ένα αρχείο. Η κωδικοποίηση που περιέχεται στο αρχείο είναι μια απευθείας μετάφραση από την δομή του γραφήματος δράσεων σε προτασιακή λογική. Το αρχείο αυτό περιέχει την αρχική και τελική κατάσταση των μεταβλητών, όλες τις δράσεις που έχουν εκτελεστεί για να φτάσει τελικά στο στόχο και γενικά όλους τους περιορισμούς οι οποίοι υπάρχουν στο πρόβλημα.

Το αρχείο αυτό, όπου η μορφή του απεικονίζεται στην σχήμα 9 και σχήμα 10 αποτελεί το αρχείο εισόδου στο πρόγραμμα που μετατρέπει τους κανονικούς διαζευκτικούς περιορισμούς σε pseudo - Boolean περιορισμούς.

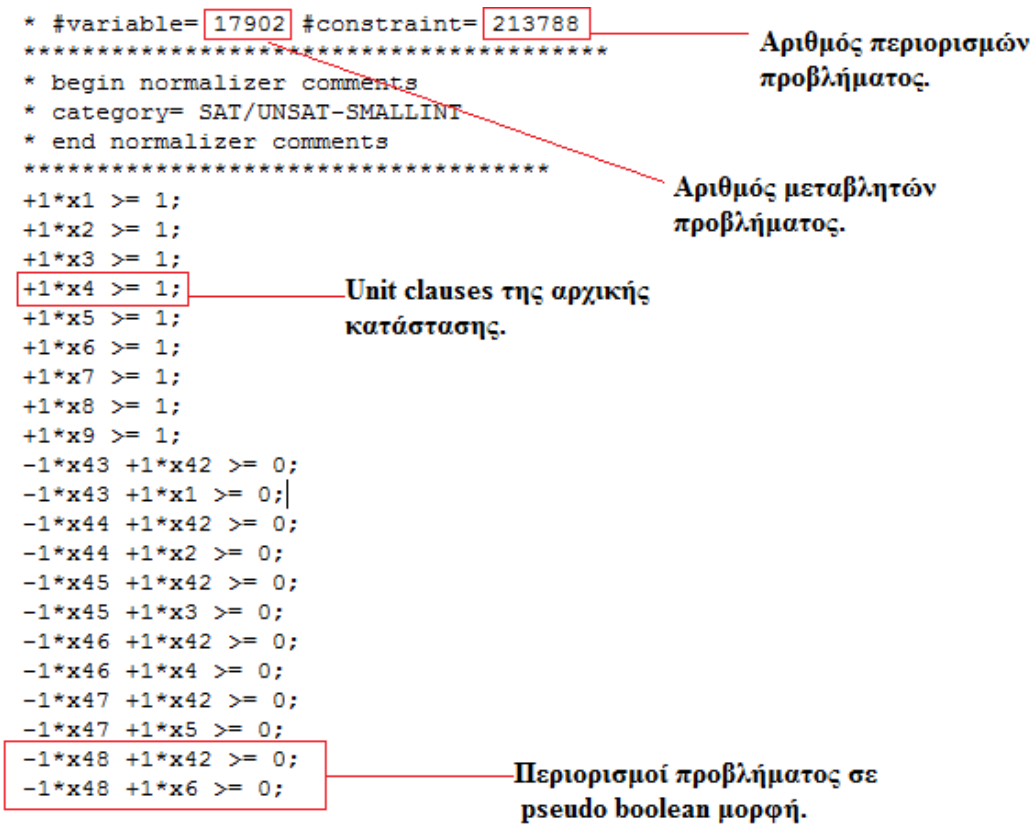


Σχήμα 9 «Μορφή αρχείου»



Σχήμα 10 «Μορφή Αρχείου»

Το πρόγραμμα παίρνει δύο παραμέτρους, το όνομα του αρχείου που βγάζει ο SATPLANMAX και το όνομα του αρχείου το οποίο θα δημιουργήσει στη συνέχεια. Το πρόγραμμα επεξεργάζεται το αρχείο εισόδου γραμμή προς γραμμή. Διαβάζει μια γραμμή κάθε φόρα και μετατρέπει τους όρους σε μεταβλητές που δημιουργούν το νέο ψευδοδυαδικό περιορισμό. Εάν για παράδειγμα το -48 6 0 όπως φαίνεται στο σχήμα 10 περάσει από το πρόγραμμα θα επεξεργαστεί και θα δώσει τον ακόλουθο ψευδοδυαδικό περιορισμό -1*x48 + 1*x6 >= 0 όπως φαίνεται και στην εικόνα 3. Ο χαρακτήρας '-' δηλώνει ότι είναι το συμπλήρωμα της μεταβλητής που δημιουργήθηκε από το προηγούμενο αρχείο. Εκτενέστερη αναφορά για την λειτουργία του προγράμματος εμπερικλείεται στον ψευδοκώδικα που ακολουθεί. Το αρχείο εξόδου έχει την μορφή που απεικονίζεται στο σχήμα 11. Ο ψευδοκώδικας του προγράμματος παρουσιάζεται στο παράρτημα Α.



Σχήμα 11 «Μορφή αρχείου εξόδου»

6.2 Εργαλείο επίλυσης MINISAT+

Τα τελευταία χρόνια τα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής έχουν εξελιχτεί και εφαρμόζονται σε μεγάλο βαθμό τόσο στην Ηλεκτρονική Σχεδίαση όσο και στο τομέα Αυτοματισμού. Ωστόσο η μεγάλη εξέλιξη τους και η χρήση τους σε μεγάλο αριθμό πεδίων εφαρμογής, παρουσίασε την ανάγκη της χρήσης περιορισμών πέραν από την προτασιακή λογική του SAT. Μια δημοφιλής προσέγγιση είναι η χρήση του εργαλείου επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής σε βασικές αποφάσεις για να πετύχει μια πιο υψηλού επιπέδου διαδικασία και να δουλεύει σε πιο πλούσια λογική. Λογική που είναι μέρος ενός βρόγχου που αφαιρεί ή βελτιώνει. Μια επίσης προσέγγιση είναι η επέκταση του

εργαλείου επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής για να είναι σε θέση να χειριστεί και άλλα είδη περιορισμών, η ακόμη να εργαστεί σε διάφορα πεδία ορισμών, όπως για παράδειγμα τα πεπερασμένα σύνολα.[10]

Εδώ θα μελετήσουμε πώς το εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής μπορεί να χρησιμοποιηθεί για την επίλυση ψευδοδυαδικών προβλημάτων μεταφράζοντας τα σε προτάσεις περιορισμών. Τα προβλήματα αυτά, είναι επίσης γνωστά και ως προβλήματα προγραμματισμού γραμμικών ακεραίων 0-1 περιορισμών (integer linear programming ILP). Από την σκοπιά του εργαλείου επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής οι ψευδοδυαδικοί περιορισμοί θεωρούνται σαν μια γενίκευση των προτάσεων περιορισμού. Συγκεκριμένα ένας ψευδοδυαδικός - περιορισμός αποτελεί μια ανισότητα σε ένα γραμμικό συνδυασμό των δυαδικών μεταβλητών. Η ανισότητα είναι της μορφής $C_0 p_0 + C_1 p_1 + C_{n-1} p_{n-1} \geq C_n$ όπου οι μεταβλητές $p_1,..., p_{n-1} \in \{0,1\}$. Αν όλες οι σταθερές C_i είναι 1, τότε ο ψευδοδυαδικός περιορισμός είναι ισοδύναμός με μια τυποποιημένη πρόταση ικανοποιήσιμοτητας προτασιακής λογικής.

Αρκετά εργαλεία επίλυσης ψευδοδυαδικών περιορισμών είναι επέκταση εργαλείων επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής που τροποποιήθηκαν για να υποστηρίζουν τέτοιους περιορισμούς. Μερικά τέτοια εργαλεία επίλυσης είναι τα PBS, PUEBLO, GALENA, και OPBDP.

Πιο κάτω αναλύεται πως χωρίς την τροποποίηση της διαδικασίας επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής οι ψευδοδυαδικοί περιορισμοί μπορούν να διαχειρίζονται και να μεταφράζονται σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής . Οι τεχνικές αυτές εφαρμόστηκαν σε ένα εργαλείο επίλυσης το ονομαζόμενο MINISAT +, που περιλαμβάνει επίσης υποστήριξη συνάρτηση βελτιστοποίησης που έχει ως αποτέλεσμα την βέλτιστη λύση του προβλήματος.

6.2.1 Γενικές γνώσεις

Ένας ψευδοδυαδικός περιορισμός αποτελεί μια ανισότητα της μορφής $C_0 p_0 + C_1 p_1 + C_{n-1} p_{n-1} \geq C_n$, όπου τα p_i είναι όροι και τα C_i ακέραιοι αριθμοί. Ένας όρος παίρνει την τιμή 1 εάν αποτιμάται με τιμή αληθείας True και τιμή 0 εάν αποτιμάται με τιμή αλήθειας False. Η αριστερή πλευρά (left-hand side LHS) είναι οι όροι και στην δεξιά πλευρά (right hand side RHS) η σταθερά C_n. Ο συντελεστής C_i ενεργοποιείται στο πλαίσιο μιας μερικής εκχώρησης, εάν στον όρο p_i ανατεθεί η τιμή αληθείας True. Ένας ψευδοδυαδικός περιορισμός ικανοποιείται εάν μετά την ανάθεση των τιμών, το άθροισμα των συντελεστών υπερβαίνει ή είναι ίσο με την σταθερά C_n που βρίσκεται στην δεξιά πλευρά.

