Διπλωματική εργασία

Αποδοτικοί αλγόριθμοι για περιορισμούς κατεύθυνσης

από τον Βασίλη Καραδήμα
αρ.μητρ. 200407

Μεταπτυχιακό φοιτητή του ΠΜΣ
Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

$\mu \prod \lambda \forall$

Επιβλέπων Καθηγητής
Μανόλης Κουμπαράκης

ΑΘΗΝΑ 2007

στην μνήμη του πατέρα μου
Ευάγγελου

Ευχαριστίες

Καταρχήν ευχαριστώ τον επιβλέποντα καθηγητή μου Μανόλη Κουμπαράκη για την βοήθεια που μου προσέφερε και την αμέριστη συμπαράστασή του, ακόμα και στις πιο δύσκολες στιγμές. Θέλω επίσης να ευχαριστήσω τον Σπύρο Σκιαδόπουλο για τις πολύ χρήσιμες προτάσεις και παρατηρήσεις του. Χωρίς τη συνεισφορά των παραπάνω, θα ήταν αδύνατη η εκπόνηση αυτής της διπλωματικής εργασίας. Τέλος, ευχαριστώ όλους τους καθηγητές μου στο ΜΠΛΑ, για τις γνώσεις που με βοήθησαν να αποκτήσω καθ όλη τη διάρκεια της φοίτησής μου πρόγραμμα.

Περιεχόμενα

1. Εισαγωγή – Στόχοι και οργάνωση της εργασίας

1.1 Εισαγωγή

Ο ποιοτικός χωρικός λογισμός είναι ένα πεδίο που συγκεντρώνει διάφορα μοντέλα και γλώσσες που αναπαριστούν κάποιες έννοιες μεταξύ αντικειμένων του χώρου μέσα από μια καθαρά ποιοτική πλευρά. Ο ποιοτικός χωρικός λογισμός αποτελεί μια πολύ ενδιαφέρουσα ερευνητική περιοχή, η οποία μπορεί να βρει εφαρμογές σε διάφορους τομείς της επιστήμης των υπολογιστών, όπως είναι τα γεωγραφικά συστήματα πληροφοριών, διάφορα συστήματα πολυμέσων, καθώς επίσης και στον τομέα της τεχνητής νοημοσύνης. Μέσα σε αυτή την πολύ πλούσια περιοχή, καλούμαστε να αντιμετωπίσουμε διάφορα προβλήματα που ανακύπτουν στο εκάστοτε μοντέλο που χρησιμοποιούμε, σχεδιάζοντας και υλοποιώντας τους κατάλληλους αλγορίθμους.

Ο στόχος αυτής της διπλωματικής εργασίας είναι η παρουσίαση, υλοποίηση και σύγκριση δύο αλγορίθμων ελέγχου συνέπειας βασικών περιορισμών κατεύθυνσης. Πιο συγκεκριμένα, οι αλγόριθμοι που θα μελετηθούν είναι αυτοί των Σκιαδόπουλου-Κουμπαράκη [SK05] και των Navarrete-Morales-Sciavicco [NMS07]. Επειδή οι αλγόριθμοι αυτοί είναι αρκετά εκτενείς, η παρουσίασή τους θα γίνει τμηματικά, αφού άλλωστε και οι ίδιοι έχουν σχεδιαστεί έτσι, ώστε να γίνει πλήρως κατανοητός ο τρόπος με τον οποίο δουλεύουν.

Θεωρούμε αρκετά σημαντική τη συνεισφορά της εργασίας για τους εξής λόγους:

- Μέσω του τρόπου παρουσίασης των δύο αλγόριθμων, πιστεύουμε ότι διευκολύνεται σημαντικά η κατανόηση του τρόπου λειτουργίας τους, καθώς επίσης και ότι γίνονται ξεκάθαρες οι ομοιότητες και οι διαφορές που υπάρχουν μεταξύ τους.
- Μέσω της υλοποίησης που δημιουργήσαμε γίνεται πλέον εφικτή η άμεση πειραματική σύγκριση των δύο αλγόριθμων. Από όσο γνωρίζουμε, δεν προϋπήρχε κάποιο πρόγραμμα που να υλοποιεί αποδοτικά τους δύο αυτούς αλγόριθμους, καθώς επίσης και τις σχετικές με αυτούς λειτουργίες.

1.2 Οργάνωση της εργασίας

Σε αυτή την ενότητα θα περιγράψουμε την οργάνωση της εργασίας. Μετά από αυτό το εισαγωγικό πρώτο κεφάλαιο, θα παρουσιάσουμε αναλυτικά στο κεφάλαιο 2 το μοντέλο των Goyal και Egenhofer [GE00] που χρησιμοποιούμε. Το μοντέλο αυτό είναι από τα πλέον διαδεδομένα μοντέλα που χρησιμοποιούνται στον ποιοτικό χωρικό λογισμό. Στη συνέχεια, θα παρουσιάσουμε στο κεφάλαιο 3 τον αλγόριθμο των Σκιαδόπουλου-Κουμπαράκη. Λόγω του μεγέθους του αλγόριθμου, όπως αναφέραμε και πιο πριν, η παρουσίαση θα γίνει τμηματικά. Σε κάθε τμήμα του αλγορίθμου, εκτός του ψευδοκώδικα θα υπάρχει και αναλυτική περιγραφή του, η οποία επιπλέον θα συνοδεύεται από παραδείγματα, όπου κρίνουμε ότι αυτά βοηθούν στην κατανόησή του. Στο κεφάλαιο 4 θα γίνει παρουσίαση του αλγορίθμου των Navarrete-Morales-Sciavicco. Αυτός ο αλγόριθμος ουσιαστικά αποτελεί μια

παραλλαγή-βελτίωση του προηγούμενου, εκμεταλλευόμενος το γνωστό θεώρημα του Helly από την συνδυαστική γεωμετρία. Ακολουθεί στο κεφάλαιο 5 η περιγραφή της υλοποίησης που δημιουργήσαμε, ώστε να είναι δυνατή η σύγκριση των δύο αλγορίθμων και σε πρακτικό επίπεδο πέραν του θεωρητικού, αφού άλλωστε θα γίνει και η αναμενόμενη ανάλυση πολυπλοκότητας. Το λογισμικό που δημιουργήσαμε δέχεται αρκετές παραμέτρους και εκτός από την σύγκριση των δύο αλγορίθμων που επιτελεί για ερευνητικούς σκοπούς, θεωρούμε ότι είναι αρκετά υψηλής ποιότητας (ή τουλάχιστον έτσι θέλουμε να πιστεύουμε) ώστε να μπορεί να χρησιμοποιηθεί και μεμονωμένα για την επίλυση πραγματικών προβλημάτων. Το λογισμικό αυτό είναι ανοιχτού κώδικα και διατίθεται ελεύθερα σε οποιονδήποτε ενδιαφέρεται, υπό την άδεια GNU GPL (General Public License). Στο κεφάλαιο 6 που ακολουθεί, παρουσιάζουμε τις μετρήσεις που έγιναν και συγκρίνουμε τους δύο αλγορίθμους ως προς την ταχύτητά τους, για τα διάφορα είδη συνόλων περιορισμών που δέχονται σαν είσοδο. Στο κεφάλαιο 7 θα αναφέρουμε τα συμπεράσματα που προέκυψαν από την μελέτη που έγινε. Τέλος, θα αναφέρουμε την βιβλιογραφία και γενικά τις πηγές που χρησιμοποιήσαμε, αφού δίχως τούτες θα ήταν αδύνατη η εκπόνηση αυτής της διπλωματικής εργασίας.

2. Το μοντέλο των περιορισμών κατεύθυνσης

2.1 Κλάσεις χωρικών σχέσεων

Όπως αναφέραμε παραπάνω, ο ποιοτικός χωρικός λογισμός είναι μια σημαντική περιοχή έρευνας που μπορεί να χρησιμοποιηθεί στους τομείς της τεχνητής νοημοσύνης, των γεωγραφικών συστημάτων πληροφοριών, των βάσεων δεδομένων, καθώς επίσης και σε διάφορα συστήματα πολυμέσων. Εύκολα λοιπόν μπορεί να αντιληφθεί κανείς, ότι όλοι αυτοί οι τομείς δεν μπορούν να καλυφθούν από ένα μόνο μοντέλο ή ένα μόνο είδος σχέσεων. Συνεπώς θα πρέπει να γίνει μια κατηγοριοποίηση ανάλογα με το είδος των σχέσεων που έχουν οριστεί για τα διάφορα μοντέλα.

Οι ερευνητές έχουν ορίσει τρεις κλάσεις χωρικών σχέσεων:

- Τοπολογικές σχέσεις
- Σχέσεις απόστασης
- Σχέσεις κατεύθυνσης

Ας εξετάσουμε λοιπόν περισσότερο αναλυτικά τις κλάσεις αυτών των σχέσεων.

2.1.1 Τοπολογικές σχέσεις

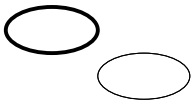
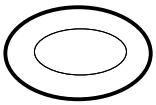
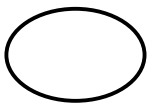
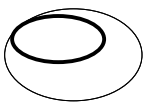
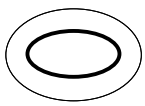
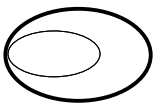
Οι τοπολογικές σχέσεις περιγράφουν πως σχετίζονται τα σύνορα και οι θέσεις των εσωτερικών και εξωτερικών σημείων δύο περιοχών. Για παράδειγμα, αν θεωρήσουμε ότι A και B είναι δύο περιοχές του χώρου, τότε μπορούμε να ορίσουμε τον περιορισμό 'A περιέχει B' που μεταφράζεται ως εξής. Το σύνολο των εσωτερικών σημείων του A είναι υπερσύνολο των σημείων του B. Ένα πολύ ενδιαφέρον στοιχείο είναι ότι οι τοπολογικές σχέσεις διατηρούνται κάτω από διάφορους τοπολογικούς μετασχηματισμούς, όπως είναι η μεταφορά και η περιστροφή. Ωστόσο τέτοιοι μετασχηματισμοί δεν διατηρούν εν γένει τις σχέσεις απόστασης και κατεύθυνσης που υπάρχουν μεταξύ των δύο περιοχών. Η τοπολογική πληροφορία είναι μια καθαρά ποιοτική πληροφορία και δεν περιέχει καθόλου ποσοτικά στοιχεία. Για παράδειγμα, δύο περιοχές είναι γειτονικές αν μοιράζονται κάποιο τμήμα των συνόρων τους, αλλά αυτή η σχέση γειτονίας δεν εξαρτάται από το μήκος του συνόρου που μοιράζονται. Τα χωρικά αντικείμενα μπορούν να αναπαρασταθούν από διάφορα πρωτογενή στοιχεία, όπως είναι σημεία, γραμμές και περιοχές. Αυτά τα στοιχεία αναφέρονται ως κελιά και ορίζονται για τις διάφορες διαστάσεις του χώρου.

Όπως αναφέραμε και πριν, μια διμελής τοπολογική σχέση R μεταξύ δύο κελιών A και B, βασίζεται στη σύγκριση των συνόρων, των εσωτερικών και των εξωτερικών σημείων των δύο κελιών. Ας συμβολίσουμε το εσωτερικό του κελιού A με A^o, το σύνορο με ∂A και το εξωτερικό με A^- (αντίστοιχα και για το κελί B).

Συνεπώς μια τοπολογική σχέση R ορίζεται ως μια μήτρα 9x9, όπου τα στοιχεία αυτής της μήτρας είναι η τομή κάθε δυνατού συνδυασμού μέρους (σύνορο, εσωτερικό, εξωτερικό) του κελιού A με μέρος του κελιού B. Δηλαδή είναι:

$$R(A,B)=\begin{pmatrix} A^o\cap B^o & A^o\cap \partial B & A^o\cap B^- \\ \partial A\cap B^o & \partial A\cap \partial B & \partial A\cap B^- \\ A^-\cap B^o & A^-\cap \partial B & A^-\cap B^- \end{pmatrix}$$

Άρα προκύπτουν διάφοροι συνδυασμοί, ανάλογα με το εάν κάποιο στοιχείο αυτού του πίνακα είναι το κενό σύνολο ή όχι. Επομένως αφού κάθε στοιχείο του πίνακα μπορεί να είναι ή να μην είναι το κενό, έχουμε συνολικά $2^9=512$ πιθανές τοπολογικές σχέσεις μεταξύ δύο οποιωνδήποτε κελιών. Είναι προφανές από τον ορισμό τους, ότι αυτές οι τοπολογικές σχέσεις είναι όλες οι δυνατές που υπάρχουν και είναι ξένες μεταξύ τους. Στον δισδιάστατο χώρο, μόνο 8 μπορούν να γίνουν αντιληπτές και συγκεκριμένα αυτές που φαίνονται στο σχήμα 1:

disjoint	meet	overlap	contains
			
$\begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$
equals	coveredBy	inside	covers
			
$\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$	$\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$

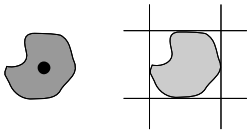
Σχήμα 1: Τοπολογικές σχέσεις

2.1.2 Σχέσεις απόστασης

Οι σχέσεις απόστασης περιγράφουν ποιοτικά την απόσταση μεταξύ δύο περιοχών, βασιζόμενες σε υπάρχοντα αριθμητικά δεδομένα. Κατά καιρούς έχουν προταθεί διάφορα μοντέλα και δεν υπάρχει μια ενοποιημένη σημασιολογία. Η ύπαρξη τέτοιων σχέσεων οφείλεται στο γεγονός ότι είναι επιθυμητή η ποιοτική γνώση για την απόσταση μεταξύ δύο περιοχών (π.χ. 'Α κοντά Β'). Μια εφαρμογή που μπορούμε να σκεφτούμε είναι ότι σε ένα γεωγραφικό σύστημα πληροφοριών, θα θέλαμε να έχουμε τη δυνατότητα να κάνουμε μια ερώτηση της μορφής 'Ποια είναι τα νοσοκομεία που είναι πιο κοντά από το σημείο που βρισκόμαστε;' όπου λέξη κοντά μπορεί να έχει την έννοια ότι η οδική χιλιομετρική απόσταση μεταξύ των δύο σημείων είναι μικρότερη του ενός χιλιομέτρου.