6.2.2 Κανονικοποίηση ψευδοδυαδικών περιορισμών

Οι ψευδοδυαδικοί περιορισμοί βρίσκονται σε υψηλό επίπεδο μη-κανονικής μορφής για το λόγο ότι υπάρχουν πολλές διαφορετικές συντάξεις, αλλά σημασιολογικά είναι όλες ισοδύναμες. Για προφανείς λόγους, θα ήταν πρακτικό να υπάρχει μια κανονική μορφή των ψευδοδυαδικών περιορισμών πριν από την μετατροπή τους σε προβλήματα ικανοποιησιμότητας προτασιακής. Στο MINISAT+ εφαρμόζονται οι ακόλουθοι απλοί κανόνες κατά τη διάρκεια της μεταγλώττισης

• ≤- περιορισμοί αλλάζουν και μετατρέπονται σε ≥- περιορισμούς αναιρώντας σταθερές

• Αρνητικοί συντελεστές εξαλείφονται αλλάζοντας το p σε ¬p και ενημερώνοντας την δεξιά πλευρά.

• Οι πολλαπλές εμφανίσεις της ίδιας μεταβλητής συγχωνεύονται σε ένα όρο Ci x ή Ci ¬x

• Οι συντελεστές ταξινομούνται κατά αύξουσα σειρά: Ci ≤ Cj αν i < j.

• Κοινότυποι περιορισμοί όπως “x + y ≥ 0” διαγράφονται. Ομοίως, κοινότυποι ανικανοποίητοι περιορισμοί (“x + y ≥ 3”) απορρίπτονται και το MINISAT + δίδει απάντηση ότι το πρόβλημα δεν ικανοποιείται.

• Συντελεστές μεγαλύτεροι από την δεξιά πλευρά αντικαθιστούνται από την δεξιά πλευρά.

• Οι συντελεστές της αριστερής πλευράς χωρίζονται με βάση το μέγιστο κοινό διαιρέτη τους. Η δεξιά πλευρά από τη φράση "δεξιά πλευρά/μέγιστος κοινός διαιρέτης", στρογγυλεμένη προς τα πάνω.

Όταν τώρα όλοι οι περιορισμοί έχουν μεταγλωττιστεί αναπαράγονται τετριμμένα συμπεράσματα. Για παράδειγμα θέτεται η μεταβλητή x = True, αν όμως η μεταβλητή πάρει την τιμή False αυτομάτως το πρόβλημα θεωρείται μη ικανοποιήσιμο. Αν η τιμή τώρα της μεταβλητής είναι ίση με x = True τότε η συγκεκριμένη μεταβλητή αφαιρείται από όλους τους περιορισμούς.

Τέλος το εργαλείο επίλυσης εκτελεί ακόμη μια μετατροπή στο ψευδοδυαδικό επίπεδο έτσι ώστε να μειώσει το μέγεθος των μεταφράσεων σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής. Οι ψευδοδυαδικοί περιορισμοί όταν είναι εφικτό χωρίζονται σε ψευδοδυαδικό μέρος και μέρος προτάσεων. Όπου και το τελευταίο αναπαριστάται απευθείας μέσα στο εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής, χωρίς επιπλέον μετάφραση.

6.3 Εργαλείο επίλυσης ψευδοδυαδικών προβλημάτων BSOLO

Ακόμη ένα εργαλείο επίλυσης ψευδοδυαδικών προβλημάτων είναι ο BSOLO. Εδώ υλοποιείται ένας διαφορετικός αλγόριθμος, ο οποίος ενσωματώνει

διάφορα στοιχεία από τους αλγόριθμους επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής σε μια διαδικασία, η όποια υλοποιεί τον αλγόριθμο Κλάδου και Φράγματος(Branch and Bound). Ο αλγόριθμος του BSOLO ενσωματώνει τα πιο βασικά χαρακτηριστικά από τις δύο προσεγγίσεις. Προσπαθεί να χρησιμοποιεί το χαμηλότερο φράγμα από το αλγόριθμό του Κλάδου και Φράγματος και τις τεχνικές κλαδέματος αναζήτησης από τους αλγόριθμους επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής, συμπεριλαμβανομένου και της μη χρονολογικής οπισθοδρόμησης στο δένδρο αναζήτησης και τους μηχανισμούς μάθησης στηριζόμενους σε συγκρούσεις. Λόγω της ουσιαστικής διαδικασίας ανάλυσης των συγκρούσεων που επιτρέπει τον εντοπισμό των μη χρονολογικών οπισθοδρομήσεων, ο αλγόριθμός BSOLO αποδίδει καλύτερα από οποιονδήποτε άλλο αλγόριθμο Κλάδου και Φράγματος σε αρκετά και διαφορετικά στιγμιότυπα. [5,19]

Τα βασικά βήματα του αλγορίθμου BSOLO μπορούν να περιγραφούν ως εξής: Ξεκινώντας είναι η αρχικοποίηση του άνω ορίου με την μεγαλύτερη δυνατή τιμή, που ορίζεται από το ακόλουθο $ub = \sum_{j=1}^{n} c_j + 1$. Στην συνέχεια η συνάρτηση consistent_state ξεκινά με τον έλεγχο κατά πόσο η τρέχουσα κατάσταση παράγει σύγκρουση. Αυτό πραγματοποιείται με την εφαρμογή του unit propagation. Σε περίπτωση που φτάσει σε σύγκρουση, καλείται η διαδικασία αναλύσεως της σύγκρουσης, η μάθηση των προτάσεων και η εκτέλεση της διαδικασίας της οπισθοδρόμησης εάν κριθεί αναγκαίο. Ακολούθως αν έχει βρεθεί μια λύση για τους περιορισμούς, ενημερώνεται το άνω όριο με βάση το ακόλουθο $ub = \sum_{j=1}^{n} c_j \cdot x_j$. Ο μόνος τρόπος για να μειωθεί η τιμή της συγκεκριμένης λύσης είναι η οπισθοδρόμηση με στόχο να βρεθεί καλύτερη λύση με χαμηλότερο κόστος.

Στο τέλος εκτιμάται το κάτω όριο πού δίδεται στην τρέχουσα ανάθεση τιμών στις μεταβλητές. Εάν η τιμή αυτή είναι υψηλότερη ή ίση από την τρέχων

άνω όριο, δημιουργείται ένα όριο σύγκρουσης και η διαδικασία ανάλυσης της σύγκρουσης καλείται να προσδιορίσει σε πιο σημείο στο δένδρο αναζήτησης πρέπει να οπισθοδρομήσει. Στην συνέχεια επιστρέφει πίσω στο δεύτερο βήμα του αλγορίθμου.

```
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 11 «Αλγόριθμος BSOLO»

Στον αλγόριθμό που περιγράφεται στην σχήμα 11 μπορούν να βρεθούν δύο τύποι συγκρούσεων. Η πρώτη ομάδα συγκρούσεων είναι οι λογικές συγκρούσεις που λαμβάνουν χώρα όταν τουλάχιστον ένας από τους περιορισμούς του προβλήματος είναι μη ικανοποιήσιμος. Η δεύτερη ομάδα συγκρούσεων είναι οι συγκρούσεις φράγματος που προκύπτουν όταν το κάτω όριο είναι μεγαλύτερο ή ίσο με το άνω όριο. Όταν παρουσιαστούν οι λογικές συγκρούσεις, ο μη ικανοποιήσιμος ψευδοδυαδικός περιορισμός δίνεται σε μια διαδικασία ανάλυσης της σύγκρουσης, η οποία μαθαίνει μια νέα πρόταση και καθορίζει σε πιο σημείο της διαδικασίας αναζήτησης θα πρέπει να οπισθοδρομήσει.

Επίσης δυνατό είναι και η μάθηση ενός καινούργιου ψευδοδυαδικού περιορισμού. Σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής και προβλήματα βελτίωσης ψευδοδυαδικών προβλημάτων(Pseudo Boolean Optimization), η διαδικασία αυτή είναι συχνά εξαιρετικά αποτελεσματική, επιτρέποντας μη χρονολογική οπισθοδρόμηση. Επιπλέον, η μάθηση των προτάσεων μπορεί να αποφύγει την επίσκεψη αχρείαστων μέρων του δένδρου αναζήτησης, αργότερα κατά τη διαδικασία αναζήτησης. Επιπλέον, κάθε φορά που εντοπίζεται μια σύγκρουση φράγματος , μια νέα πρόταση που εξηγεί την σύγκρουση του φράγματος, θα πρέπει να παρέχει την ανάλυση της σύγκρουσης για να προσδιορίσει σε πιο επίπεδο του δένδρου αναζήτησης μπορεί να οπισθοδρομήσει με ασφάλεια ο αλγόριθμος.

6.4 Αποτελέσματα προβλημάτων σε MINISAT+ και BSOLO

Όπως βλέπουμε στους πίνακες 6.5.1 και 6.5.2 τα αποτελέσματα που δίδει το εργαλείο επίλυσης MINISAT+ είναι καλύτερα από ότι το εργαλείο επίλυσης BSOLO . Συνολικά από τα 69 προβλήματα το εργαλείο επίλυσης MINISAT+ επιλύει όλα τα προβλήματα σε αντίθεση από το εργαλείο επίλυσης BSOLO που καταφέρνει να δώσει λύση μόνο στα 54 από τα 69 προβλήματα. Αξιοσημείωτη είναι και η απόδοση των δύο συστημάτων. Το εργαλείο επίλυσης MINISAT+ μπορεί να βρει την λύση του προβλήματος σε μικρότερο χρονικό διάστημα προσφέροντας περισσότερη αποδοτικότητα σε σχέση με το εργαλείο επίλυσης BSOLO. Το εργαλείο επίλυσης MINISAT+ με την χρήση της κανονικής μορφής απλοποιεί το πρόβλημα δίνοντας λιγότερες περιπτώσεις για έλεγχο γεγονός που το καθιστά πιο γρήγορα από το εργαλείο BSOLO που εξετάζει όλες τις περιπτώσεις και οπισθοδρομεί εάν δεν βρεθεί λύση. Επίσης, το εργαλείο MINISAT+ μπορεί να μειώσει ορισμένους περιορισμούς και να προβεί στις επόμενες μεταφράσεις πιο αποτελεσματικά. Σε αντίθεση με το BSOLO που επιδιώκει την μάθηση των προτάσεων για να μπορεί να αποφύγει την επίσκεψη αχρείαστων μέρων του δένδρου αναζήτησης. Αυτό το οποίο κοστίζει στο

BSOLO είναι όταν εντοπίζεται μια σύγκρουση φράγματος, μια νέα πρόταση που εξηγεί την σύγκρουση του φράγματος, θα πρέπει να παρέχει την ανάλυση της σύγκρουσης έτσι ώστε να προσδιορίσει σε πιο επίπεδο του δένδρου αναζήτησης μπορεί να οπισθοδρομήσει με ασφάλεια ο αλγόριθμος.

	Trucks	**Pathways**	**Storage**	**Rovers**	**Pipewords**
p01	0.273s	0.094s	0.065s	0.031s	0.150s
p02	0.635s	0.158s	0.063s	0.074s	0.282s
p03	1.653s	0.211s	0.056s	0.077s	0.650s
p04	2.432s	0.200s	0.079s	0.065s	0.394s
p05	164.844s	0.388s	0.084s	0.054s	0.435s
p06	5.277s	1.749s	0.081s	0.161s	0.526s
p07	2.903s	2.299s	0.248s	0.151s	2.614s
p08	20.302s	17.502s	0.201s	0.216s	4.196s
p09	43.496s	Aborted	0.214s	0.384s	89.061s
p10	Aborted	1.697s	1.243s	0.218s	112.895s
p11	Aborted	442.764s	0.561s	0.349s	7.645s
p12	Aborted	Aborted	1.932s	0.151s	241.251s
p13	Aborted	Aborted	14.574s	0.441s	35.181s
p14	Aborted	Aborted	1.104s	0.363s	Aborted
p15	Aborted	113.157s	1.091s	0.375s	Aborted
p16	Aborted	Aborted	Aborted	0.431s	Aborted
p17	Aborted	Aborted	Aborted	1.095s	Aborted
p18	Aborted	Aborted	Aborted	0.846s	1376.064s
p19	Aborted	Aborted	Aborted	2.148s	128.909s
p20	Aborted	Aborted	Aborted	3.413s	Aborted

Πίνακας 6.5.1:Χρόνοι MINISAT+

	Trucks	**Pathways**	**Storage**	**Rovers**	**Pipewords**
p01	0.728s	0.113s	0.070s	0.036s	0.159s
p02	7.253s	0.073s	0.108s	0.034s	0.498s
p03	99.875s	0.510s	0.115s	0.088s	1.887s
p04	624.688s	0.472s	0.097s	0.084s	0.653s
p05	Can't solve	0.653s	0.082s	0.172s	1.959s
p06	Can't solve	44.599s	0.109s	0.422s	0.987s
p07	Can't solve	528.901s	1.806s	0.155s	29.810s
p08	Can't solve	Can't solve	3.769s	0.536s	86.109s
p09	Can't solve	Aborted	1.612s	1.090s	Can't solve
p10	Aborted	Can't solve	198.413s	0.631s	Can't solve
p11	Aborted	Can't solve	4.486s	4.995s	Can't solve
π12	Aborted	Aborted	6.284s	0.394s	Can't solve
p13	Aborted	Aborted	11.964s	3.208s	Can't solve
p14	Aborted	Aborted	27.786s	1.002s	Aborted
p15	Aborted	Can't solve	8.689s	4.291s	Can't solve
p16	Aborted	Aborted	Aborted	4.143s	Aborted
p17	Aborted	Aborted	Aborted	12.940s	Aborted
p18	Aborted	Aborted	Aborted	4.983s	Can't solve
p19	Aborted	Aborted	Aborted	54.266s	Can't solve
p20	Aborted	Aborted	Aborted	136.152s	Aborted

Πίνακας 6.5.2:Χρόνοι BSOLO

ΚΕΦΑΛΑΙΟ 7

7. Βελτίωση Ψευδοδυαδικών Προβλημάτων

7.1 Εισαγωγή

Την εξέλιξη των αλγορίθμων που αφορούν την προτασιακή ικανοποιησιμότητα διαδέχθηκαν οι ψευδοδυαδικοί αλγόριθμοι και οι αλγόριθμοι βελτιστοποίησης (Pseudo Boolean Optimization PBO) που έχουν σαν στόχο την εύρεση βέλτιστων λύσεων. Η εξέλιξη στους τομείς αυτούς έφερε ως αποτέλεσμα την χρήση των αποδοτικότερων τεχνικών που χρησιμοποιούνται στη προτασιακή ικανοποιησιμότητα όπως μάθηση διαζευκτικών προτάσεων, τις δομές lazy data, τις μεθόδους διάγνωσης σύγκρουσης και την επέκταση τους σε ψευδοδυαδικούς αλγόριθμους βελτιστοποίησης.[11,12]

Στους ψευδοδυαδικούς αλγόριθμους βελτιστοποίησης στόχος είναι η εύρεση μιας ανάθεση τιμών η οποία θα ελαχιστοποιεί το κόστος της συνάρτησης αλλά ταυτοχρόνως θα ικανοποιεί όλους τους περιορισμούς. Ένα απλό ψευδοδυαδικό πρόβλημα όταν επεκταθεί μπορεί εύκολα να μετατραπεί σε ψευδοδυαδικό πρόβλημα βελτιστοποίησης με μια μικρή μετατροπή. Όταν βρεθεί η ανάθεση των τιμών που ικανοποιεί το πρόβλημα, ο αλγόριθμος να μην σταματάει αλλά να παίρνει την τιμή της συγκεκριμένης λύσης και να προσθέτει ένα καινούργιο περιορισμό έτσι ώστε η καινούργια λύση να αποτελεί βελτίωση της προηγούμενης(σχήμα 12).