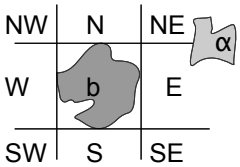
2.1.3 Σχέσεις κατεύθυνσης

Οι σχέσεις κατεύθυνσης περιγράφουν την σχετική κατεύθυνση μεταξύ δύο περιοχών. Για παράδειγμα, αν θεωρήσουμε ότι Α και Β είναι δύο περιοχές του χώρου, τότε μπορούμε να ορίσουμε τον περιορισμό 'Α νοτιοδυτικά Β'. Για τις σχέσεις κατεύθυνσης έχουν προταθεί διάφορα μοντέλα, στα οποία μια περιοχή αναπαριστάται είτε με ένα σημείο (π.χ. το κέντρο βάρους), είτε με κάποιο πλαίσιο (συνήθως το ελάχιστο πλαίσιο που την περικλείει, το οποίο από εδώ και στο εξής θα αναφέρουμε για λόγους συντομογραφίας ως mbb – Minimum Bounding Box). Οι δύο αυτές προσεγγίσεις απεικονίζονται στο σχήμα 2 που ακολουθεί:



Σχήμα 2: Προσεγγίσεις αναπαράστασης περιοχής

Το μοντέλο των Goyal και Egenhofer [GE00] (το οποίο στη συνέχεια επεκτάθηκε από τον Σκιαδόπουλο και τον Κουμπαράκη [SK05]), είναι το μοντέλο με το οποίο ασχολούμαστε σε αυτή τη διπλωματική εργασία και ακολουθεί τη δεύτερη προσέγγιση. Συγκεκριμένα σε αυτό το μοντέλο ο χώρος χωρίζεται σε εννιά τμήματα. Το mbb της περιοχής ονομάζεται Β, ενώ τα υπόλοιπα τμήματα ονομάζονται με τα αρχικά της αντίστοιχης κατεύθυνσης (π.χ. SW για νοτιοδυτικά). Αυτός ο διαχωρισμός φαίνεται στο σχήμα 3:



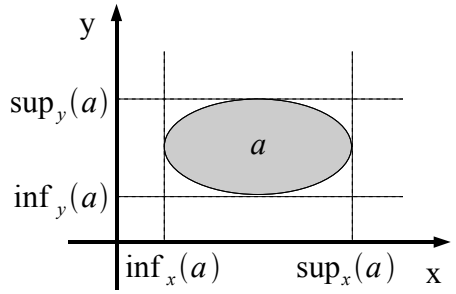
Σχήμα 3: Διαχωρισμός του χώρου σε τμήματα

Η περιοχή που περικλείεται από το mbb (στο σχήμα η b) λέγεται περιοχή αναφοράς, ενώ η περιοχή που σχετίζεται με αυτή (στο σχήμα η α) λέγεται πρωτεύουσα περιοχή. Το αρχικό μοντέλο των Goyal και Egenhofer [GE00] θεωρούσε μόνο συνεκτικές περιοχές, ωστόσο η επέκταση αυτού του μοντέλου που θα παρουσιάσουμε στην συνέχεια, επιτρέπει την ύπαρξη και μη συνεκτικών περιοχών (αποτελούμενες από υποπεριοχές που μπορεί να συνδέονται στα σύνορά τους, να έχουν οπές κ.ο.κ.). Είναι λοιπόν προφανές ότι αυτό το μοντέλο έχει σαφώς μεγαλύτερη εκφραστική δύναμη από το αρχικό. Για παράδειγμα, σε ένα γεωγραφικό σύστημα πληροφοριών, το Ηνωμένο Βασίλειο θα πρέπει να αποτελείται από δύο περιοχές, την Αγγλία και την Βόρεια Ιρλανδία.

2.2 Τυπικός ορισμός του μοντέλου των σχέσεων κατεύθυνσης

Θεωρούμε ότι βρισκόμαστε στον Ευκλείδιο χώρο $\Re^2$. Οι περιοχές μέσα σε αυτό το χώρο ορίζονται ως μη κενά και φραγμένα σύνολα σημείων του $\Re^2$. Έστω ότι a είναι μια τυχαία περιοχή του $\Re^2$. Όπως φαίνεται στο παρακάτω σχήμα, παίρνοντας τις προβολές επί των αξόνων x και y, ορίζουμε το ελάχιστο πλαίσιο που την περικλείει, ως το σύνολο:

$$mbb(a)=\{(x,y) \mid \inf_x(a)\le x\le \sup_x(a),\ \ \inf_y(a)\le y\le \sup_y(a)\}$$



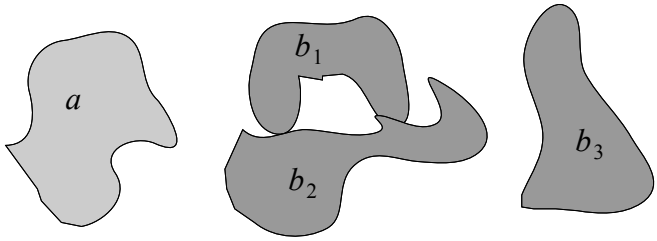
Σχήμα 4: Ελάχιστο πλαίσιο που περικλείει μια περιοχή

Το $mbb(a)$ λέγεται μη τετριμμένο αν και μόνο αν δεν είναι σημείο ή ευθύγραμμο τμήμα του χώρου, δηλαδή ανν ισχύει $\inf_x(a)<\sup_x(a)$ και $\inf_y(a)<\sup_y(a)$.

Σε αυτή την μελέτη μας ενδιαφέρουν δύο κατηγορίες περιοχών.

- Περιοχές που είναι ομομορφικές ως προς τον κλειστό μοναδιαίο κυκλικό δίσκο (δηλ. το σύνολο $\{(x,y) \mid x^2+y^2\le 1\}$). Η κλάση αυτών των περιοχών ονομάζεται REG. Αυτή η κλάση περιλαμβάνει όλες τις συνεκτικές περιοχές, ενώ δεν περιλαμβάνει μη συνεκτικές περιοχές, περιοχές με οπές, μεμονωμένα σημεία, μεμονωμένες γραμμές ή συνδυασμούς των παραπάνω.

- Περιοχές που είναι πεπερασμένες ενώσεις στοιχείων της REG. Αυτή η κλάση ονομάζεται REG^* και προφανώς αποτελεί μια φυσική επέκταση της προηγούμενης κλάσης, στην περίπτωση όπου είναι επιθυμητή η συμπερίληψη και μη συνεκτικών περιοχών του χώρου (όπως στο προηγούμενο παράδειγμα με το Ηνωμένο Βασίλειο). Στο σχήμα 5 που ακολουθεί βλέπουμε ότι οι περιοχές a, b_1, b_2, b_3 ανήκουν στην REG (και στην REG^*) και η περιοχή $b = b_1 \cup b_2 \cup b_3$ ανήκει στην REG^*.



Σχήμα 5: Παραδείγματα περιοχών

Έστω τώρα a, b δύο περιοχές στην REG^*. Ορίζουμε τις διμελείς σχέσεις $B, S, SW, W, NW, N, NE, E, SE$, τις οποίες λέμε **ατομικές**, ως εξής:

$$
\begin{aligned}
a\,B\,b &\Leftrightarrow \inf_x(b) \le \inf_x(a), \sup_x(a) \le \sup_x(b), \inf_y(b) \le \inf_y(a), \sup_y(a) \le \sup_y(b)\\
a\,S\,b &\Leftrightarrow \inf_x(b) \le \inf_x(a), \sup_x(a) \le \sup_x(b), \sup_y(a) \le \inf_y(b)\\
a\,SW\,b &\Leftrightarrow \sup_x(a) \le \inf_x(b), \sup_y(a) \le \inf_y(b)\\
a\,W\,b &\Leftrightarrow \sup_x(a) \le \inf_x(b), \inf_y(b) \le \inf_y(a), \sup_y(a) \le \sup_y(b)\\
a\,NW\,b &\Leftrightarrow \sup_x(a) \le \inf_x(b), \sup_y(b) \le \inf_y(a)\\
a\,N\,b &\Leftrightarrow \sup_x(a) \le \sup_x(b), \sup_y(b) \le \inf_y(a), \inf_x(b) \le \inf_x(a)\\
a\,NE\,b &\Leftrightarrow \sup_x(b) \le \inf_x(a), \sup_y(b) \le \inf_y(a)\\
a\,E\,b &\Leftrightarrow \sup_x(b) \le \inf_x(a), \inf_y(b) \le \inf_y(a), \sup_y(a) \le \sup_y(b)\\
a\,SE\,b &\Leftrightarrow \sup_y(a) \le \inf_y(b), \sup_x(b) \le \inf_x(a)
\end{aligned}
$$

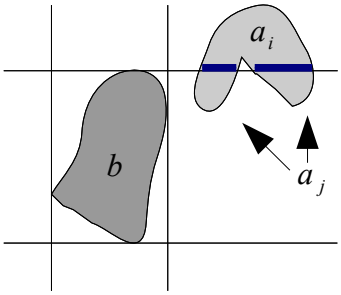
Για κάθε $k = 1,..,9$ ορίζουμε τη διμελή σχέση $R_1 : .. : R_k$, για την οποία έχουμε ότι $R_1, .., R_k \in \{B, S, SW, W, NW, N, NE, E, SE\}$ και $R_i \neq R_j, \forall\, i, j = 1,..,9$ με $i \neq j$, ως εξής:

$$a\,R_1 : .. : R_k\,b \Leftrightarrow \exists a_1, .., a_k \in REG^* (a = a_1 \cup .. \cup a_k \ \wedge\ a_1\,R_1\,b \ \wedge\ .. \ \wedge\ a_k\,R_k\,b)$$

Οι παραπάνω σχέσεις που ορίσαμε, ονομάζονται **βασικές σχέσεις κατεύθυνσης**. Εξ ορισμού είναι προφανές ότι οι ατομικές σχέσεις ανήκουν στο σύνολο των βασικών σχέσεων κατεύθυνσης (για $k = 1$). Οι μεταβλητές $a_1, .., a_k$

που εμφανίζονται στον παραπάνω ορισμό λέγονται συνιστώσες μεταβλητές της μεταβλητής a .

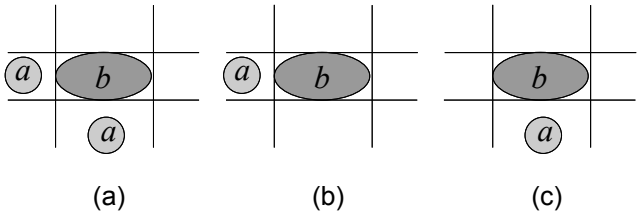
Παρατηρούμε ότι δύο οποιεσδήποτε συνιστώσες μεταβλητές a_i, a_j αν και δεν έχουν κοινά εσωτερικά σημεία, μπορεί να έχουν κοινά σημεία στα σύνορά τους. Αυτό είναι άμεση συνέπεια του ορισμού των ατομικών σχέσεων, αφού ορίζονται με ασθενείς ανισότητες και συνεπώς ο Ευκλείδιος χώρος δεν διαμερίζεται, αφού τα γειτονικά τμήματα που δημιουργούνται μοιράζονται τα οριακά σημεία. Τούτο φαίνεται εύκολα στο σχήμα 6:



Σχήμα 6: Κοινά σημεία μεταξύ γειτονικών περιοχών

Το σύνολο των βασικών σχέσεων κατεύθυνσης περιέχει $2^9-1=511$ στοιχεία (δηλαδή όσο το μέγεθος του δυναμοσυνόλου των ατομικών σχέσεων, αφαιρώντας από αυτό το κενό σύνολο). Το σύνολο αυτό το συμβολίζουμε με D^* . Στο αρχικό μοντέλο των Goyal και Egenhofer [GE00] υπήρχαν μόλις 218 σχέσεις (όπου το σύνολο των σχέσεων λέγεται D), αφού δεν επιτρέπονταν μη συνεκτικές περιοχές, οπότε έπρεπε να απαλειφθούν οι αντίστοιχες σχέσεις (π.χ. $S{:}W$).

Αφού λοιπόν το D^* περιέχει αυτές τις 511 σχέσεις, μπορούμε να πάρουμε το δυναμοσύνολό του 2^{D^*}, που περιέχει $2^{511}-1$ σχέσεις (όπως πριν, αφαιρούμε το κενό σύνολο). Τα στοιχεία λοιπόν αυτού του δυναμοσυνόλου λέγονται **σχέσεις κατεύθυνσης** και μπορούν να χρησιμοποιηθούν για την αναπαράσταση τόσο **ορισμένης** όσο και **αόριστης** πληροφορίας. Για παράδειγμα, ας θεωρήσουμε τον περιορισμό $a\{S, W\}b$ που μεταφράζεται ως ' η περιοχή a βρίσκεται νότια της περιοχής b **ή** η περιοχή a βρίσκεται δυτικά της περιοχής b '. Μια παρατήρηση που πρέπει να κάνουμε είναι ότι το **ή** λειτουργεί σαν αποκλειστικό **ή** αφού οι βασικές σχέσεις είναι ξένες μεταξύ τους. Επίσης θα πρέπει να προσέξουμε τη διαφορά της σχέσης $\{S, W\}$ με τη σχέση $S{:}W$. Ο περιορισμός $a\,S{:}W\,b$ μεταφράζεται ως ' η περιοχή a βρίσκεται νότια **και** δυτικά της περιοχής b '. Δηλαδή η αοριστία της πληροφορίας για την πρώτη περίπτωση, έγκειται στο ότι γνωρίζουμε ότι ισχύει **μία** μόνο περίπτωση από τις δύο (στο σχήμα που ακολουθεί είναι οι περιπτώσεις (b) και (c)), αλλά δεν ξέρουμε ποια. Για την δεύτερη περίπτωση η πληροφορία είναι πλήρως καθορισμένη (περίπτωση (a)). Το σχήμα 7 που ακολουθεί αποσαφηνίζει πλήρως τη διαφορά μεταξύ αυτών των περιπτώσεων.