```
1: function PBO_SOLVER(P)
2:     ub ← +∞
3:     while true do
4:         if Solution_Found(p) then
5:             ub ← Σ_{j∈N} c_j v(x_j)
6:             add constraint Σ_{j∈N} c_j x_j ≤ ub − 1 to P
7:         else
8:             Decide(P)
9:         end if
10:        while Deduce(P)=CONFLICT do
11:            if Diagnose(P)=CONFLICT then
12:                return ub
13:            end if
14:        end while
15:    end while
16: end function
```

Σχήμα 12 «Αλγόριθμος Βελτιστοποίησης»

Το σκεπτικό της μεθόδου αυτής είναι η εκτέλεση της γραμμικής αναζήτησης για εύρεση μεταβλητών που θα ικανοποιούν τη συνάρτηση κόστους ξεκινώντας από την μεγαλύτερη τιμή. Μετά σε κάθε επόμενο βήμα το κόστος της συνάρτησης θα μειώνεται και η διαδικασία θα επαναλαμβάνεται έως ότου βρεθεί η βέλτιστη λύση. Αν φτάσει σε σημείο που το αποτέλεσμα δεν ικανοποιείται τότε η βέλτιστη λύση είναι αυτή που βρέθηκε ένα βήμα προηγουμένως

7.2 Μέθοδοι Βελτιστοποίησης

Μια κλασσική προσέγγιση ψευδοδυαδικού προβλήματος βελτιστοποίησης είναι η χρήση της προς τα πίσω αναζήτησης στο χώρο, δηλαδή η χρήση του αλγορίθμου Κλάδου και Φράγματος. Η μέθοδος αυτή χρησιμοποιεί ένα άνω όριο που υπολογίζεται με την αξία της συνάρτησης του κόστους. Η τιμή αυτή ενημερώνεται όταν βρίσκεται ένα χαμηλότερο κόστος λύσης για τα προβλήματα. Σε κάθε κόμβο του δένδρου αναζήτησης εντοπίζεται το κατώτερο όριο για το κόστος της συνάρτησης με βάση τις μέχρι τώρα αναθέσεις τιμών. Εάν τώρα η τιμή αυτή είναι μεγαλύτερη ή ίση από την τιμή του άνω ορίου τότε η αναζήτηση σταματάει και «κλαδεύει» το συγκεκριμένο κόμβο καθώς δεν μπορεί να επεκταθεί η μέχρι τώρα ανάθεση τιμών στο συγκεκριμένο κόμβο.

Ιδίως για προβλήματα με χαμηλούς περιορισμούς, η χρήση του κατώτερου ορίου είναι πολύ σημαντική για την αποδοτικότητα του αλγορίθμου, επειδή υψηλότερες τιμές που υπολογίζονται στους κόμβους κλαδεύονται μέσω της αναζήτησης νωρίτερα. Αρκετές τεχνικές μπορούν να χρησιμοποιηθούν για την εκτίμηση του κατώτερου ορίου. Μερικές από αυτές είναι η μεγίστη ανεξαρτησία των περιορισμών (Maximum Independent set of constraints (MIS)) ή η χαλάρωση του γραμμικού προγραμματισμού(Linear-programming relaxations).

7.3 Δημιουργία Συνάρτησης Βελτιστοποίησης (Objective Function)

Στο κομμάτι που ακολουθεί θα μελετηθεί η δημιουργία της συνάρτησης βελτιστοποίησης. Η δημιουργία της συνάρτησης βελτιστοποίησης (Objective Function) γίνεται με τον ακόλουθο τρόπο. Από το αρχείο του προβλήματος εντοπίζονται οι στόχοι του που πρέπει να ικανοποιηθούν.

Αν για παράδειγμα το αρχείο του προβλήματος δηλώνει ότι ο στόχος είναι το «(goal1)» θα πρέπει από το αρχείο εξόδου του εργαλείου επίλυσης SATMAXPLAN να βρεθούν οι μεταβλητές οι οποίες αντιστοιχούν στο συγκεκριμένο στόχο κατά το τελευταίο επίπεδο της λύσης. Αν για παράδειγμα το πρόβλημα επιλύθηκε σε 12 επίπεδα θα πρέπει να βρεθεί η μεταβλητή στην οποία ο στόχος «(goal1)» πραγματοποιείται στο επίπεδο 12.
Η μορφή θα είναι η ακόλουθη:

c var 9266 action (GOAL1( )) 12
c var 9268 action (GOAL3( )) 12
c var 9270 action (GOAL2( )) 12

Στην συνέχεια οι μεταβλητές στις οποίες αντιστοιχούν οι στόχοι εντάσσονται στη συνάρτηση βελτιστοποίησης(objective function) του αρχείο που έχει σαν είσοδο το εργαλείο επίλυσης ψευδοδυαδικών προβλημάτων όπως αναγράφεται στη συνέχεια με το αρνητικό πρόσημο για μεγιστοποίηση του προβλήματος

min: -1*x9266 -1*x9268 -1*x9270

και ακολούθως από το αρχείο αφαιρούνται και τα unit clauses. Δηλαδή αφαιρούνται από το αρχείο οι περιορισμοί της μορφής

+1*x9266 >= 1;
+1*x9268 >= 1;

+1*x9270 >= 1;

Στους πίνακες 7.3.1 και 7.3.2 αναφέρονται συγκριτικά οι χρόνοι εκτέλεσης ψευδοδυαδικών προβλημάτων Pathways και Trucks με το εργαλείο MINISAT+ όταν υπάρχει και όταν δεν υπάρχει βέλτιστη συνάρτηση στο επίπεδο στο οποίο βρίσκει λύση το εργαλείο SATMAXPLAN καθώς δύο και τέσσερα βήματα πριν να βρεθεί η λύση.

	Pathways Without Objective	**Pathways With Objective function**	**Pathways With Objective function 2 steps before solution**	**Pathways With Objective function 4 steps before solution**
P01	0.094s	Time: 0.033s Steps:5 Goals:1	No goals in this step	No goals in this step
P02	0.158s	Time:0.094s Steps:7 Goals:2	Time:0.124s Steps:5 Goals:1	No goals in this step
P03	0.211s	Time: 0.191s Steps:8 Goals:3	Time: 0.164s Steps:6 Goals1	No goals in this step
P04	0.301s	Time: 0.171s Steps:8 Goals:4	Time: 0.131s Steps:6 Goals:3	No goals in this step
P05	0.442s	Time: 0.351s Steps:9 Goals:5/6	Time: 0.278s Steps:7 Goals:3/6	Time:0.235s Steps:5 Goals:1/6
P06	1.749s	Time: 1.123s Steps:12 Goals:7/8	Time:0.904s Steps:10 Goals:5/8	Time:0.663s Steps:8 Goals:3/8
P07	2.299s	Time:5.050s Steps:13 Goals:10/11	Time:4.796s Steps:11 Goals:6/11	Time:1.185s Steps:9 Goals:2/11
P08	13.578s	Time:2.067s Steps:17 Goals:11/12	Time: 600.989s Steps:15 Goals:8/12	Time:159.365s Steps:13 Goals:5/12
P10	1.697s	Time:2.264s Steps:15 Goals:14/14	Time:1.779s Steps:13 Goals:11/14	Time: 1.525s Steps:11 Goals:7/14
P11	442.764s	Time:256.567s Steps:17 Goals:15	Time:94.064s Steps:15 Goals:12	Time:52.533s Steps:13 Goals:12
P15	113.157s	Time: 92.701s Steps:18 Goals:19/19	Time: 234.258s Steps:16 Goals:15/19	Time:678.419s Steps:14 Goals:13/19

Πίνακας 7.3.1

	Trucks Without Objective	**Trucks With Objective function**	**Trucks With Objective function 2 steps before solution**	**Trucks With Objective function 4 steps before solution**
P01	0.350s	Time:0.321s Steps: 11 Goals:3/3	Time:0.285s Steps: 9 Goals:2/3	Time: 0.213s Steps: 7 Goals:2/3
P02	0.718s	Time:0.657s Steps:14 Goals:3/4	Time:0.671s Steps:12 Goals:3/4	Time:0.673 Steps:10 Goals:2/4
P03	1.653s	Time: 4.904s Steps:16 Goals:4/5	Time:2.311s Steps:14 Goals:4/5	Time:5.038s Steps:2 Goals:3/5
P04	3.005s	Time: 24.695s Steps:18 Goals:5/6	Time:12.491s Steps:16 Goals:4/6	Time:20.422s Steps:14 Goals:4/6
P05	223.538s	Time: 41.253s Steps:19 Goals:5/6	Time: 109.357s Steps:17 Goals:5/6	Time: 504.706s Steps:15 Goals:5/6
P06	5.277s	Time: 52.884s Steps:23 Goals:7/8	Time:435.575s Steps:21 Goals:5/8	Time: 1111.825s Steps:19 Goals:3/8
P07	3.170s	Time: 67.166s Steps:18 Goals:5/6	Time:42.937s Steps:16 Goals:4/6	Time: 63.179s Steps:14 Goals:4/6
P08	20.302s	Time:517.606s Steps:19 Goals:7/7	Time:477.712s Steps:17 Goals:5/7	Time: 548.737s Steps:15 Goals:5/7
P09	43.496s	Time: 773.608s Steps:21 Goals:7/8	Time: 1265.064s Steps:19 Goals:6/8	Time:1552.860s Steps:17 Goals:6/8

Πίνακας 7.3.2

Όπως διαπιστώνεται από τα πιο πάνω αποτελέσματα δαπανάται περισσότερος χρόνος για να βρεθεί η βέλτιστη λύση του προβλήματος.

Επιπλέον, παρατηρείται πώς όταν προσπαθεί το εργαλείο επίλυσης MINISAT+ να βρει την βέλτιστη λύση του προβλήματος σε τέσσερα και δύο επίπεδα πριν την κανονική του λύση, ο χρόνος εκτέλεσης είναι αρκετά μεγαλύτερος και όπως είναι λογικό δεν ικανοποιούνται όλοι οι στόχοι του προβλήματος.

Επίσης, μέσα από τα αποτελέσματα διακρίνεται ότι η βέλτιστη λύση στα ψευδοδυαδικά προβλήματα δεν βρίσκεται στα ίδια βήματα με την βέλτιστη λύση του εργαλείου SATMAXPLAN. Το εργαλείο βρίσκει μεν λύση αλλά δεν καταφέρνει να βρει την βέλτιστη λύση για όλους τους στόχους.

Εφόσον η βέλτιστη λύση των ψευδοδυαδικών προβλημάτων δεν μπορεί να βρεθεί στα ίδια βήματα με την βέλτιστη λύση που δίνει το εργαλείο SATMAXPLAN θα δοκιμαστεί μια άλλη μέθοδος. Με την μέθοδο αυτή θα μελετηθεί κατά πόσο είναι πιο γρήγορος ο χρόνος εκτέλεσης αλλά και κατά πόσο βρίσκεται η βέλτιστη λύση στα ίδια χρονικά βήματα. Αρχίζοντας με τέσσερα βήματα πριν την λύση του προβλήματος εντοπίζονται οι στόχοι οι οποίοι υλοποιούνται την δεδομένη χρονική στιγμή. Ακολούθως οι μεταβλητές που αντιστοιχούν στους στόχους εισάγονται στην συνάρτηση και αφαιρούνται όλα τα unit clauses. Μετέπειτα οι στόχοι για τους οποίους βρέθηκε η βέλτιστη λύση αφαιρούνται από τη συνάρτηση βελτιστοποίησης και εισάγονται σαν unit clauses στο αρχείο. Στην συνάρτηση τώρα εισάγονται οι μεταβλητές που αντιστοιχούν στους εναπομείναντες στόχους δύο βήματα πριν από την λύση. Η ίδια διαδικασία συνεχίζεται έως ότου για όλους τους στόχους βρεθεί η βέλτιστη λύση του προβλήματος. Στους πίνακες 7.3.3, 7.3.4, 7.3.5 και 7.3.6 παρουσιάζονται οι χρόνοι εκτέλεσης των προβλημάτων Pathways Trucks Storage και Openstacks με την πιο πάνω μέθοδο καθώς. Επίσης, παρουσιάζεται σε ποιο βήμα βρίσκεται η βέλτιστη λύση.

Problems					
P01	Time: 0m0.033s Steps:5 Optimum Found:1				
P02	Time:0m0.094s Steps:7 Optimum Found :2				
P03	Time: 0m0.169s Steps:8 Optimum Found:3				
P04	Time: 0m0.168s Steps:8 Optimum Found:4				
P05	Time:0.235s Steps:5 Optimum Found:1/6	Time:0.226s Steps:7 Optimum Found:1/5	Time: 0.351s Steps:9 Optimum Found:2/4	Time: 0.318s Steps:10 Optimum Found:1/2	Time: 0.333s Steps:11 Optimum Found:1/1
P06	Time:0.776s Steps:8 Optimum Found:3/8	Time:0.778s Steps:10 Optimum Found:3/5	Time: 0.661s Steps:12 Optimum Found:1/2	Time: 0.730s Steps:13 Optimum Found:0/1	Time: 0.846s Steps:14 Optimum Found:1/1
P07	Time:1.185s Steps:9 Optimum Found:5/11	Time:1.380s Steps:11 Optimum Found:3/6	Time:1.235s Steps:13 Optimum Found:2/3	Time:1.198s Steps:14 Optimum Found:0/1	Time:1.259s Steps:15 Optimum Found:1/1

Πίνακας 7.3.3
Προβλήματα Pathways μέχρι να βρεθεί η βέλτιστη λύση

p08	Time:159.365s Steps:13 Optimum Found:6/12	Time: 11.781s Steps:15 Optimum Found:3/6	Time:12.110s Steps:17 Optimum Found:2/3	Time:11.953s Steps:18 Optimum Found:0/1	Time:3.192s Steps:19 Optimum Found:1/1
p10	Time: 1.525s Steps:11 Optimum Found:7/14	Time:1.477s Steps:13 Optimum Found:4/7	Time:1.500s Steps:15 Optimum Found:2/3	Time:1.878s Steps:15 Optimum Found:1/1	
p11	Time:52.533s Steps:13 Optimum Found:12/15	Time:20.912s Steps:15 Optimum Found:1/3	Time:20.031s Steps:17 Optimum Found:1/2	Time:4.937s Steps:18 Optimum Found:1/1	
p12	Time: 92.701s Steps:15 Optimum Found:10/16	Time: 92.701s Steps:17 Optimum Found:1/6	Time: 74.689s Steps:19 Optimum Found:3/5	Time:11.102s Steps:20 Optimum Found:2/2	
p13	Time: 92.701s Steps:15 Optimum Found:1/17	Time: 92.701s Steps:17 Optimum Found:1/2	Time: 74.689s Steps:19 Optimum Found:1/1		
p15	Time:678.419s Steps:14 Optimum Found:13/19	Time: 5.139s Steps:16 Optimum Found:3/6	Time: 9.467s Steps:18 Optimum Found:2/3	Time:4.969s Steps:19 Optimum Found:1/1	

Συνέχεια πίνακα 7.3.3
Προβλήματα Pathways μέχρι να βρεθεί η βέλτιστη λύση

Problems				
p01	Time: 0.213s Steps: 7 Optimum Found:2/3	Time:0.162 Steps: 9 Optimum Found:0/1	Time:0.235s Steps: 11 Optimum Found:1/1	
p02	Time:0.673 Steps:10 Optimum Found:2/4	Time:0.449s Steps:12 Optimum Found:1/2	Time:0.529s Steps:14 Optimum Found:0/1	Time:0.491s Steps:16 Optimum Found:1/1
p03	Time:5.038s Steps:2 Optimum Found:3/5	Time:1.249s Steps:14 Optimum Found:1/2	Time: 1.140s Steps:16 Optimum Found:0/1	Time:1.580s Steps:19 Optimum Found:1/1
p04	Time:20.422s Steps:14 Optimum Found:4/6	Time:2.660s Steps:16 Optimum Found:1/2	Time: 2.272s Steps:18 Optimum Found:0/1	Time: 1.833s Steps:21 Optimum Found:1/1
p05	Time: 504.706s Steps:15 Optimum Found:5/7	Time:19.856s Steps:17 Optimum Found:1/2	Time: 7.400s Steps:19 Optimum Found:0/1	Time: 8.682s Steps:20 Optimum Found:1/1
p06	Time: 3450.777s Steps:19 Optimum Found:6/8	Time:57.838s Steps:21 Optimum Found:1/2	Time:97.435s Steps:23 Optimum Found:1/2	Time:29.866s Steps:24 Optimum Found:1/1
p07	Time: 63.179s Steps:14 Optimum Found:4/6	Time:7.492s Steps:16 Optimum Found:0/2	Time:4.878s Steps:18 Optimum Found:1/2	Time:4.878s Steps:18 Optimum Found:1/2
p08	Time:548.737s Steps:15 Optimum Found:5/7	Time:25.094s Steps:17 Optimum Found:0/2	Time:19.213s Steps:19 Optimum Found:2/2	
p09	Time:1359.098s Steps:17 Optimum Found:6/8	Time:90.772s Steps:19 Optimum Found:0/2	Time: 122.788s Steps:21 Optimum Found:1/2	Time:107.169s Steps:24 Optimum Found:1/1

Πίνακας 7.3.4
Προβλήματα Trucks μέχρι να βρεθεί η βέλτιστη λύση

Problems				
p01	Time: 0.008s Steps:3 Optimum Found:1/1			
p02	Time: 0.010s Steps:3 Optimum Found:1/1			
p03	Time: 0.013s Steps:3 Optimum Found:1/1			
p04	Time: 0.040s Steps:4 Optimum Found:1/2	Time: 0.048s Steps:6 Optimum Found:0/1	Time: 0.050s Steps:6 Optimum Found:1/1	
p05	Time: 0.075s Steps:4 Optimum Found:1/2	Time: 0.041s Steps:6 Optimum Found:1/1		
p06	Time: 0.076s Steps:4 Optimum Found:1/2	Time: 0.041s Steps:6 Optimum Found:1/1		
p07	Time: 0.189s Steps:10 Optimum Found:1/3	Time: 0.171s Steps:12 Optimum Found:1/2	Time: 0.211s Steps:14 Optimum Found:1/1	
p08	Time: 0.200s Steps:4 Optimum Found:1/3	Time: 0.178s Steps:6 Optimum Found:1/2	Time: 0.194s Steps:8 Optimum Found:1/1	
p09	Time: 0.260s Steps:3 Optimum Found:1/3	Time: 0.164s Steps:5 Optimum Found:1/2	Time: 0.126s Steps:7 Optimum Found:1/1	
p10	Time: 0.775s Steps:14 Optimum Found:3/4	Time: 0.706s Steps:16 Optimum Found:0/1	Time: 0.617s Steps:18 Optimum Found:1/1	

Πίνακας 7.3.5
Προβλήματα Storage μέχρι να βρεθεί η βέλτιστη λύση

Problems				
p11	Time: 0.520s Steps:7 Optimum Found:3/4	Time: 0.455s Steps:9 Optimum Found:1/1		
p12	Time: 0.560s Steps:5 Optimum Found:2/4	Time: 0.589s Steps:7 Optimum Found:1/2	Time: 0.555s Steps:9 Optimum Found:1/1	
p13	Time: 3.480s Steps:14 Optimum Found:4/5	Time: 2.021s Steps:16 Optimum Found:0/1	Time: 1.206s Steps:18 Optimum Found:1/1	
p14	Time:0.665s Steps:7 Optimum Found:3/5	Time: 0.632s Steps:9 Optimum Found:1/2	Time: 0.593s Steps:11 Optimum Found:0/1	Time: 0.867s Steps:13 Optimum Found:1/1
p15	Time:0.751s Steps:5 Optimum Found:3.5	Time: 0.799s Steps:7 Optimum Found:1/2	Time: 0.817s Steps:9 Optimum Found:1/1	

Συνέχεια πίνακα 7.3.5
Προβλήματα Storage μέχρι να βρεθεί η βέλτιστη λύση

Problems				
p01	Time: 11.743s Steps:19 Optimum Found:3/5	Time: 9.946s Steps:21 Optimum Found:0/2	Time: 1.059s Steps:23 Optimum Found:2/2	
p02	Time: 12.265s Steps:19 Optimum Found:3/5	Time: 18.678s Steps:21 Optimum Found:0/2	Time: 4.063s Steps:23 Optimum Found:2/2	
p03	Time: 22.427s Steps:19 Optimum Found:3/5	Time: 10.473s Steps:21 Optimum Found:0/2	Time: 0.851s Steps:23 Optimum Found:2/2	
p04	Time:18.693s Steps:19 Optimum Found:3/5	Time: 13.039s Steps:21 Optimum Found:0/2	Time: 1.343s Steps:23 Optimum Found:2/2	
p05	Time:23.093s Steps:19 Optimum Found:3/5	Time: 15.037s Steps:21 Optimum Found:0/2	Time: 1.896s Steps:23 Optimum Found:2/2	

Πίνακας 7.3.6
Προβλήματα Openstacks μέχρι να βρεθεί η βέλτιστη λύση

Όπως έχει ήδη αναφερθεί στους πιο πάνω πίνακες παρουσιάζονται οι χρόνοι εκτέλεσης των προβλημάτων Pathways, Trucks, Storage και Openstacks που χρησιμοποιούν την προαναφερθείσα μέθοδο για εύρεση της βέλτιστης λύσης. Μέσα από τα αποτελέσματα τα οποία λήφθηκαν παρατηρείται ότι σε αρκετές περιπτώσεις ο χρόνος εύρεσης της βέλτιστης συνάρτησης δεν πραγματοποιείται στο ίδιο χρονικό βήμα όπως στο εργαλείο SATMAXPLAN. Σε αρκετά προβλήματα από το πεδίο Storage η εύρεση της βέλτιστης λύσης συμπίπτει αλλά σε ποιο δύσκολα προβλήματα όπως είναι τα προβλήματα του πεδίου Trucks η βέλτιστη λύση βρίσκεται σε περισσότερα βήματα. Όπως είναι λογικό κάθε επιπλέον βήμα δημιουργεί περισσότερους περιορισμούς και μεταβλητές.

Αξιοσημείωτο είναι όμως το γεγονός ότι αν και ο αλγόριθμός χρειάζεται περισσότερα χρονικά βήματα για να βρει την βέλτιστη λύση οι χρόνοι εκτέλεσης οι οποίοι καταγράφονται στα πλείστα προβλήματα είναι εμφανέστατα ποιο γρήγοροι. Στον πινάκα 5.3.5.2 του κεφαλαίου 5 που καταγράφονται οι χρόνοι εκτέλεσης των προβλημάτων μέσω του SATMAXPLAN παρατηρείται ότι το πρόβλημα p11 του πεδίου Pathways χρειάζεται 3119.013s δευτερόλεπτα για να φτάσει σε λύση η οποία λύση βρίσκεται στο επίπεδο 17. Η βέλτιστη λύση τώρα μέσω του MINISAT+ βρίσκεται στο επίπεδο 18 και έχει συνολικό χρόνο εκτέλεσης μέχρι την βέλτιστη λύση μόλις 98.213s δευτερόλεπτα. Πιο γρήγορη εύρεση της λύσης μέσω του MINISAT+ παρατηρείται και σε προβλήματα που βρίσκουν την βέλτιστη λύση στα ίδια χρονικά βήματα. Για παράδειγμα το πρόβλημα p05 του πεδίου Openstacks. Η λύση του μέσω SATMAXPLAN είναι 702.946s δευτερόλεπτα στο επίπεδο 23 σε αντίθεση με το συνολικό χρόνο εκτέλεσης του MINISAT+ που είναι 40.026s δευτερόλεπτα σε 23 επίπεδα. Αυτό συμβαίνει επειδή σε κάθε επίπεδο πλέον μειώνονται οι στόχοι οι οποίοι χρειάζονται να βελτιστοποιηθούν καθιστώντας έτσι πιο εύκολη και γρήγορη την αναζήτηση των εναπομεινάντων στόχων.

ΚΕΦΑΛΑΙΟ 8

8. Συμπεράσματα

8.1 Σύνοψη

8.2 Μελλοντική Εργασία

8.1 Σύνοψη

Η συνεχής ανάπτυξη και μελέτη της τεχνητής νοημοσύνης προϋποθέτει την συνεχή εύρεση καινούργιων μεθόδων αλλά και αλγόριθμων για εύρεση λύσεων με όση το δυνατό καλύτερη απόδοση. Μέρος της έρευνας της τεχνητής νοημοσύνης είναι και η επίλυση προβλημάτων προγραμματισμού δράσης που στοχεύει στην επίτευξη της δημιουργίας αυτόνομου έλεγχου σε περίπλοκα συστήματα. Ουσιαστικά ο στόχος της επίλυσης προβλημάτων προγραμματισμού δράσης είναι να εντοπίσει μια ακολουθία από πράξεις όπου από την αρχική κατάσταση ενός προβλήματος, μεταβαίνει στο στόχο του και στην επίλυση του. Αρκετοί αλγόριθμοι των προβλημάτων προγραμματισμού δράσεων έχουν εφαρμοστεί επιτυχώς σε τομείς όπως η ρομποτική, σε συλλογή πληροφοριών, σε έλεγχο εκτόξευσης πυραύλων κ.α.

Η προσπάθεια επίλυσης τέτοιων προβλημάτων γινόταν με την χρήση αλγορίθμων όπως ο Depth first, ο A* κ.α. Με την χρήση όμως αυτών των αλγορίθμων προέκυψαν αρκετές δυσκολίες εφόσον υπήρχαν αρκετές περιττές δράσεις καθιστώντας την λύση μη αποδοτική.

Κατά την εξέλιξη του ο τομέας επίλυσης προβλημάτων ικανοποιήσιμοτητας προτασιακής λογικής δημιούργησε μια βασική γλώσσα αναπαράστασης που διασπά το πρόβλημα σε καταστάσεις, πράξεις και στόχους. Η διάσπαση αυτή βοηθά τους αλγορίθμους να εκμεταλλεύονται πλήρως την λογική δομή του προβλήματος αλλά και να εφαρμόζουν πράξεις επάνω στην ίδια τη γλώσσα. Η βασική γλώσσα αναπαράστασης ενός κλασσικού σχεδιαστή είναι γνωστή και σαν STRIPS γλώσσα. Η γλώσσα STRIPS είναι ένα μέρος των εργαλείων που επιλύουν το πρόβλημα οι οποίοι αναζητούν μέσα στις αναπαραστάσεις του κόσμου ένα μοντέλο που υλοποιεί το στόχο. Για κάθε μοντέλο του κόσμου θεωρείται ότι υπάρχει ένα σύνολο από εφαρμόσιμους τελεστές, όπου καθένας από αυτούς μετατρέπει το μοντέλο σε κάποιο άλλο μοντέλο κόσμου.

Υπήρξαν αρκετοί αλγόριθμοι που προσπαθούσαν να λύσουν τα προβλήματα όσο το δυνατό πιο αποδοτικά. Μερικοί από τους αλγόριθμους αυτούς ήταν, η προς τα εμπρός αναζήτηση στο χώρο, η προς τα πίσω αναζήτηση στο χώρο, οι ευρετικές συναρτήσεις για αναζήτηση στο χώρο, ο Partial Order Planner και οι ευρετικές συναρτήσεις για Partial Order Planner. Αρκετές όμως από τις ευρετικές συναρτήσεις που προτάθηκαν αποδείχθηκαν ανακριβείς, Μετέπειτα την εμφάνιση της έκανε μια ειδική δομή, η λεγόμενη γράφημα δράσεων που προσφέρει ένα μέσο οργάνωσης και διατήρησης των πληροφοριών αναζήτησης καθώς και αποδοτικότερες λύσεις σε προβλήματα δυναμικού προγραμματισμού. Σημαντικό ρόλο έχει και στην επίλυση των προβλημάτων ικανοποιήσιμοτητας προτασιακής λογικής, έστω και αν η πολυπλοκότητα του προγραμματισμού δράσεων σε γλώσσα STRIPS είναι PSPACE – hard. Κύριος στόχος της ήταν η δημιουργία καλύτερων ευρετικών συναρτήσεων αλλά και η εξαγωγή της λύση του προβλήματος με την χρήση ενός αλγορίθμου του GraphPlan.

Τον αλγόριθμό GraphPlan διαδέχθηκε ο SatPlan όπου ανταγωνιζόταν τη συγκεκριμένη για εκείνη την περίοδο τεχνολογία προβλημάτων προγραμματισμού δράσεων. Αργότερα την εμφάνιση της έκανε και η ευρετική συνάρτηση, μια μέθοδος η οποία φάνηκε να είναι κυρίαρχος στη λύση προβλημάτων προγραμματισμού δράσης μεταφρασμένα σε προβλήματα ικανοποιήσιμοτητας προτασιακής λογικής, αλλά εν τέλει την κυριαρχία την κράτησε ο SatPlan. Η βασική ανάπτυξη της θεωρίας αυτής ήταν η εισαγωγή των παράλληλων κωδικοποιήσεων. Αρκετές αναβαθμίσεις έγιναν στο μοντέλο αυτό όπου και τελικά κατέληξε στο εργαλείο SATMAXPLAN για επίλυση και βελτιστοποίηση των STRIPS προβλημάτων. Το εργαλείο του SATMAXPLAN εκτελεί περισσότερη διάχυση περιορισμών από της προηγούμενες εκδόσεις. Αυτό το πετυχαίνει με σύνολα από όρους που δεν περιέχουν κανένα πλεονασμό. Επίσης, σε συνδυασμό με ένα καινούργιο solver τον precosat μπορεί να επιλύσει

περισσότερα προβλήματα και παρουσιάζει μια αξιοσημείωτη πρόοδο στη ανάπτυξη της λύσης προβλημάτων προγραμματισμού δράσης μεταφρασμένα σε προβλήματα ικανοποιήσιμοτητας προτασιακής λογική. Επίσης στο εργαλείο SATMAXPLAN χρησιμοποιείται μια νέα κωδικοποίηση η Graphplan- direct, μια άμεση μετάφραση της δομής του γραφήματος δράσεων σε προτασιακή λογική.

Τα τελευταία χρόνια τα εργαλεία επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογική έχουν εξελιχτεί και εφαρμόζονται σε μεγάλο βαθμό τόσο στην Ηλεκτρονική Σχεδίαση όσο και στο τομέα Αυτοματισμού. Ωστόσο η μεγάλη εξέλιξη τους και η χρήση τους σε μεγάλο αριθμό πεδίων εφαρμογής, παρουσίασε την ανάγκη της χρήσης περιορισμών πέραν από την προτασιακή λογική του SAT..