Σχήμα 7: Διαφορές μεταξύ των σχέσεων

Έχοντας λοιπόν ορίσει τις σχέσεις κατεύθυνσης, μπορούμε εύκολα να ορίσουμε και τους αντιστοίχους περιορισμούς. Ένας **περιορισμός κατεύθυνσης** είναι ένας τύπος της μορφής $a\,R\,b$, όπου τα a, b είναι μεταβλητές που παίρνουν τιμές από το REG^* και R είναι μια σχέση κατεύθυνσης. Αν η R είναι **βασική** (αντίστοιχα **ατομική**) σχέση κατεύθυνσης τότε ο περιορισμός λέγεται **βασικός** (αντίστοιχα **ατομικός**) περιορισμός κατεύθυνσης.

Μερικά παραδείγματα περιορισμών κατεύθυνσης είναι τα εξής:

$$a\,N\,b,\ a\,SW{:}W{:}NW\,b,\ a\{B,\ S{:}SE,\ E\}\,b$$

Ο πρώτος περιορισμός κατεύθυνσης είναι ατομικός (και βασικός), ενώ ο δεύτερος είναι βασικός και μη ατομικός. Ο τρίτος περιορισμός δεν είναι βασικός.

Αφού ορίσαμε και τυπικά το μοντέλο των περιορισμών κατεύθυνσης θα δούμε στη συνέχεια κάποια θεμελιώδη προβλήματα που καλούμαστε να αντιμετωπίσουμε μέσα σε αυτό το πλαίσιο του λογισμού κατευθύνσεων.

2.3 Θεμελιώδη προβλήματα λογισμού κατευθύνσεων

2.3.1 Πρόβλημα συνέπειας

Το πλέον σημαντικό πρόβλημα που καλούμαστε να αντιμετωπίσουμε στον χωρικό λογισμό κατευθύνσεων και με το οποίο θα ασχοληθούμε εκτενώς σε αυτή τη διπλωματική εργασία, είναι το πρόβλημα της συνέπειας. Δοθέντος ενός τυχαίου συνόλου από περιορισμούς κατεύθυνσης, θα πρέπει να αποφασίσουμε αν αυτό το σύνολο είναι συνεπές, δηλαδή αν υπάρχει μια **ανάθεση τιμών** (στην συγκεκριμένη περίπτωση περιοχών του χώρου) στις μεταβλητές που εμφανίζονται σε αυτούς τους περιορισμούς, που να τους ικανοποιεί όλους. Αν υπάρχει μια τέτοια ανάθεση, τότε αυτή η ανάθεση λέγεται **λύση** και το σύνολο των περιορισμών λέγεται **συνεπές**, διαφορετικά λέγεται **ασυνεπές**. Εκτός από το πρόβλημα της συνέπειας σαν πρόβλημα απόφασης αυτό καθεαυτό, πολλές φορές είναι επιθυμητή η εύρεση μιας λύσης. Για την ειδική περίπτωση που οι περιορισμοί κατεύθυνσης είναι βασικοί, έχουν προταθεί αποδοτικοί αλγόριθμοι (πολυωνυμικού χρόνου) από τους Σκιαδόπουλο-Κουμπαράκη και τους Navarrete-Morales-Sciavicco, οι οποίοι θα παρουσιαστούν αναλυτικά στη συνέχεια. Για την γενική περίπτωση που οι

περιορισμοί κατεύθυνσης δεν είναι όλοι κατ' ανάγκη βασικοί αλλά υπάρχουν και μη βασικοί, το πρόβλημα δυστυχώς αποδεικνύεται ότι είναι NP-πλήρες, σύμφωνα με την απόδειξή του Σκιαδόπουλου στο [SK05].

2.3.2 Πρόβλημα αντιστροφής

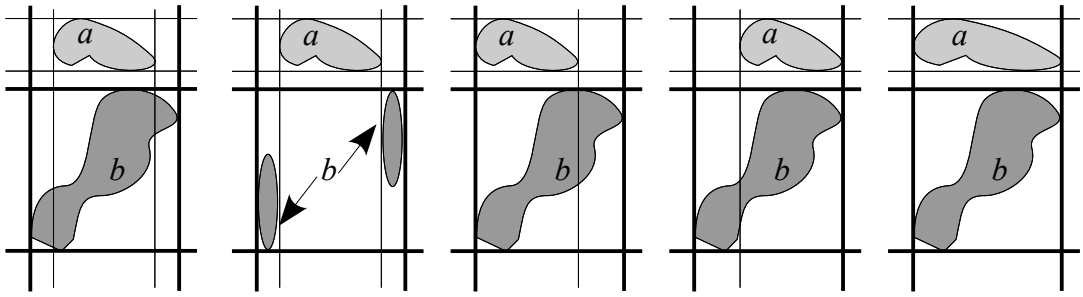
Σύμφωνα με τον Allen [ALLEN83] και άλλους ερευνητές, μια βασική πράξη που πρέπει να ορίσουμε όταν έχουμε μια temporal ή spatial άλγεβρα, είναι η πράξη της αντιστροφής. Στη περίπτωση του λογισμού κατευθύνσεων, η πράξη αυτή ορίζεται ως εξής:

Έστω R μια σχέση κατεύθυνσης και έστω a, b δύο τυχαίες περιοχές στην κλάση REG^*. Η **αντίστροφη** σχέση της R, που συμβολίζουμε με $inv(R)$, είναι η σχέση κατεύθυνσης που ικανοποιεί τη συνθήκη:

$$a\ inv(R)\ b \ \Leftrightarrow\ b\ R\ a$$

Έστω ότι έχουμε δύο περιοχές a, b και ας υποθέσουμε ότι $a R b$, όπου R είναι μια βασική σχέση κατεύθυνσης. Τότε η αντίστροφη σχέση $inv(R)$ δεν είναι εν γένει και αυτή βασική. Για παράδειγμα, αν ισχύει $a N b$ τότε έχουμε (όπως φαίνεται και στο σχήμα 8 που ακολουθεί):

$$inv(N) = \{S{:}SW{:}SE,\ SW{:}SE,\ S{:}SE,\ S{:}SW,\ S\}$$



Σχήμα 8: Αντίστροφες σχέσεις της N

Από τα παραπάνω προκύπτει ότι η σχετική θέση δύο περιοχών a, b του χώρου μπορεί να καθοριστεί πλήρως, μόνο όταν προσδιορίσουμε την κατεύθυνση της a ως προς την b και η κατεύθυνση της b ως προς την a. Θα πρέπει λοιπόν να σχεδιαστούν αποδοτικοί αλγόριθμοι, οι οποίοι να υπολογίζουν γρήγορα την αντίστροφη σχέση. Από τον ορισμό της αντίστροφης σχέσης μπορούμε εύκολα να διαπιστώσουμε ότι το πρόβλημα της αντιστροφής ανάγεται άμεσα στο πρόβλημα της συνέπειας. Μπορούμε λοιπόν να χρησιμοποιήσουμε τον αλγόριθμο συνέπειας Σκιαδόπουλου-Κουμπαράκη για να υπολογίσουμε την αντίστροφη σχέση μιας βασικής σχέσης κατεύθυνσης με τον εξής τρόπο. Καταρχήν, θα πρέπει να

παρατηρήσουμε ότι για κάθε βασικό περιορισμό κατεύθυνσης $a R_1 b$, μια βασική σχέση κατεύθυνσης R_2 ικανοποιεί τον περιορισμό $b R_2 a$ (και συνεπώς ανήκει στην αντίστροφη σχέση) ανν το σύνολο των περιορισμών $\{a R_1 b, b R_2 a\}$ είναι συνεπές. Επομένως, αν εξετάσουμε διαδοχικά όλα αυτά τα σύνολα των περιορισμών (που είναι προφανώς 511 το πλήθος), διαπιστώνουμε ποιες βασικές σχέσεις ανήκουν στην αντίστροφη σχέση. Η ίδια προσέγγιση μπορεί να χρησιμοποιηθεί και για το πρόβλημα της σύνθεσης που ακολουθεί στη συνέχεια.

2.3.3 Πρόβλημα σύνθεσης

Εκτός από την πράξη της αντιστροφής, μια επίσης πολύ σημαντική πράξη που πρέπει να ορίζεται σε μια temporal ή spatial άλγεβρα είναι η πράξη της **σύνθεσης**. Προκειμένου να επιλυθεί το πρόβλημα της συνέπειας, πολλές φορές γίνεται χρήση διαφόρων μεθόδων διάδοσης περιορισμών (π.χ. συνέπεια μονοπατιού). Η διάδοση των περιορισμών χρησιμοποιεί κατά κόρον την πράξη της σύνθεσης, οπότε εύκολα καταλαβαίνουμε πόσο σημαντική είναι η ύπαρξη αποδοτικών αλγορίθμων που να υπολογίζουν γρήγορα τη σύνθεση δύο σχέσεων. Στο μοντέλο του χωρικού λογισμού κατευθύνσεων, έχουν οριστεί δύο μορφές σύνθεσης.

Η πρώτη μορφή σύνθεσης ονομάζεται **υπαρξιακή σύνθεση** και δεν είναι άλλη από τη γνωστή σύνθεση σχέσεων όπως έχει οριστεί στη θεωρία συνόλων. Δηλαδή αν R_1, R_2 είναι δύο σχέσεις κατεύθυνσης, τότε η σύνθεσή τους $R_1 \circ R_2$ είναι η σχέση:

$$R_1 \circ R_2 = \{(a,b) \mid \exists c ((a,c) \in R_1 \ \wedge \ (c,b) \in R_2)\}$$

Σύμφωνα με τις παρατηρήσεις των Renz και Ligozat [RL05], η χρήση της υπαρξιακής σύνθεσης σε κάποιες περιπτώσεις είναι προβληματική. Καταρχήν θα πρέπει να σημειώσουμε ότι η σύνθεση αρκεί να υπολογιστεί μόνο για ζεύγη ατομικών σχέσεων, αφού όπως έχει δείξει ο Allen [ALLEN83], η σύνθεση μη ατομικών σχέσεων ισούται με την ένωση των συνθέσεων μεταξύ των ατομικών σχέσεων που περιλαμβάνονται μέσα σε εκείνες (δηλ. τις μη ατομικές σχέσεις). Ωστόσο, σύμφωνα με τον ορισμό της σύνθεσης θα έπρεπε να εξετάσουμε άπειρο αριθμό πλειάδων για να υπολογίσουμε τη σύνθεση ατομικών σχέσεων, κάτι που προφανώς είναι ανέφικτο. Παρόλα αυτά, σε κάποιες περιπτώσεις όπου το σύμπαν των μεταβλητών είναι διατεταγμένο ή γενικά καλά δομημένο (π.χ. point άλγεβρα ή η interval άλγεβρα του Allen), μπορούμε να υπολογίσουμε τη σύνθεση χρησιμοποιώντας τους ορισμούς των σχέσεων. Στην περίπτωση του χωρικού λογισμού κατευθύνσεων οι μεταβλητές παίρνουν τιμές από τυχαίες περιοχές του χώρου. Το πεδίο αυτό δεν είναι καλά δομημένο και συνεπώς δεν μπορούμε να υπολογίσουμε τη σύνθεση με αυτό το τρόπο. Το γεγονός αυτό οδήγησε στον ορισμό μιας νέας μορφής σύνθεσης.

Η δεύτερη μορφή σύνθεσης ονομάζεται **αδύναμη σύνθεση** ή **σύνθεση βασιζόμενη στη συνέπεια**, αφού όπως θα διαπιστώσουμε αμέσως από τον ορισμό της, ο υπολογισμός της ανάγεται στο πρόβλημα της συνέπειας. Αν R_1, R_2 είναι δύο σχέσεις κατεύθυνσης, τότε η σύνθεσή τους $R_1 * R_2$ είναι η σχέση:

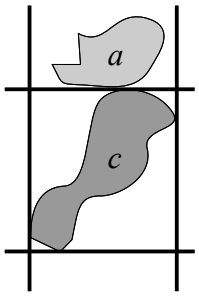
$$R_1 * R_2 = \{(a,b) \mid (a,b) \in R \wedge R \in D^* \wedge R \cap (R_1 \circ R_2) \neq \emptyset\}$$

Από τους ορισμούς της υπαρξιακής και της αδύναμης σύνθεσης προκύπτει ότι $R_1 \circ R_2 \subseteq R_1 * R_2$. Το πλεονεκτήματα της αδύναμης σύνθεσης σε σχέση με την υπαρξιακή είναι τα εξής:

- Παραμένουμε στο σύνολο των σύνολο των σχέσεων κατεύθυνσης 2^{D^*}, αφού αυτό εξ' ορισμού είναι κλειστό ως προς τις πράξεις της αδύναμης σύνθεσης, ένωσης, τομής και αντιστροφής.
- Ο ορισμός της μας παρέχει ένα προφανή τρόπο υπολογισμού, αφού για να υπολογίσουμε την αδύναμη σύνθεση δύο βασικών σχέσεων κατεύθυνσης έστω R_1, R_2 θα πρέπει απλώς να εξετάσουμε για κάθε βασική σχέση κατεύθυνσης R, αν είναι συνεπές το σύνολο $\{a R c, a R_1 b, b R_2 c\}$.

Εκτός από αυτό τον προφανή τρόπο υπολογισμού, ο Σκιαδόπουλος [SK04] έχει σχεδιάσει έναν περισσότερο πολύπλοκο και αποδοτικό αλγόριθμο, που υπολογίζει την αδύναμη σύνθεση δύο βασικών σχέσεων. Σε πολλές περιπτώσεις είναι πολύ να δύσκολο να αποδειχτεί ότι τα δύο είδη σύνθεσης ταυτίζονται, ενώ η μη ταύτισή τους αποδεικνύεται με ένα αντιπαράδειγμα, σαν αυτό που έχει δώσει ο Σκιαδόπουλος [SK05] και ακολουθεί στη συνέχεια.

Ας θεωρήσουμε τις μεταβλητές περιοχής a, b, c και τους περιορισμούς κατεύθυνσης $a N b$ και $b N c$. Από αυτούς τους περιορισμούς, ο μοναδικός περιορισμός που προκύπτει ο περιορισμός $a N c$, πράγμα που αναπαριστάται από το γεγονός ότι $N * N = N$. Αν τα δύο είδη σύνθεσης ταυτίζονταν, τότε θα είχαμε $N \circ N = N$. Αυτό εξ' ορισμού σημαίνει ότι για κάθε ζεύγος περιοχών a, c τέτοιες ώστε $a N c$, υπάρχει μια περιοχή b τέτοια ώστε $a N b$ και $b N c$. Αυτό όμως δεν συμβαίνει για τις περιοχές a, c του παρακάτω σχήματος:



Σχήμα 9: Παράδειγμα μη ταύτισης υπαρξιακής και αδύναμης σύνθεσης

3. Ο αλγόριθμος Σκιαδόπουλου – Κουμπαράκη

3.1 Γενική εικόνα

Προτού προχωρήσουμε στην παρουσίαση του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, θα πρέπει να δώσουμε μερικούς ακόμα ορισμούς.

Ορισμός 3.1 Ένας περιορισμός **διάταξης** είναι ένας τύπος που βρίσκεται σε κάποια από τις μορφές:

$$\inf_x(a) \sim \inf_x(b),\ \ \sup_x(a) \sim \sup_x(b),\ \ \inf_x(a) \sim \sup_x(b)$$
$$\inf_y(a) \sim \inf_y(b),\ \ \sup_y(a) \sim \sup_y(b),\ \ \inf_y(a) \sim \sup_y(b)$$

με $a, b \in REG^*$ και $\sim \in \{ <, >, \leq, \geq, = \}$.

Ορισμός 3.2 Ένα σύνολο περιορισμών διάταξης λέγεται **κανονικό** αν και μόνο αν περιλαμβάνει περιορισμούς της μορφής $\inf_x(a) < \sup_x(a)$ και $\inf_y(a) < \sup_y(a)$ για κάθε μεταβλητή περιοχής a που εμφανίζεται στο σύνολο.

Χρησιμοποιώντας τους ορισμούς των βασικών σχέσεων κατεύθυνσης που είδαμε στο προηγούμενο κεφάλαιο, μπορούμε να μετατρέψουμε ένα τυχαίο σύνολο βασικών περιορισμών κατεύθυνσης C, σε ένα σύνολο S που περιέχει δύο είδη περιορισμών:

- Ατομικούς περιορισμούς κατεύθυνσης.
- Περιορισμούς ένωσης που εκφράζουν ότι μια περιοχή είναι μη συνεκτική, δηλαδή περιορισμοί της μορφής $r = r_1 \cup .. \cup r_n$ όπου $r, r_1, .., r_n \in REG^*$.

Αν υπήρχε κάποιος αλγόριθμος που να μπορούσε να αποφασίσει για τη συνέπεια του συνόλου S, τότε προφανώς θα έλυνε το πρόβλημα της συνέπειας και για το αρχικό σύνολο C. Επειδή όμως δεν υπάρχει ένας τέτοιος γνωστός αλγόριθμος, ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη σαν πρώτο βήμα μετατρέπει το αρχικό σύνολο των περιορισμών κατεύθυνσης σε ένα ισοδύναμο (ως προς τη συνέπεια) σύνολο περιορισμών διάταξης. Επειδή ο αλγόριθμος αυτός είναι αρκετά εκτεταμένος, πριν προχωρήσουμε αναλυτικά στην περιγραφή του, θα αναφέρουμε περιληπτικά μια σκιαγράφηση των βημάτων, από τα οποία αποτελείται. Για να αποφανθεί λοιπόν ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη για τη συνέπεια ενός συνόλου από βασικούς περιορισμούς κατεύθυνσης C, στους οποίους εμφανίζονται οι μεταβλητές $a_1, .., a_n$, εκτελεί τα εξής τρία βήματα:

(1) Το πρώτο βήμα που ονομάζεται `TRANSFORM`, όπως αναφέραμε και πιο πριν

μετασχηματίζει το αρχικό σύνολο C των περιορισμών κατεύθυνσης σε ένα σύνολο περιορισμών διάταξης O. Το βήμα TRANSFORM εξετάζει διαδοχικά κάθε περιορισμό του C και στη συνέχεια εισάγει τους περιορισμούς διάταξης που αντιστοιχούν σε αυτόν, στο σύνολο O. Συνεπώς το σύνολο O περιέχει όλους εκείνους τους περιορισμούς διάταξης που περιλαμβάνουν τις προβολές ως προς τους άξονες x, y των μεταβλητών περιοχής $a_1,..,a_n$ καθώς επίσης και των συνιστωσών μεταβλητών που αντιστοιχούν σε αυτές. Αυτή η απεικόνιση γίνεται προφανώς χρησιμοποιώντας τους ορισμούς του προηγούμενου κεφαλαίου. Εκτός από αυτούς, εισάγονται επιπλέον περιορισμοί που συνεπάγονται από τον τρέχοντα περιορισμό κατεύθυνσης που εξετάζεται. Η εισαγωγή αυτών των περιορισμών θα δικαιολογηθεί στη συνέχεια.

(2) Στο δεύτερο βήμα, ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη βρίσκει μια λύση (εφόσον υπάρχει) για το σύνολο των περιορισμών διάταξης O. Προκειμένου να το επιτύχει αυτό, μπορεί να χρησιμοποιήσει διάφορους αλγόριθμους, όπως για παράδειγμα τον αλγόριθμο CSPAN του Van Beek [VB92]. Αν δεν υπάρχει καμία λύση του O, τότε ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη τερματίζει τυπώνοντας στην έξοδο 'Ασυνεπές'. Στην αντίθετη περίπτωση, ακολουθεί μια ανάθεση μη τετριμμένων mbb στις μεταβλητές περιοχής $a_1,..,a_n$ και στις συνιστώσες μεταβλητές που αντιστοιχούν σε αυτές, χρησιμοποιώντας τη λύση που βρέθηκε για το O. Στη συνέχεια, ο αλγόριθμος επεκτείνει αυτή τη λύση που βρέθηκε και κατασκευάζει μια μεγιστική λύση, με την έννοια ότι οποιαδήποτε περαιτέρω επέκταση της μεγιστικής αυτής λύσης, παύει να είναι λύση του O. Όπως θα διαπιστώσουμε στη συνέχεια, η κατασκευή αυτής της μεγιστικής λύσης είναι απαραίτητη για το τρίτο και τελευταίο βήμα του αλγορίθμου.

(3) Τα προηγούμενα δύο βήματα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, θεώρησαν για κάθε περιορισμό στο C, μόνο τους ατομικούς περιορισμούς κατεύθυνσης που προέκυπταν από τον ορισμό του. Αυτό το τελευταίο βήμα ασχολείται με τους περιορισμούς ένωσης που αντιστοιχούν σε κάθε μη ατομικό περιορισμό στο C. Μέχρι στιγμής, δεν έχει βρεθεί κάποιος αποδοτικός αλγόριθμος που να αντιμετωπίζει τέτοιου είδους μεικτούς περιορισμούς, δηλαδή διάταξης και ένωσης. Αυτό που κάνει λοιπόν ο αλγόριθμος σε αυτό το βήμα είναι ο μετασχηματισμός των περιορισμών ένωσης σε πολύπλοκη έκφραση από περιορισμούς διάταξης και στη συνέχεια πραγματοποιείται ένας έλεγχος για το εάν αυτή η πολύπλοκη έκφραση ικανοποιείται από τη μεγιστική λύση που προέκυψε από το δεύτερο βήμα. Αν ισχύει αυτό τότε ο αλγόριθμος τερματίζει και τυπώνει στην έξοδο 'Συνεπές', αλλιώς τυπώνει 'Ασυνεπές'. Αυτός ο έλεγχος πραγματοποιείται χρησιμοποιώντας τον αλγόριθμο που ονομάζεται **G**LOBAL**C**HECK**C**ONSTRAINT**NTB**. Πρόκειται για το πλέον ενδιαφέρον σημείο του αλγορίθμου, χάρη στο οποίο αποφεύγεται ο χειρισμός των υπολογιστικά

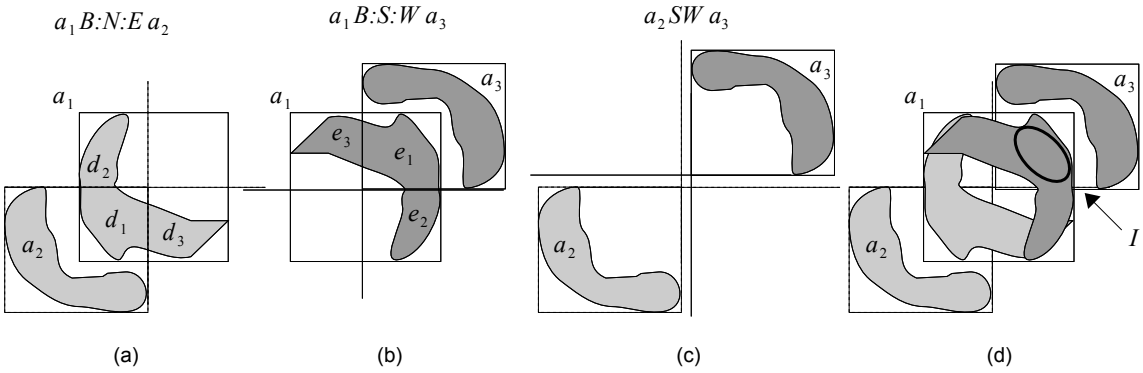
ακριβών περιορισμών ένωσης.

Εδώ θα πρέπει να αναφέρουμε πριν συνεχίσουμε, ότι καθ' όλη την περιγραφή του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη θα χρησιμοποιήσουμε το παράδειγμα που ακολουθεί, για να δούμε πως δουλεύει στην πράξη ο αλγόριθμος.

Ας υποθέσουμε ότι έχουμε τις μεταβλητές περιοχής a_1, a_2, a_3 που εμφανίζονται στο εξής σύνολο βασικών περιορισμών κατεύθυνσης:

$$C=\{a_1\,B{:}N{:}E\,a_2,\ a_1\,B{:}S{:}W\,a_3,\ a_2\,SW\,a_3\}$$

για το οποίο έχουμε μια οπτική αναπαράσταση στο σχήμα 10:



Σχήμα 10: Απεικόνιση παραδείγματος

Στο παραπάνω σχήμα βλέπουμε περιοχές στο (a), (b), (c) που ικανοποιούν τους περιορισμούς $a_1 B{:}N{:}E\,a_2,\ a_1 B{:}S{:}W\,a_3,\ a_2 SW\,a_3$ αντίστοιχα. Σε αυτό το παράδειγμα δεν υπάρχει μια ανάθεση που να ικανοποιεί όλους τους περιορισμούς ταυτόχρονα, δηλαδή το C είναι ασυνεπές. Αυτό επιβεβαιώνεται κοιτώντας το (d). Παρατηρούμε ότι ο πρώτος περιορισμός του C, εξαναγκάζει την περιοχή a_1 να μοιάζει με την περιοχή του γκρι ανοιχτού χρώματος, ενώ ο δεύτερος περιορισμός εξαναγκάζει την ίδια περιοχή να μοιάζει με την περιοχή του γκρι σκούρου χρώματος. Ας προσέξουμε τώρα την περιοχή I που έχει κυκλωθεί. Βλέπουμε ότι ο περιορισμός $a_1 B{:}N{:}E\,a_2$ απαιτεί ότι η περιοχή I δεν ανήκει στο a_1, ενώ αντίθετα ο περιορισμός $a_1 B{:}S{:}W\,a_3$ απαιτεί να ανήκει η I στο a_1, γεγονός που κάνει το σύνολο να είναι ασυνεπές.

3.2 Πρώτο βήμα – Μετασχηματισμός

Έστω C ένα σύνολο βασικών περιορισμών κατεύθυνσης, στους οποίους εμφανίζονται οι μεταβλητές $a_1, .., a_n$. Το πρώτο βήμα που λέγεται TRANSFORM,

όπως αναφέραμε και πιο πριν μετασχηματίζει το αρχικό σύνολο C των περιορισμών διάταξης σε ένα σύνολο O από περιορισμούς κατεύθυνσης. Αυτός ο αλγόριθμος έχει τέσσερα βήματα τα οποία εφαρμόζει διαδοχικά σε κάθε περιορισμό κατεύθυνσης του C. Ας θεωρήσουμε λοιπόν ένα τυχαίο περιορισμό $a_i R_1 : \ldots : R_k a_j$ που ανήκει στο C.