Μια προσέγγιση ήταν το εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής να χρησιμοποιηθεί για την επίλυση ψευδοδυαδικών προβλημάτων μεταφράζοντας τα σε προτάσεις περιορισμών. Τα προβλήματα αυτά, είναι επίσης γνωστά και ως προβλήματα προγραμματισμού γραμμικών ακεραίων 0-1 (integer linear programming ILP). Από την σκοπιά του εργαλείου επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής οι ψευδοδυαδικοί περιορισμοί (PB-περιορισμοί) θεωρούνται σαν μια γενίκευση των προτάσεων περιορισμού. Αρκετά εργαλεία επίλυσης ψευδοδυαδικών περιορισμών αποτελούσαν επέκταση των εργαλείων επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής που τροποποιήθηκαν για να υποστηρίζουν τέτοιους περιορισμούς και στηρίζονται σε τεχνικές που προϋπήρχαν πριν από τις τεχνικές Chaff για επίλυση προβλημάτων ικανοποίησης προτασιακής λογικής. Τα εργαλεία αυτά συνεχίζουν να αποδίδουν σε αρκετό υψηλό επίπεδο.

Την εξέλιξη των αλγορίθμων που αφορούν την προτασιακή ικανοποιησιμότητα διαδέχθηκαν οι ψευδοδυαδικοί αλγόριθμοι και οι αλγόριθμοι βελτιστοποίησης τους, που έχουν σαν στόχο τη σημαντική βελτίωση των λύσεων. Η εξέλιξη στους τομείς αυτούς έφερε ως αποτέλεσμα την χρήση των αποδοτικότερων τεχνικών που χρησιμοποιούνται στη προτασιακή ικανοποιησιμότητα όπως μάθηση διαζευκτικών προτάσεων, τις δομές lazy data, τις μεθόδους διάγνωσης σύγκρουσης και την επέκταση τους σε ψευδοδυαδικούς αλγόριθμους βελτιστοποίησης.

Στους ψευδοδυαδικούς αλγόριθμους βελτιστοποίησης στόχος είναι η εύρεση μιας ανάθεσης τιμών η οποία θα ελαχιστοποιεί το κόστος της συνάρτησης αλλά ταυτοχρόνως θα ικανοποιεί όλους τους περιορισμούς. Ένα απλό ψευδοδυαδικό πρόβλημα όταν επεκταθεί μπορεί εύκολα να μετατραπεί σε ψευδοδυαδικό πρόβλημα βελτιστοποίησης με μια μικρή μετατροπή. Όταν βρεθεί η ανάθεση των τιμών που ικανοποιεί το πρόβλημα, ο αλγόριθμος να μην σταματάει αλλά να παίρνει την τιμή της συγκεκριμένης λύσης και να προσθέτει ένα καινούργιο περιορισμό έτσι ώστε η καινούργια λύση να αποτελεί βελτίωση της προηγούμενης.

Στην εργασία αυτή μελετήθηκε η δημιουργία της συνάρτησης βελτιστοποίησης για προβλήματα προγραμματισμού δράσης και μέσα από τα αποτελέσματα διαπιστώθηκε πως δαπανάται περισσότερος χρόνος για να λυθεί το πρόβλημα όταν γίνεται χρήση της συνάρτησης σε αντίθεση με το χρόνο εκτέλεσης όταν δεν χρησιμοποιείται. Επιπλέον, με την χρήση της συνάρτησης η βέλτιστη λύση στα ψευδοδυαδικά προβλήματα δεν βρισκόταν στα ίδια βήματα με την βέλτιστη λύση του εργαλείου SATMAXPLAN. Το εργαλείο MINISAT+ κατέληγε σε λύση αλλά δεν καταφέρνει να βρει την βέλτιστη λύση για όλους τους στόχους.

Εν συνεχεία, μελετήθηκε μια άλλη προσέγγιση για εύρεση της βέλτιστης λύσης. Σε αρκετά προβλήματα η βέλτιστη λύση βρισκόταν σε περισσότερα βήματα στο MINISAT+ από ότι στο SATMAXPLAN. Η μέθοδος αυτή σε κάθε επίπεδο πρόσθετε σαν unit clauses όσους στόχους βελτιστοποιούνταν. Επιπλέον, σε κάθε επίπεδο οι στόχοι που είχαν βελτιστοποιηθεί αφαιρούνταν με αποτέλεσμα ο αριθμός των στόχων σε κάθε βήμα να μειωνόταν, καθιστώντας έτσι πιο εύκολη και γρήγορη την αναζήτηση των εναπομεινάντων στόχων.

8.2 Μελλοντική Εργασία

Όπως έχει παρατηρηθεί στη τελευταία πειραματική μελέτη η βέλτιστη λύση για τα προβλήματα δεν βρισκόταν στο SAT και στο PB εργαλείο στα ίδια χρονικά βήματα. Στις πλείστες περιπτώσεις το PB χρειαζόταν ένα ή περισσότερα επιπλέον βήματα για να βρει την βέλτιστη λύση για όλους τους στόχους. Εάν και χρειαζόταν περισσότερα βήματα για να βρει την λύση ο χρόνος εύρεσης της, ήταν μικρότερος σε σχέση με το χρόνο εύρεσης της λύσης στο SAT.

Ξεκινώντας σε ένα χρονικό βήμα του προβλήματος, πριν βρεθεί η λύση, δημιουργήθηκε μια συνάρτηση βελτιστοποίησης όπου στόχο είχε να βρει την βέλτιστη λύση για τους στόχους την δεδομένη χρονική στιγμή. Όπως είναι λογικό δεν μπορούσαν όλοι οι στόχοι να ικανοποιηθούν, έτσι όσοι όροι ικανοποιούνταν, προσθέτονταν στην συνέχεια σαν unit clauses στο πρόβλημα και ενημερωνόταν η συνάρτηση με τους εναπομείναντες στόχους που δεν είχαν ικανοποιηθεί. Στην συνέχεια γινόταν προσπάθεια οι στόχοι αυτοί να επιλυθούν στο επόμενο χρονικό βήμα. Η διαδικασία αυτή επαναλαμβανόταν έως ότου καταλήξει στην βέλτιστη λύση, δηλαδή την εύρεση βέλτιστης λύσης για όλους τους στόχους.

Με αυτό το τρόπο θα μπορούσε να δημιουργηθεί ένας αλγόριθμός όπου σε κάθε στάδιο θα ενημερώνει ποιοι όροι της βέλτιστης συνάρτησης έχουν ικανοποιηθεί. Όσοι όροι ικανοποιούνται θα εισάγονται σαν επιπλέον περιορισμοί

στο πρόβλημα και θα αφαιρούνται από την βέλτιστη συνάρτηση. Οι περιορισμοί αυτοί θα είναι unit clauses. Με αυτό το τρόπο δεν θα γίνεται προσπάθεια σε επόμενο βήμα να ικανοποιηθούν. Στην συνάρτηση τώρα θα εισάγονται οι εναπομείναντες περιορισμοί που αντιστοιχούν στο επόμενο βήμα. Έτσι ο αλγόριθμός εύρεσης της βέλτιστης λύσης θα εστιάζεται στους όρους για τούς οποίου δεν έχει βρεθεί ακόμη η βέλτιστη λύση.

Μια παραλλαγή του πιο πάνω θα ήταν σε κάθε επίπεδο να μη προσπαθεί να βελτιώνει όλους τους στόχους. Έστω ένα πρόβλημα για βελτιστοποίηση. Μόλις ο αλγόριθμός φτάσει σε στόχο ο οποίος δεν βελτιστοποιείται στο παρών στάδιο θα σταματά και θα προχωράει στο επόμενο επίπεδο. Εάν έχει καταφέρει να βελτιστοποιήσει μερικούς στόχους πριν σταματήσει, τότε οι στόχοι αυτοί θα εισαχθούν σαν unit clauses στο αρχείο και θα συνεχιστεί η όλη διαδικασία μέχρις ότου βελτιστοποιηθούν όλοι οι στόχοι.

Μια άλλη μέθοδος όπου θα μπορούσε να βελτιστοποιήσει τη λύση των ψευδοδυαδικών προβλημάτων είναι η δημιουργία της βέλτιστης συνάρτησης με την χρήση περιορισμών πληθικότητας. Όπως έχει ήδη παρατηρηθεί οι περιορισμοί πληθικότητας είναι ουσιαστικά ένα ξεχωριστό είδος ψευδοδυαδικών περιορισμών που μπορούν να μεταφράζονται σε ισοδύναμούς συνδυασμούς από διαξεύξεις προτάσεων. Χωρίς όμως μια βοηθητική μεταβλητή το μέγεθός της μετάφρασης θα είναι εκθετικό. Μια σκέψη θα ήταν όλοι οι στόχοι να αναπαρασταθούν σαν ένας πληθικός περιορισμός στο πρόβλημα με βάση μια καινούργια μεταβλητή y. Η μεταβλητή y μπορεί στην συνέχεια να εισάγεται στην βέλτιστη συνάρτηση. Αυτό μπορεί να βοηθήσει γιατί η εκμάθηση των περιορισμών πληθικότητας έχει θεωρηθεί ως συμβιβασμός μεταξύ κανονική διαζευκτικής μορφής και ψευδοδυαδικούς μεθόδους μάθησης. Οι περιορισμοί είναι σε θέση να διατηρούν ορισμένες από τις πληροφορίες που αναπαρίστανται από τους ψευδοδυαδικούς περιορισμούς και ταυτόχρονα να οδηγεί σε μικρότερη διάχυση.

Βιβλιογραφία

1. Andreas Sideris, Yannis Dimopoulos, "Constraint Propagation in Propositional Planning",20th International Conference on Automated Planning and Scheduling, (ICAPS'10), "Toronto, Canada, 2010.

2. Avrim L. Blum , Merrick L. Furst, "Fast planning through planning graph analysis ",Artificial Intelligence 90 (1997) 28 l-300

3. Fadi A. Aloul, "Search techniques for SAT-based Boolean optimization" The Franklin Institute. Published by Elsevier Ltd, 2006.

4. Fadi A. Aloul,Karem A. Sakallah, "Solution and Optimization of Systems of Pseudo-Boolean Constraints",IEEE Transactions On Computers, vol. 56, no. 10 , October 2007.

5. 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

6. Henry Kautz ,Bart Selman,Joerg Hoffmann, "SatPlan: Planning as Satisfiability", Proceedings of the 10th European conference on Artificial intelligence, p.359-363, August 1992

7. Henry Kautz, "Deconstructing Planning as Satisfiability",Proceedings of the 21st National Conference on Artificial Intelligence. (AAAI-06), Boston, MA, 2006.

8. Jose Santos Vasco Manquinho , "Learning Techniques for Pseudo-Boolean Solving",7th International Workshop on Implementation of Logics, November 2008.

9. Michael D. Ernst, Todd D. Millstein, and Daniel S. Weld , "Automatic SAT-Compilation of Planning Problems", Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Aichi, Japan, August,1997, pp. 1169-1176.

10. Niklas Een,Niklas Sorensson, "Translating Pseudo-Boolean Constraints into SAT",Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 1-25.

11. Olivier Bailleux,Yacine Boufkhad,Olivier Roussel, "A Translation of Pseudo-Boolean Constraints to SAT",Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 191-200.

12. Olivier Roussel, Vasco M. Manquinho , "Pseudo Boolean and Cardinality Constraints",Handbook of Satisfiability Armin Biere,Marijn Heule,Hans van Maaren and Toby Walsh(Eds) IOS Press, 2009.

13. Richard E. Fikes, Ni Is .J. Ni lsson, "STRIPS: A new approach to the application of theorem proving to problem solving",Session No. 15, Heuristic Problem Solving.

14. Stuart Russel and Peter Norvig, Artificial Intelligence: A Modern Approach(Second Edition), Chapter 11 Planning, Prentice Hall, 2002.

15. Subbarao Kambhampati, Eric Parker and Eric Lambrecht "Understanding and Extending Graphplan" Proceedings of the 4th European Conference on Planning: Recent Advances in AI Planning Pages: 260 – 272, 1997

16. T. Bylander, "The Computational Complexity of Propositional STRIPS Planning" Artificial Intelligence, 69:165-204, 1994.

17. Vasco M. Manquinho, João P. Marques Silva, Arlindo L. Oliveira,Karem A. Sakallah, "Satisfiability-Based Algorithms for 0-1 Integer Programming",IEEE/ACM International Workshop on Logic Synthesis, June 1998.

18. Vasco M. Manquinho, Olivier Roussel "The First Evaluation of Pseudo-Boolean Solvers (PB'05) " Journal on Satisfiability, Boolean Modeling and Computation 2 ,103-143, 2006.

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

20. Vasco M. Manquinho,João P. Marques-Silva "Effective Lower Bounding Techniques for Pseudo-Boolean Optimization" Proceedings of the Design, Automation and Test in Europe Conference and Exhibition ,2005 linear optimization.

21. Vasco Manquinho and Joao Marques-Silva , " On Applying Cutting Planes in DLL-Based Algorithms for Pseudo-Boolean Optimization", Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT), June 2005.

22. Yannis Dimopoulos, Alfonso Gerevini, Patrik Haslum,Alessandro Saetti , "The Benchmark Domains of the Deterministic Part of IPC-5",Booklet of the 2006 Planning Competition, ICAPS'06, 2006.

Παράρτημα Α