3.2.1 Βήμα 1

Σε αυτό το βήμα ο αλγόριθμος TRANSFORM, χρησιμοποιεί τους ορισμούς των ατομικών σχέσεων κατεύθυνσης και εισάγει τους αντίστοιχους περιορισμούς διάταξης στο σύνολο O. Πιο συγκεκριμένα, αυτό το βήμα διακρίνει δύο περιπτώσεις:

1) Αν $k=1$ τότε ο περιορισμός $a_i R_1 : \ldots : R_k a_j$ είναι ατομικός, οπότε απλώς εισάγουμε τους περιορισμούς διάταξης που αντιστοιχούν σε αυτό τον ατομικό περιορισμό, σύμφωνα με τον ορισμό από το προηγούμενο κεφάλαιο της ατομικής σχέσης που περιλαμβάνει.
2) Αν $k>1$ τότε ο περιορισμός $a_i R_1 : \ldots : R_k a_j$ δεν είναι ατομικός, οπότε θα πρέπει να θεωρήσουμε τις συνιστώσες μεταβλητές $a_{il}^j, .., a_{ik}^j$ που αντιστοιχούν στην μεταβλητή a_i ως προς τον περιορισμό $a_i R_1 : \ldots : R_k a_j$. Παρατηρούμε ότι θα πρέπει να εισάγουμε και ένα δείκτη j σε αυτές τις συνιστώσες μεταβλητές προκειμένου να τις ξεχωρίζουμε, αφού είναι πιθανό η μεταβλητή a_i να εμφανίζεται σαν κύρια μεταβλητή και σε άλλους μη ατομικούς περιορισμούς κατεύθυνσης. Στη συνέχεια συνεχίζουμε όπως πριν, δηλαδή εισάγουμε τους περιορισμούς διάταξης που αντιστοιχούν στους ατομικούς περιορισμούς $a_{il}^j R_1 a_j, .., a_{ik}^j R_k a_j$ που δημιουργήθηκαν.

Ας δούμε τι γίνεται στο παράδειγμα που χρησιμοποιούμε. Θα πρέπει να Θυμίσουμε ότι έχουμε $C=\{a_1 B : N : E a_2,\ a_1 B : S : W a_3,\ a_2 SW a_3\}$. Εξετάζοντας τον πρώτο περιορισμό βλέπουμε ότι είναι μη ατομικός, οπότε θεωρούμε τις συνιστώσες μεταβλητές d_1, d_2, d_3 του a_1, από τους περιορισμούς των οποίων έχουμε:

$$\underbrace{\inf_x(a_2) \le \inf_x(d_1),\ \sup_x(d_1) \le \sup_x(a_2),\ \inf_y(a_2) \le \inf_y(d_1),\ \sup_y(d_1) \le \sup_y(a_2)}_{d_1 B a_2}$$

$$\underbrace{\inf_x(a_2) \le \inf_x(d_2),\ \sup_x(d_2) \le \sup_x(a_2),\ \sup_y(a_2) \le \inf_y(d_2)}_{d_2 N a_2}$$

$$\underbrace{\sup_x(a_2)\le\inf_x(d_3),\inf_y(a_2)\le\inf_y(d_3),\sup_y(d_3)\le\sup_y(a_2)}_{d_3\,E\,a_2}$$

Ομοίως δουλεύουμε και για τον δεύτερο περιορισμό, εισάγοντας νέες συνιστώσες μεταβλητές e_1, e_2, e_3. Ο τρίτος περιορισμός είναι ατομικός, συνεπώς απλά προσθέτουμε στο σύνολο O τους ακόλουθους περιορισμούς διάταξης:

$$\underbrace{\sup_x(a_2)\le\inf_x(a_3),\sup_y(a_2)\le\inf_y(a_3)}_{a_2\,SW\,a_3}$$

3.2.2 Βήμα 2

Στο δεύτερο βήμα του ο αλγόριθμος TRANSFORM εισάγει τους προφανείς περιορισμούς διάταξης που ισχύουν για όλες τις μεταβλητές περιοχής (τις αρχικές και τις συνιστώσες) και σχετίζουν τις προβολές των άκρων τους. Συνεπώς το σύνολο των περιορισμών διάταξης γίνεται κανονικό.

Στο παράδειγμα που χρησιμοποιούμε, είδαμε ότι από το πρώτο βήμα θεωρήσαμε τις συνιστώσες μεταβλητές $\{d_1, d_2, d_3, e_1, e_2, e_3\}$ του a_1, επομένως για κάθε μεταβλητή $r\in\{a_1, a_2, a_3, d_1, d_2, d_3, e_1, e_2, e_3\}$ εισάγουμε τους περιορισμούς:

$\inf_x(r)<\sup_x(r)$ και $\inf_y(r)<\sup_y(r)$

3.2.3 Βήμα 3

Σε αυτό το βήμα του ο αλγορίθμου TRANSFORM, εισάγονται οι περιορισμοί που εκφράζουν ότι οι συνιστώσες μεταβλητές $a^j_{il},..,a^j_{ik}$ που δημιουργήθηκαν στο πρώτο βήμα, είναι υποπεριοχές της a_i. Οπότε για κάθε συνιστώσα μεταβλητή r του a_i εισάγουμε στο σύνολο O τους ακόλουθους περιορισμούς:

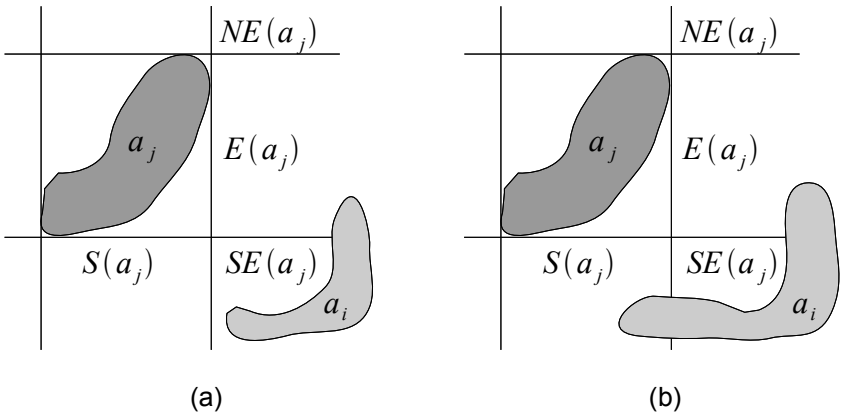
$$\inf_x(a_i)\le\inf_x(r),\sup_x(r)\le\sup_x(a_i),\inf_y(a_i)\le\inf_y(r),\sup_y(r)\le\sup_y(a_i)$$

Στο παράδειγμα που χρησιμοποιούμε, είδαμε ότι για το a_1 προέκυψαν οι συνιστώσες μεταβλητές $\{d_1, d_2, d_3, e_1, e_2, e_3\}$, επομένως πρέπει να προσθέσουμε στο O για όλα τα $r\in\{d_1, d_2, d_3, e_1, e_2, e_3\}$ τους παραπάνω περιορισμούς.

3.2.4 Βήμα 4

Το τέταρτο βήμα του ο αλγορίθμου TRANSFORM είναι το πιο ενδιαφέρον, γιατί σύμφωνα με αυτό εισάγονται οι πιο αυστηρές σχέσεις διάταξης που ισχύουν μεταξύ των άκρων των προβολών δύο μεταβλητών περιοχής. Δεδομένου ενός μη

ατομικού περιορισμού $a_i R_1 :...: R_k a_j$ παρατηρούμε ότι ενδεχομένως να ισχύουν κάποιοι περιορισμοί διάταξης οι οποίοι να μην έχουν εισαχθεί από τα προηγούμενα βήματα. Για παράδειγμα, όπως βλέπουμε στο σχήμα 11 για τις περιπτώσεις (a),(b) ισχύουν οι περιορισμοί $\sup_x(a_j) \leq \inf_x(a_i)$, $\inf_x(a_j) \leq \inf_x(a_i)$ αντίστοιχα.



Σχήμα 11: Αυστηρότεροι περιορισμοί διάταξης

Ωστόσο αυτοί οι περιορισμοί δεν προκύπτουν από τα βήματα 1 – 3 του αλγορίθμου TRANSFORM. Συνεπώς το τέταρτο βήμα θα πρέπει να εξετάσει όλους τους μη ατομικούς περιορισμούς κατεύθυνσης και να εισάγει αυτούς τους αυστηρότερους περιορισμούς διάταξης που αναφέραμε. Με μια απλή ανάλυση περιπτώσεων μπορούμε να διαπιστώσουμε ότι θα πρέπει να εξεταστούν 8 πιθανές περιπτώσεις. Για κάθε μη ατομικό περιορισμό $a_i R_1 :...: R_k a_j$ του C, εξετάζεται αν το σύνολο των σχέσεων $\{R_1, .., R_k\}$ είναι υποσύνολο κάποιου ή κάποιων από τα ακόλουθα σύνολα:

$\{NE, E, SE\}, \{NE, E, SE, N, B, S\}, \{NW, W, SW\}, \{NW, W, SW, N, B, S\},$
$\{NW, N, NE\}, \{NW, N, NE, W, B, E\}, \{SW, S, SE\}, \{SW, S, SE, W, B, E\}$

Αν ισχύει αυτό, τότε εισάγονται οι αντίστοιχοι περιορισμοί που φαίνονται στον ψευδοκώδικα του αλγορίθμου που ακολουθεί στη συνέχεια. Για παράδειγμα, αν το σύνολο των σχέσεων $\{R_1, .., R_k\}$ είναι υποσύνολο του $\{NE, E, SE\}$ τότε εισάγεται στο O ο περιορισμός:

$\sup_x(a_j) \leq \inf_x(a_i)$

Αυτό συμβαίνει γιατί σε μια τέτοια περίπτωση, θα πρέπει να αναπαριστάται στο σύνολο των περιορισμών διάταξης το γεγονός ότι ' η περιοχή a_j βρίσκεται στα

ανατολικά της περιοχής a_i '. Στο προηγούμενο σχήμα όπως είδαμε, παρουσιάζεται μια τέτοια περίπτωση. Με ανάλογους ισχυρισμούς μπορούμε να διαπιστώσουμε ποιοι περιορισμοί διάταξης ισχύουν για τις υπόλοιπες περιπτώσεις.

Ας δούμε τώρα τι γίνεται στο παράδειγμα που χρησιμοποιούμε. Θυμίζουμε ότι έχουμε $C=\{a_1 B{:}N{:}E\, a_2,\ a_1 B{:}S{:}W\, a_3,\ a_2 SW\, a_3\}$. Εξετάζοντας λοιπόν τον πρώτο περιορισμό βλέπουμε ότι είναι μη ατομικός και επιπλέον ισχύει ότι:

$\{B, N, E\} \subseteq \{NE, E, SE, N, B, S\}$ και $\{B, N, E\} \subseteq \{NW, N, NE, W, B, E\}$

Οπότε το τέταρτο βήμα προσθέτει στο O τους εξής περιορισμούς:

$\inf_x(a_2) \leq \inf_x(a_1)$ και $\inf_y(a_2) \leq \inf_y(a_1)$

Ομοίως εργαζόμαστε και για τον δεύτερο περιορισμό, αφού παρατηρούμε ότι είναι επίσης μη ατομικός και επιπλέον ισχύει ότι:

$\{B, S, W\} \subseteq \{NW, W, SW, N, B, S\}$ και $\{B, S, W\} \subseteq \{SW, S, SE, W, B, E\}$

3.2.5 Ψευδοκώδικας αλγορίθμου μετασχηματισμού

Ο ψευδοκώδικας που ακολουθεί, συνοψίζει τις λειτουργίες που επιτελούνται στα τέσσερα βήματα του αλγορίθμου TRANSFORM.

```
Algorithm TRANSFORM
Input: Set of basic direction contraints C
Output: Set of order constraints O

Begin
 O = Ø
 For each a_i R_1 :..: R_k a_j ∈ C Do

#      Step 1
       If k = 1 Then
              Add to O the order constraints that define atomic
              constraint a_i R_1 a_j
       Else
              Introduce component variables a_i1^j, .., a_ik^j of a_i
              For each a_it^j ∈ {a_i1^j, .., a_ik^j} Do
                     Add to O the order constraints that define atomic
                     constraint a_it^j R_t a_j
              End For each
```

End If

\# **Step 2**

For each $r \in \{a_i, a_j, a_{il}^j, .., a_{ik}^j\}$ **Do**

$O = O \cup \{\inf_x(r) < \sup_x(r), \inf_y(r) < \sup_y(r)\}$

End For each

\# **Step 3**

For each $r \in \{a_{il}^j, .., a_{ik}^j\}$ **Do**

$O = O \cup \{\inf_x(a_i) \le \inf_x(r), \sup_x(r) < \sup_x(a_i)\}$

$O = O \cup \{\inf_y(a_i) \le \inf_y(r), \sup_y(r) < \sup_y(a_i)\}$

End For each

\# **Step 4**

If $k > 1$ **Then**

If $\{R_1, .., R_k\} \subseteq \{NE, E, SE\}$ **Then**

$O = O \cup \{\sup_x(a_j) \le \inf_x(a_i)\}$

Else If $\{R_1, .., R_k\} \subseteq \{NE, E, SE, N, B, S\}$ **Then**

$O = O \cup \{\inf_x(a_j) \le \inf_x(a_i)\}$

If $\{R_1, .., R_k\} \subseteq \{NW, W, SW\}$ **Then**

$O = O \cup \{\sup_x(a_i) \le \inf_x(a_j)\}$

Else If $\{R_1, .., R_k\} \subseteq \{NW, W, SW, N, B, S\}$ **Then**

$O = O \cup \{\sup_x(a_i) \le \sup_x(a_j)\}$

If $\{R_1, .., R_k\} \subseteq \{NW, N, NE\}$ **Then**

$O = O \cup \{\sup_y(a_j) \le \inf_y(a_i)\}$

Else If $\{R_1, .., R_k\} \subseteq \{NW, N, NE, W, B, E\}$ **Then**

$O = O \cup \{\inf_y(a_j) \le \inf_y(a_i)\}$

If $\{R_1, .., R_k\} \subseteq \{SW, S, SE\}$ **Then**

$O = O \cup \{\sup_y(a_i) \le \inf_y(a_j)\}$

Else If $\{R_1, .., R_k\} \subseteq \{SW, S, SE, W, B, E\}$ **Then**

$O = O \cup \{\sup_y(a_i) \le \sup_y(a_j)\}$

End If

End For each

Return O

End

3.2 Δεύτερο βήμα – Εύρεση και επέκταση λύσης

3.2.1 Εύρεση λύσης – Γενικά

Στο δεύτερο βήμα του, ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη ελέγχει τη συνέπεια του συνόλου των περιορισμών διάταξης O που προέκυψαν μετά την εκτέλεση του πρώτου βήματος (δηλ. του αλγορίθμου TRANSFORM). Σύμφωνα με τα προηγούμενα που αναφέραμε, προκύπτει ότι όλοι οι περιορισμοί που εισήχθησαν στο σύνολο O από τον αλγόριθμο TRANSFORM, συνεπάγονται λογικά από τους περιορισμούς κατεύθυνσης του αρχικού συνόλου C. Επομένως, αν βρεθεί ότι το σύνολο O είναι ασυνεπές, θα πρέπει να είναι ασυνεπές και το σύνολο C. Σε αυτή την περίπτωση ο αλγόριθμος τερματίζεται, τυπώνοντας στην έξοδο 'Ασυνεπές'.