```
Άνοιξε το αρχείο εισόδου για να το διαβάσεις.
Άνοιξε το αρχείο εξόδου για να γράψεις.

Εάν το αρχείο το οποίο διαβάζει είναι άδειο
{
    Τύπωσε μήνυμα λάθους.
}
Εάν δεν είναι άδειο τότε συνέχισε.
Διάβασε μια γραμμή του αρχείου. Αν η γραμμή που διάβασες δεν είναι κενή
συνέχισε.
{
Εάν η γραμμή η οποία έχεις διαβάσεις δεν έχει τους χαρακτήρες 'c' ή 'p
τότε επεξεργαζόμαστε την συγκεκριμένη γραμμή.
    {
            Εάν ο πρώτος χαρακτήρας της γραμματοσειράς που διάβαζες δεν
            ισούται με το 0,το κενό η το χαρακτήρα '\n' επεξεργαζόμαστε το
            χαρακτήρα.
        {
                Εάν ο πρώτος χαρακτήρας της γραμματοσειράς που
                διαβάζουμε δεν ισούται με το κενό και ισούται με το
                χαρακτήρα '-'
                {
           Μετρά το μήκος της γραμματοσειράς.
           Τύπωσε στο αρχείο "-1*x".

           Ενόσω η βοηθητική μεταβλητή δεν είναι όσο το μέγεθος
           του χαρακτήρα
               {
                       Τύπωσε ένα, ένα χαρακτήρα από τη
                   γραμματοσειρά στο αρχείο.
               }
           Τύπωσε στο αρχείο " ".
         Αύξησε τη μεταβλητή που μετράει τις αρνητικές μεταβλητές.