Προκειμένου να γίνει ο έλεγχος της συνέπειας του συνόλου O, μπορούν να χρησιμοποιηθούν διάφοροι αλγόριθμοι. Η ουσία είναι ότι το σύνολο O είναι ένα δίκτυο (με την έννοια της θεωρίας γραφημάτων) άλγεβρας σημείων (την οποία από εδώ και στο εξής θα αναφέρουμε για λόγους συντομογραφίας ως PA – Point Algebra) και σαν τέτοιο μπορούν να εφαρμοστούν σε αυτό αρκετοί αλγόριθμοι που αναφέρονται στη βιβλιογραφία. Επιπλέον, επειδή έχουμε μια ειδική περίπτωση ενός PA δικτύου, αφού οι μόνες σχέσεις που εμφανίζονται στο O είναι οι $\{<, \leq\}$, θα διαπιστώσουμε αργότερα ότι είναι δυνατή η απλοποίηση κάποιων από αυτούς τους αλγόριθμους.

Σύμφωνα με τον Knuth, όπως αναφέρεται στο [VB92], μπορούμε να χρησιμοποιήσουμε τοπολογική διάταξη σε PA δίκτυα, εάν το σύνολο των σχέσεων περιορίζεται σε αυστηρές ανισότητες, δηλαδή είναι το $\{<, >, ?\}$. Ως γνωστόν, η τοπολογική διάταξη απαιτεί χρόνο $O(n^2)$ αλλά δυστυχώς δεν μπορεί να εφαρμοστεί στη δική μας περίπτωση λόγω της ύπαρξης της σχέσης $\leq$.

Αν οι επιτρεπόμενες σχέσεις είναι οι $\{<, \leq, =, >, \geq, ?\}$ δηλαδή αν δεν επιτρέπεται η $\neq$, ένα PA δίκτυο μπορεί να θεωρηθεί ως ένα σύνολο γραμμικών ισοτήτων και ανισοτήτων, όπου κάθε στοιχείο αυτού του συνόλου είναι σε μία από τις μορφές: $x_i - x_j < 0$, $x_i - x_j \leq 0$, $x_i - x_j = 0$. Άρα μπορούμε να χρησιμοποιήσουμε τους γνωστούς αλγόριθμους που εφαρμόζονται σε σύνολα γραμμικών ανισοτήτων, πχ. τον simplex ή τον αλγόριθμο του Karmarkar, προκειμένου να βρούμε μια λύση. Ωστόσο είναι επιθυμητό να χρησιμοποιήσουμε περισσότερο αποδοτικούς αλγόριθμους, πράγμα που επιτυγχάνεται όταν το γραμμικό πρόβλημα έχει μια συγκεκριμένη μορφή που εμφανίζεται στο λεγόμενο πρόβλημα του συντομότερου μονοπατιού.

Στο πρόβλημα του συντομότερου μονοπατιού καλούμαστε να βρούμε το συντομότερο μονοπάτι μεταξύ δύο κορυφών s, t ενός διευθυνόμενου γραφήματος, πάνω στις ακμές του οποίου υπάρχουν 'βάρη' (τυπικά οι αποστάσεις μεταξύ των προσκείμενων κορυφών). Ο Παπαδημητρίου, όπως αναφέρεται στο [VB92], έχει δείξει ένα τρόπο με τον οποίο το πρόβλημα του συντομότερου μονοπατιού μπορεί να αναχθεί σε ένα γραμμικό πρόβλημα. Ας υποθέσουμε ότι l_{ij} είναι το βάρος της

διευθυνόμενης ακμής (i, j) και x_i είναι το μήκος του συντομότερου μονοπατιού από το s στο i. Θεωρούμε προφανώς ότι το μήκος του συντομότερου μονοπατιού από το s στον εαυτό του είναι μηδέν. Ο στόχος μας είναι η ελαχιστοποίηση του x_t, δηλαδή του μήκους του συντομότερου μονοπατιού από το s στο t. Επειδή το συντομότερο μονοπάτι από το s στο j μπορεί να περνάει από το i, θα πρέπει να ισχύει $x_j \le x_i + l_{ij}$ ή ισοδύναμα $x_j - x_i \le l_{ij}$. Συνεπώς προκύπτει το εξής γραμμικό πρόβλημα:

$$
\begin{aligned}
&Min\, x_t \\
&x_j - x_i \le l_{ij}\ ,\ \ i, j = 1,.., n \\
&x_s = 0
\end{aligned}
$$

Αν όλα τα βάρη l_{ij} είναι μη αρνητικά, τότε μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Dijkstra που απαιτεί χρόνο $O(n^2)$, διαφορετικά θα πρέπει να χρησιμοποιήσουμε τον αλγόριθμο Floyd-Warshall που απαιτεί χρόνο $O(n^3)$. Το μόνο που μένει τώρα, είναι να δείξουμε πως γίνεται η μετατροπή του προβλήματός μας στο πρόβλημα του συντομότερου μονοπατιού. Η μετατροπή γίνεται με τον εξής τρόπο:

$$
\begin{aligned}
&x_i = x_j \ \rightarrow\ x_i - x_j \le 0,\ \ x_j - x_i \le 0 \\
&x_i \le x_j \ \rightarrow\ x_i - x_j \le 0,\ \ x_j - x_i \le +\infty \\
&x_i < x_j \ \rightarrow\ x_i - x_j \le -e,\ \ x_j - x_i \le +\infty
\end{aligned}
$$

Όπου $-e$ είναι μια μικρή αρνητική τιμή που χρησιμοποιείται για τη μετατροπή της γνήσιας ανισότητας σε μη γνήσια. Συνοψίζοντας αυτή τη μέθοδο, θα πρέπει να αναφέρουμε ότι αν οι επιτρεπόμενες σχέσεις στο δίκτυο είναι οι $\{\le, =, \ge, ?\}$ τότε μπορεί να χρησιμοποιηθεί ο αλγόριθμος Dijkstra που είναι $O(n^2)$, αλλιώς αν το σύνολο των επιτρεπόμενων σχέσεων είναι το $\{<, \le, =, \ge, >, ?\}$, τότε θα πρέπει να εφαρμοστεί ο αλγόριθμος Floyd-Warshall που είναι $O(n^3)$.

Σύμφωνα με τον Van Beek [VB92], ένας από τους πλέον αποδοτικούς αλγόριθμους για το πρόβλημα είναι ο CSPAN. Αυτός έχει πολυπλοκότητα $O(n^2)$ για αυτό και επιλέχτηκε να χρησιμοποιηθεί από τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη. Ακολουθεί στη συνέχεια αναλυτική περιγραφή του.

3.2.2 Εύρεση λύσης – Χρησιμοποιώντας τον αλγόριθμο CSPAN

Η βασική ιδέα για την δημιουργία του αλγορίθμου CSPAN από τον Van Beek [VB92], προέκυψε από την παρατήρηση ότι η τοπολογική διάταξη δεν μπορεί να δουλέψει από μόνη της, διότι στη γενική περίπτωση οι επιτρεπόμενες σχέσεις είναι οι $\{\emptyset, <, \le, =, \ge, >, \ne, ?\}$ ενώ η τοπολογική διάταξη επιτρέπει μόνο τις

σχέσεις $\{<, >, ?\}$. Εξάλλου, για να εκτελεστεί η τοπολογική διάταξη, θα πρέπει ως γνωστόν το γράφημα να είναι ακυκλικό, γεγονός που δεν ισχύει εν γένει. Διαισθητικά θα πρέπει κατά κάποιο τρόπο να 'αφαιρεθούν' οι 'προβληματικές' σχέσεις $\{\not=, =, \leq, \geq, \neq\}$ και να εξασφαλιστεί ότι το γράφημα που θα προκύψει είναι ακυκλικό, ώστε να μπορέσει στη συνέχεια να εφαρμοστεί σε αυτό η τοπολογική διάταξη. Για την περιγραφή του αλγορίθμου που ακολουθεί, θεωρούμε ότι ένα PA δίκτυο αναπαριστάται από τον πίνακα γειτνίασης C, όπου θεωρούμε ότι το στοιχείο C_{ij} είναι η ετικέτα της ακμής (i, j).

'Αφαίρεση' της $=$

Προκειμένου να αφαιρεθεί η $=$ από το δίκτυο, θα πρέπει να βρεθούν τα όλα τα ζεύγη των κορυφών στα οποία οι κορυφές εξαναγκάζονται να είναι ίσες, με την έννοια ότι σε κάθε δυνατή λύση (αν υπάρχει τέτοια) οι μεταβλητές που αντιστοιχούν σε αυτές τις κορυφές παίρνουν την ίδια τιμή. Ουσιαστικά αυτό που πρέπει να γίνει, είναι μια διαμέριση του συνόλου των κορυφών σε κλάσεις ισοδυναμίας, έτσι ώστε δύο κορυφές να ανήκουν στην αυτή κλάση ισοδυναμίας αν και μόνο αν είναι ίσες. Προφανώς δύο τυχαίες κορυφές u, w είναι ίσες αν και μόνο αν συμμετέχουν σε κάποιο κύκλο της μορφής:

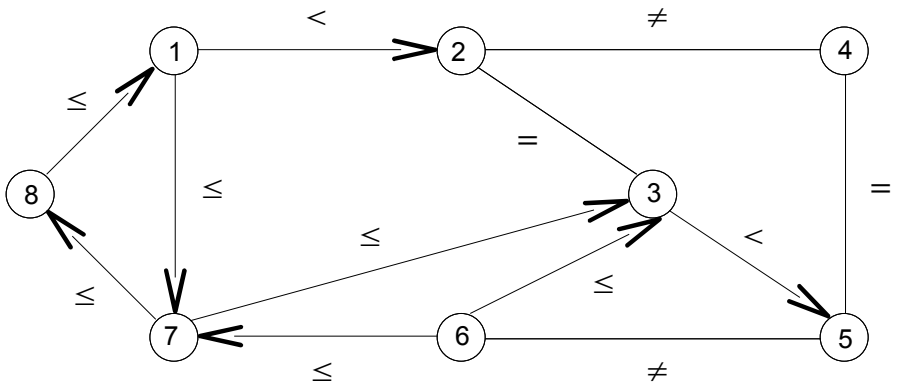
$$u \leq .. \leq w \leq .. \leq u$$

όπου μπορεί κάποιες από τις $\leq$ να είναι $=$. Συνεπώς οι u, w ανήκουν στην αυτή κλάση ισοδυναμίας αν και μόνο αν υπάρχει ένα μονοπάτι από τη u στη v και από τη v στη u, που διέρχονται από ακμές των οποίων οι ετικέτες είναι κάποια από τις σχέσεις $\leq$ ή $=$. Ο καθορισμός των κλάσεων ισοδυναμίας είναι ένα γνωστό πρόβλημα στη θεωρία γραφημάτων και ταυτίζεται με την εύρεση των ισχυρά συνεκτικών συνιστωσών. Ο αλγόριθμος του Tarjan είναι ένας γνωστός αποδοτικός αλγόριθμος που βρίσκει τις ισχυρά συνεκτικές συνιστώσες σε χρόνο $O(n^2)$.

Στη συνέχεια μπορούμε να κατασκευάσουμε ένα συνεπτυγμένο γράφημα $\hat{C}$, του οποίου οι κορυφές είναι οι ισχυρά συνεκτικές συνιστώσες $\{S_1, .., S_m\}$ του C. Οι ετικέτες των ακμών του $\hat{C}$ δίνονται από τον τύπο:

$$\hat{C}_{S_i, S_j} = \bigcap_{\substack{u \in S_i \\ w \in S_j}} C_{uw}, \quad i, j = 1, .., m$$

Ο παραπάνω τύπος εκφράζει τη διαισθητική ιδέα ότι κάθε ακμή στο συνεπτυγμένο γράφημα, θα πρέπει να έχει σαν ετικέτα την πιο 'περιορισμένη' ετικέτα μεταξύ αυτών που βρίσκονται στις ακμές του αρχικού γραφήματος και ενώνουν τις κορυφές της μίας με της άλλης ισχυρά συνεκτικής συνιστώσας. Το παράδειγμα που ακολουθεί στο σχήμα 12, ξεκαθαρίζει πλήρως την κατάσταση.



Σχήμα 12: Παράδειγμα PA δικτύου

Ας υποθέσουμε λοιπόν ότι έχουμε το PA δίκτυο που απεικονίζεται στο σχήμα 12. Όπως συνηθίζεται, οι ακμές (i,i) καθώς και εκείνες οι οποίες έχουν σαν ετικέτα το $?$ δεν απεικονίζονται στο σχήμα. Επίσης, οι ακμές που υπάρχουν και προς τις δύο κατευθύνσεις μεταξύ ενός ζεύγους κορυφών, π.χ. $(4,5)$ και $(5,4)$, για λόγους ευκολίας απεικονίζονται στο σχήμα σαν μία μη διευθυνόμενη ακμή. Εφαρμόζοντας στο παραπάνω δίκτυο τον αλγόριθμο του Tarjan προκύπτουν οι εξής ισχυρά συνεκτικές συνιστώσες:

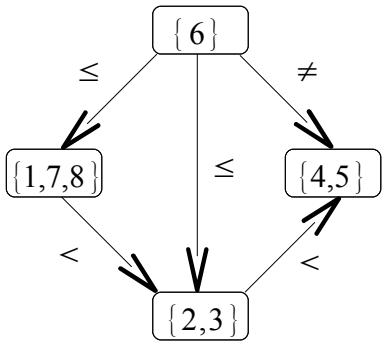
$$S_1=\{1,7,8\},\ S_2=\{2,3\},\ S_3=\{4,5\},\ S_4=\{6\}$$

Αυτές είναι και οι κορυφές του συνεπτυγμένου γραφήματος. Ο υπολογισμός των ακμών μεταξύ τους γίνεται με την εφαρμογή του τύπου που δώσαμε πιο πάνω. Για παράδειγμα, για τις ακμές (S_1,S_1) και (S_2,S_3) έχουμε αντίστοιχα:

$$\begin{aligned}
\hat{C}_{S_1,S_1} &= C_{17}\cap C_{18}\cap C_{71}\cap C_{78}\cap C_{81}\cap C_{87} \Rightarrow \\
\hat{C}_{S_1,S_1} &= \{<,=\}\cap\{>,=\}\cap\{>,=\}\cap\{<,=\}\cap\{<,=\}\cap\{>,=\} \Rightarrow \\
\hat{C}_{S_1,S_1} &= \{=\}
\end{aligned}$$

$$\begin{aligned}
\hat{C}_{S_2,S_3} &= C_{24}\cap C_{25}\cap C_{34}\cap C_{35} \\
\hat{C}_{S_2,S_3} &= \{<,>\}\cap\{<,>,=\}\cap\{<,>,=\}\cap\{<\} \\
\hat{C}_{S_2,S_3} &= \{<\}
\end{aligned}$$

Με αυτό το τρόπο υπολογίζουμε όλες τις ακμές του συνεπτυγμένου γραφήματος, το οποίο φαίνεται στο ακόλουθο σχήμα:



Σχήμα 13: Συνεπτυγμένο γράφημα

'Αφαίρεση' της $\not\varnothing$

Προκειμένου να αφαιρέσουμε την κενή σχέση από το δίκτυο, θα πρέπει να διαπιστώσουμε αν αυτό είναι ασυνεπές, αφού η ύπαρξη αυτής καθεαυτής μιας κενής σχέσης, τούτο ακριβώς εκφράζει. Παρατηρούμε ότι το δίκτυο είναι ασυνεπές αν και μόνο αν υπάρχει ένας κύκλος σε κάποια από τις εξής μορφές:

i. $u = .. = w \neq u$
ii. $u \leq .. \leq w \leq .. \leq u \neq w$, όπου μπορεί κάποιες από τις $\leq$ να είναι $=$
iii. $u < .. < w < .. < u$, όπου μπορεί όλες εκτός από τουλάχιστον μία από τις $<$ να είναι $\leq$ ή $=$

Οι πρώτες δύο περιπτώσεις ασυνέπειας ανιχνεύονται όταν αναγνωρίζουμε όλα τα ζεύγη των κορυφών που εξαναγκάζονται να είναι ίσα και τα συν πτύσουμε σε μια κορυφή. Μπορούμε επίσης να ανιχνεύσουμε και την τρίτη περίπτωση, εξετάζοντας απλώς τις ακμές που έχουν ετικέτα $<$, κατά την αναγνώριση των ισχυρά συνεκτικών συνιστωσών.

'Αφαίρεση' των $\leq, \geq$

Οι σχέσεις $\leq, \geq$ μπορούν πλέον να αντικατασταθούν από τις σχέσεις $<, >$. Αυτό συμβαίνει γιατί υποθέτοντας ότι έχουμε αφαιρέσει τις σχέσεις $=, \not\varnothing$, τότε δεν χρειάζεται οι εναπομένουσες ακμές να περιέχουν την $=$, αφού οι μεταβλητές που συνδέουν δεν είναι κατά ανάγκη ίσες, αφού αν ήταν τότε θα είχαν μπει μέσα στην ίδια ισχυρά συνεκτική συνιστώσα. Επομένως αν υπάρχει μια συνεπής ανάθεση όταν μια ακμή έχει σαν ετικέτα τη $\leq$, τότε υπάρχει μια συνεπής ανάθεση όταν σε αυτή την ακμή βάλουμε τη $<$ στη θέση της $\leq$. Μετά από αυτές τις αλλαγές, θα εξακολουθεί να υπάρχει μια συνεπής ανάθεση, αν βέβαια υπήρχε πριν.

'Αφαίρεση' της $\neq$

Μπορούμε τώρα να εκτελέσουμε την τοπολογική διάταξη στο συνεπτυγμένο γράφημα (που προφανώς είναι ακυκλικό) και στη συνέχεια να βρούμε μια συνεπή ανάθεση. Από την κατασκευή του αλγορίθμου και τον χειρισμό των υπόλοιπων 'ανεπιθύμητων' σχέσεων $\{\not\varnothing, =, \leq, \geq\}$ εξασφαλίζεται αμέσως ο σωστός χειρισμός της $\neq$. Η έξοδος της τοπολογικής διάταξης είναι ως γνωστόν μια απαρίθμηση των κορυφών του γραφήματος στο οποίο εκτελέστηκε. Για το

συγκεκριμένο παράδειγμα που αναφέραμε πριν, η έξοδος είναι:

$\{4,5\},\ \{2,3\},\ \{1,7,8\},\ \{6\}$

Οπότε μια συνεπής ανάθεση μπορεί να προκύψει χρησιμοποιώντας μια γνησίως αύξουσα αλυσίδα τιμών, δηλαδή αρχίζοντας με μια τυχαία τιμή (π.χ. το 1) για την πρώτη ισχυρά συνεκτική συνιστώσα και κατόπιν συνεχίζοντας με τις υπόλοιπες αυξάνοντας διαδοχικά αυτή τη τιμή. Τη τιμή αυτή λαμβάνουν όλες οι μεταβλητές που περιέχονται στη συνιστώσα. Για το παραπάνω παράδειγμα που χρησιμοποιήσαμε μια συνεπής ανάθεση είναι η εξής:

$\underbrace{\{4,5\}}_{1},\ \underbrace{\{2,3\}}_{2},\ \underbrace{\{1,7,8\}}_{3},\ \underbrace{\{6\}}_{4}$

Όπως αναφέραμε και νωρίτερα, ο αλγόριθμος CSPAN έχει χρονική πολυπλοκότητα $O(n^2)$. Αυτό οφείλεται στο ότι η εύρεση των ισχυρά συνεκτικών συνιστωσών απαιτεί χρόνο $O(n^2)$, για την κατασκευή του συνεπτυγμένου γραφήματος εξετάζεται κάθε ακμή του αρχικού μία μόνο φορά οπότε το κόστος και σε αυτό το βήμα είναι $O(n^2)$ (και συγκεκριμένα $O(e)$, όπου e το πλήθος των ακμών) και τέλος η τοπολογική διάταξη απαιτεί ως γνωστόν επίσης χρόνο $O(n^2)$ (μάλιστα απαιτεί χρόνο $O(m^2)$, όπου m το πλήθος των ισχυρά συνεκτικών συνιστωσών για το οποίο βέβαια ισχύει ότι $m \le n$).

Ο ψευδοκώδικας που ακολουθεί, συνοψίζει τις λειτουργίες που επιτελεί ο αλγόριθμος CSPAN.

```
Algorithm CSPAN
Input: A PA network represented by its adjacency matrix C
Output: A consistent assignment to its variables, if there is such an
assignment, else it returns 'Inconsistent'

Begin
# Step 1
Identify the strongly connected components of C using only edges
labeled with {<},{<,=},{=} . Lets assume that S_1,..,S_m is the output
of the Tarjan algorithm.

# Step 2
For i,j := 1,..,m Do
        Ĉ_{S_i,S_j} = {<,=,>}
        For each u∈S_i, w∈S_j Do
                Ĉ_{S_i,S_j} = Ĉ_{S_i,S_j} ∩ C_uw
```

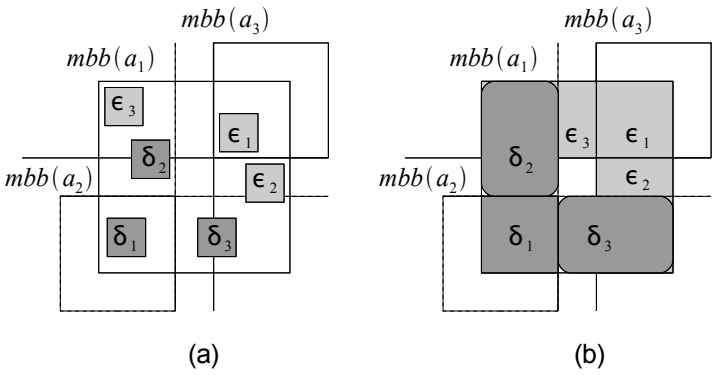
```
            If  Ĉ_{S_i,S_j} = Ø  Then
                Return 'Inconsistent'
            EndIf
        End For each
End For

# Step 3
Replace any remaining {<,=} labels in Ĉ with {<}.
Perform a topological sort of Ĉ, using only the edges that are
labeled with {<}.

# Step 4
Assign strictly increasing values to the output of the previous step.
Note that all variables in the same component must have the same
value.

Return assignment
End
```

Ας δούμε τώρα τι γίνεται στο παράδειγμα που χρησιμοποιούμε. Θυμίζουμε ότι έχουμε $C=\{a_1\,B{:}N{:}E\,a_2,\ a_1\,B{:}S{:}W\,a_3,\ a_2\,SW\,a_3\}$ και $\{d_1, d_2, d_3, e_1, e_2, e_3\}$ είναι συνιστώσες μεταβλητές της a_1 ως προς τον πρώτο και δεύτερο περιορισμό αντίστοιχα. Είχαμε χρησιμοποιήσει τον αλγόριθμο TRANSFORM για να μετατρέψουμε το σύνολο C σε ένα σύνολο περιορισμών διάταξης O. Ο αλγόριθμος CSPAN δεχόμενος σαν είσοδο το σύνολο O, επειδή το σύνολο αυτό είναι συνεπές, αναθέτει στις μεταβλητές $a_1, a_2, a_3, d_1, d_2, d_3, e_1, e_2, e_3$ τα μη τετριμμένα πλαίσια που ονομάζουμε $\alpha_1, \alpha_2, \alpha_3, \delta_1, \delta_2, \delta_3, \epsilon_1, \epsilon_2, \epsilon_3$ αντίστοιχα (όπως φαίνεται στο σχήμα 14(a) που ακολουθεί).



Σχήμα 14: Πριν και μετά την επέκταση της λύσης

3.2.3 Επέκταση λύσης

Όπως βλέπουμε στο σχήμα 14(b), η παραπάνω λύση που βρέθηκε μπορεί να επεκταθεί σε μια μεγιστική λύση. Για παράδειγμα, το πλαίσιο δ_1 μπορεί να επεκταθεί προς τα δυτικά μέχρι να φτάσει την κάθετη γραμμή $y=\inf_x(a_1)$ και προς τα ανατολικά μέχρι να φτάσει την κάθετη γραμμή $y=\sup_x(a_2)$. Αυτή η διαισθητική ιδέα μπορεί να εφαρμοστεί με ανάλογο τρόπο και στις υπόλοιπες περιοχές και διατυπώνεται τυπικά στο ακόλουθο λήμμα.