           }
           Αλλιώς εάν η γραμματοσειρά που διάβασες δεν ισούται
        το κενό ή το χαρακτήρα '\n'
           {
              Τύπωσε στο αρχείο "+1*x" και μετά την γραμματοσειρά.
              Αύξησε τη μεταβλητή που μετράει τις θετικές μεταβλητές.

         }
           Αύξησε τη μεταβλητή που μετράει τις όλες τις μεταβλητές της
```

```
γραμματοσειράς.
Κόψε την γραμμή μέχρι το επόμενο κενό " ".
Ενόσω η γραμματοσειρά δεν είναι κενή.
{
    Εάν ο πρώτος χαρακτήρας της γραμματοσειράς που
   διαβάζουμε δεν ισούται με το κενό και ισούται
   με το χαρακτήρα '-'
     {
            Μετρά το μήκος της γραμματοσειράς.
           Τύπωσε στο αρχείο "-1*x".

   Ενόσω η βοηθητική μεταβλητή δεν είναι όσο το
   μέγεθος του χαρακτήρα.
          {
     Τύπωσε ένα, ένα χαρακτήρα από τη
      Γραμματοσειρά στο αρχείο.
       }
        Τύπωσε στο αρχείο " ".
        Αύξησε τη μεταβλητή που μετράει τις αρνητικές μεταβλητές.

    }

                Αλλιώς εάν η γραμματοσειρά που διάβασες δεν
                ισούται το κενό ή το χαρακτήρα '0'.
    {
                Τύπωσε στο αρχείο "+1*x" και μετά την
                γραμματοσειρά.
                Αύξησε τη μεταβλητή που μετράει τις θετικές
                Μεταβλητές.

    }
    Εάν ο χαρακτήρας της γραμματοσειράς που
  Διαβάζουμε ισούται με το '0'
    Αύξησε τη μεταβλητή που μετράει τις όλες
                τις μεταβλητές της γραμματοσειράς.

    Κόψε την γραμμή μέχρι το επόμενο κενό " ".
}

Εάν η μεταβλητή που μετράει τις θετικές μεταβλητές
ισούται με τη μεταβλητή που μετράει όλες τις
          μεταβλητές της γραμματοσειράς.
{
    Τύπωσε στο αρχείο ">= 1;\n".
```

```
        }
        Αλλιώς
        {

             Τύπωσε στο αρχείο ">= %d;\n" οπού d ο
            μετρητής που μετράει τις αρνητικές μεταβλητές
           μειωμένο κατά 1 και με το αρνητικό πρόσημο.
        }
        Αρχικοποίηση των μεταβλητών που μετράνε τις
                    αρνητικές μεταβλητές τις θετικές καθώς και όλες τις
                    μεταβλητές της γραμματοσειράς με το 0.
    }

 }
```

Αλλιώς εάν ο πρώτος χαρακτήρας της γραμματοσειράς που διαβάζουμε ισούται με το χαρακτήρα 'p'.

{

Κόψε την γραμμή μέχρι το επόμενο κενό " ".

Αύξησε μια βοηθητική μεταβλητή.

```
    Εάν η γραμματοσειρά δεν είναι κενή.
        {
        Εάν η γραμματοσειρά δεν είναι κενή και εάν ο πρώτος
                    χαρακτήρας της γραμματοσειράς δεν ισούται με το ή το 'c'.
            {
            Εάν η βοηθητική μεταβλητή ισούται με 3.
                        {
                        Τύπωσε στο αρχείο "* #variable= %s " όπου s η
                        γραμματοσειρά.
            }
            Εάν η βοηθητική μεταβλητή ισούται με 4.
            {
                        Τύπωσε στο αρχείο "#constraint= %s όπου s η
                        γραμματοσειρά και τα ακόλουθα.
                        "**************************************\n"
                        "* begin normalizer comments\n"
                        "* category= SAT/UNSAT-SMALLINT\n"
                         "* end normalizer comments\n"
                            "**************************************\n"
            }
            }
            Κόψε την γραμμή μέχρι το επόμενο κενό " ".
            Αύξησε μια βοηθητική μεταβλητή.              }
```

```
        }
    }

    Κλείσε το αρχείο εξόδου.

}
```