Λήμμα 1 Έστω s^0 μία λύση του συνόλου περιορισμών διάταξης O, η οποία αναθέτει τα πλαίσια $\alpha_i, \alpha_j, \alpha_{i1}, .., \alpha_{il}$ στις μεταβλητές περιοχής $a_i, a_j, a_{i1}, .., a_{il}$ αντίστοιχα. Τότε μπορεί να προκύψει μια νέα λύση u^0 του O, η οποία μάλιστα είναι μεγιστική με τον εξής τρόπο. Για κάθε μη ατομικό βασικό περιορισμό κατεύθυνσης $a_i R_1 :..: R_k a_j \in C$ και για κάθε συνιστώσα μεταβλητή a_{im} της a_{i}, όπου $1 \le m \le l$, κάνουμε τις παρακάτω αντικαταστάσεις:

- Αν έχουμε $a_{\text{im}} B a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \inf_x(\alpha_j)\}$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \sup_x(\alpha_j)\}$
  $\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \inf_y(\alpha_j)\}$, $\sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \sup_y(\alpha_j)\}$

- Αν έχουμε $a_{\text{im}} S a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \inf_x(\alpha_j)\}$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \sup_x(\alpha_j)\}$
  $\inf_y(\alpha_{\text{im}}) \leftarrow \inf_y(\alpha_i)$, $\sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \inf_y(\alpha_j)\}$

- Αν έχουμε $a_{\text{im}} SW a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow \inf_x(\alpha_i)$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \inf_x(\alpha_j)\}$
  $\inf_y(\alpha_{\text{im}}) \leftarrow \inf_y(\alpha_i)$, $\sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \inf_y(\alpha_j)\}$

- Αν έχουμε $a_{\text{im}} W a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow \inf_x(\alpha_i)$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \inf_x(\alpha_j)\}$
  $\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \inf_y(\alpha_j)\}$, $\sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \sup_y(\alpha_j)\}$

- Αν έχουμε $a_{\text{im}} NW a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow \inf_x(\alpha_i)$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \inf_x(\alpha_j)\}$
  $\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \sup_y(\alpha_j)\}$, $\sup_y(\alpha_{\text{im}}) \leftarrow \sup_y(\alpha_i)$

- Αν έχουμε $a_{\text{im}} N a_j$ τότε αντικαθιστούμε:
  $\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \inf_x(\alpha_j)\}$, $\sup_x(\alpha_{\text{im}}) \leftarrow min\{\sup_x(\alpha_i), \sup_x(\alpha_j)\}$

$$\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \sup_y(\alpha_j)\},\ \ \sup_y(\alpha_{\text{im}}) \leftarrow \sup_y(\alpha_i)$$

- Αν έχουμε $a_{\text{im}}\, NE\, a_j$ τότε αντικαθιστούμε:
  $$\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \sup_x(\alpha_j)\},\ \ \sup_x(\alpha_{\text{im}}) \leftarrow \sup_x(\alpha_i)$$
  $$\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \sup_y(\alpha_j)\},\ \ \sup_y(\alpha_{\text{im}}) \leftarrow \sup_y(\alpha_i)$$

- Αν έχουμε $a_{\text{im}}\, E\, a_j$ τότε αντικαθιστούμε:
  $$\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \sup_x(\alpha_j)\},\ \ \sup_x(\alpha_{\text{im}}) \leftarrow \sup_x(\alpha_i)$$
  $$\inf_y(\alpha_{\text{im}}) \leftarrow max\{\inf_y(\alpha_i), \inf_y(\alpha_j)\},\ \ \sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \sup_y(\alpha_j)\}$$

- Αν έχουμε $a_{\text{im}}\, SE\, a_j$ τότε αντικαθιστούμε:
  $$\inf_x(\alpha_{\text{im}}) \leftarrow max\{\inf_x(\alpha_i), \sup_x(\alpha_j)\},\ \ \sup_x(\alpha_{\text{im}}) \leftarrow \sup_x(\alpha_i)$$
  $$\inf_y(\alpha_{\text{im}}) \leftarrow \inf_y(\alpha_i),\ \ \sup_y(\alpha_{\text{im}}) \leftarrow min\{\sup_y(\alpha_i), \inf_y(\alpha_j)\}$$

3.3 Τρίτο βήμα – Χειρισμός περιορισμών ένωσης

Φτάνοντας στο τρίτο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, πρέπει να θυμίσουμε αυτό που είχε αναφερθεί στην αρχή του κεφαλαίου, δηλαδή ότι ένα σύνολο βασικών περιορισμών κατεύθυνσης C μπορεί να μετατραπεί σε ένα ισοδύναμο ως προς τη συνέπεια σύνολο S, που περιέχει ατομικούς περιορισμούς κατεύθυνσης και περιορισμούς ένωσης. Μέχρι στιγμής ο αλγόριθμος δεν είχε ασχοληθεί καθόλου με το τελευταίο είδος περιορισμών.

Ας υποθέσουμε ότι $C_{a_i}=\{c_1,..,c_m\}$ είναι το σύνολο των μη ατομικών περιορισμών κατεύθυνσης του C που έχουν ως πρωτεύουσα μεταβλητή την a_i. Έστω $S_1=\{a^1_{i1},..,a^1_{ik_1}\},..,S_m=\{a^m_{i1},..,a^m_{ik_m}\}$ είναι τα σύνολα των συνιστωσών μεταβλητών της a_i ως προς τους περιορισμούς $\{c_1,..,c_m\}$ αντίστοιχα. Με Σ_i συμβολίζουμε ένα στιγμιότυπο του συνόλου S_i. Τώρα, το σύνολο S θα πρέπει εξ ορισμού του να περιλαμβάνει τους ακόλουθους περιορισμούς:

$$a_i=a^1_{i1}\cup..\cup a^1_{ik_1}$$
$$\vdots$$
$$a_i=a^m_{i1}\cup..\cup a^m_{ik_m}$$

Έστω μια τυχαία συνιστώσα μεταβλητή $s\in S_1\cup..\cup S_m$. Προφανώς θα πρέπει να υπάρχει μια σχέση της παραπάνω μεταβλητής με οποιαδήποτε συνιστώσα μεταβλητή από τα σύνολα $S_1,..,S_m$. Η σχέση αυτή εκφράζεται στο το ακόλουθο

λήμμα, που διατυπώθηκε και αποδείχτηκε από το Σκιαδόπουλο [SK05].

Λήμμα 2 Έστω περιοχή $a \in REG^*$. Αν $S_1, .., S_m$ είναι σύνολα υποπεριοχών της a τέτοια ώστε να ισχύουν:

1. $(\forall r \in S_i)(r \in REG^*)$
2. $a = \bigcup_{r \in S_1} r = .. = \bigcup_{r \in S_m} r$

Τότε υπάρχει μια πλειάδα $(s_1, .., s_m) \in S_1 \times .. \times S_m$ τέτοια ώστε η να μην είναι τετριμμένο πλαίσιο η τομή $mbb(s) \cap mbb(s_1) \cap .. \cap mbb(s_m)$.

Η παραπάνω συνθήκη, η οποία από εδώ και στο εξής θα αποκαλείται 'περιορισμός NTB', απαιτεί ότι η τομή των παραπάνω $m+1$ πλαισίων να είναι μη τετριμμένο πλαίσιο. Αυτός ο περιορισμός παρατηρούμε ότι μπορεί εύκολα να απεικονιστεί σε περιορισμούς διάταξης. Τούτο προκύπτει από τον ορισμό της τομής για περιοχές του χώρου, αφού αν θεωρήσουμε δύο τυχαίες περιοχές a, b τότε η τομή τους $c = a \cap b$ ορίζεται ως εξής:

$$\inf_x(c) = max\{\inf_x(a), \inf_x(b)\}, \quad \sup_x(c) = min\{\sup_x(a), \sup_x(b)\}$$
$$\inf_y(c) = max\{\inf_y(a), \inf_y(b)\}, \quad \sup_y(c) = min\{\sup_y(a), \sup_y(b)\}$$

και το πλαίσιο c είναι μη τετριμμένο αν και μόνο αν ισχύει:

$$\inf_x(c) < \sup_x(c) \ \ \& \ \ \inf_y(c) < \sup_y(c)$$

Οι παραπάνω παρατηρήσεις οδήγησαν στην ιδέα ότι αφού ούτως ή άλλως θα πρέπει να ικανοποιείται ο περιορισμός NTB για όλες τις μεταβλητές του C που συμμετέχουν ως πρωτεύουσες μεταβλητές σε μη ατομικούς περιορισμούς, αυτό θα πρέπει να συμβαίνει και με τη λύση που προέκυψε από το δεύτερο βήμα. Επιπλέον, ο Σκιαδόπουλος [SK05] έδειξε ότι ικανή και αναγκαία συνθήκη για την ικανοποιησιμότητα του συνόλου C είναι η ύπαρξη μιας μεγιστικής λύσης για το O και η ικανοποιησιμότητα του περιορισμού NTB για αυτήν. Συνεπώς πρέπει και αρκεί για κάθε μεταβλητή περιοχής να εξεταστεί αν, για τα πλαίσια τα οποία ανατέθηκαν από το δεύτερο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, ικανοποιείται ο περιορισμός NTB. Αυτό ακριβώς επιτυγχάνεται από τον αλγόριθμο **C**HECK**C**ONSTRAINT**NTB**, ο οποίος περιγράφεται με τον ψευδοκώδικα που ακολουθεί.

```
Algorithm CHECKCONSTRAINTNTB
Input: Sets of regions Σ1,..,Σm that were assigned to the component
       variables that correspond to a specific region variable
```

```
Output: 'True' if constraint NTB is satisfied, else it returns
        'False'

Begin
For each  s∈Σ_1∪..∪Σ_m  Do
        Q={s}
        For each  Σ′∈{Σ_1,..,Σ_m}  Do
                Q′=Ø
                For each  s′∈Σ′ & q∈Q  Do
                        If  is_non_trivial(s′∩q)  Then  Q′=Q′∪{s′∩q}
                End For each
                If  Q′=Ø  Then Return 'False'
                Q=Q′
        End For each
End For each
Return 'True'
End
```

Το τρίτο βήμα λοιπόν του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, που ονομάζεται **G**LOBAL**C**HECK**C**ONSTRAINT**NTB**, απλώς εκτελεί για όλες τις μεταβλητές περιοχής τον αλγόριθμο **C**HECK**C**ONSTRAINT**NTB**.

```
Algorithm GLOBALCHECKCONSTRAINTNTB
Input: A maximal solution  u^0  of  O
Output: 'True' if constraint NTB is satisfied, else it returns
        'False'

Begin
For each  a∈{a_1,..,a_n}  Do
        Assume that regions  Σ_1,..,Σ_m  were assigned to the component
        variables that correspond to  a  by  u^0
        If CHECKCONSTRAINTNTB( Σ_1,..,Σ_m ) = 'False' Then Return 'False'
End For each
Return 'True'
End
```

3.4 Συνοψίζοντας

3.4.1 Ψευδοκώδικας συνολικού αλγόριθμου

Έχοντας πλέον περιγράψει αναλυτικά όλα τα βήματα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, μπορούμε στη συνέχεια να παρουσιάσουμε τον

ψευδοκώδικα για το συνολικό αλγόριθμο.

```
Algorithm Skiadopoulos-Koubarakis
Input: A set of basic direction constraints C
Output: 'Consistent' if C is consistent, else it returns
'Inconsistent'

Begin
# Step 1
 O = TRANSFORM( C )

# Step 2
 Find a solution s^0 of O using CSPAN
 If CSPAN returns 'Inconsistent' Then Return 'Inconsistent'
 Extend s^0 to a maximal solution u^0

# Step 3
 If GLOBALCHECKCONSTRAINTNTB( u^0 ) returns 'False' Then
      Return 'Inconsistent'
 Else
      Return 'Consistent'
 End If
End
```

3.4.2 Πολυπλοκότητα αλγόριθμου

Εύκολα διαπιστώνουμε ότι ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη [SK05] έχει χρονική πολυπλοκότητα $O(n^5)$. Αυτό αποδεικνύεται με τον εξής τρόπο. Έστω ότι n είναι το πλήθος των μεταβλητών περιοχής, οπότε το πλήθος των περιορισμών στο C είναι $O(n^2)$.

Το πρώτο βήμα, δηλαδή ο αλγόριθμος TRANSFORM, απαιτεί χρόνο $O(n^2)$ αφού για κάθε περιορισμό επιτελεί λειτουργίες σταθερού χρόνου. Συγκεκριμένα παράγει το πολύ 9 νέες μεταβλητές στο πρώτο του βήμα, επομένως ο συνολικός αριθμός μεταβλητών περιοχής είναι $O(n^2)$. Επίσης, ο αριθμός των περιορισμών διάταξης και των μεταβλητών διάταξης είναι $O(n^2)$.

Το δεύτερο βήμα χωρίζεται σε δύο στάδια, την εύρεση και την επέκταση της λύσης. Η εύρεση γίνεται με τον αλγόριθμο CSPAN, ο οποίος όπως είχαμε αναφέρει νωρίτερα απαιτεί χρόνο $O(n^2)$. Αυτό όμως συμβαίνει όταν δέχεται είσοδο ένα σύνολο περιορισμών διάταξης με n μεταβλητές. Στην περίπτωσή μας έχουμε $O(n^2)$ μεταβλητές, οπότε ο χρόνος που απαιτείται είναι $O(n^4)$. Η επέκταση της λύσης, χρησιμοποιώντας τις αντικαταστάσεις του λήμματος 1, απαιτεί χρόνο $O(n^2)$ αφού

αυτές οι αντικαταστάσεις είναι σταθερού χρόνου και επιτελούνται σε $O(n^2)$ πλήθος μεταβλητών. Συνεπώς ο χρόνος που απαιτείται για ολόκληρο το δεύτερο βήμα είναι $O(n^4)$.

Το τρίτο βήμα, αποτελείται από τον **G**LOBAL**C**HECK**C**ONSTRAINT**NTB** ο οποίος προφανώς καλεί $O(n)$ φορές τον **C**HECK**C**ONSTRAINT**NTB**. Ο τελευταίος από ότι διαπιστώνουμε ουσιαστικά αποτελείται από πέντε(!) nested loops (γεγονός που φαίνεται 'ξεδιπλώνοντας' το πρώτο και τελευταίο 'διπλό' loop) και θα δείξουμε ότι απαιτεί χρόνο $O(n^4)$. Το πρώτο 'διπλό' loop εκτελείται $O(n)$ φορές, αφού κάθε σύνολο Σ έχει το πολύ 9 στοιχεία, διότι σε κάθε περιορισμό μπορούν το πολύ 9 συνιστώσες μεταβλητές να αντιστοιχούν σε μια πρωτεύουσα μεταβλητή και το πλήθος των συνόλων Σ είναι το πολύ n. Για τον ίδιο λόγο και το δεύτερο loop εκτελείται επίσης $O(n)$ φορές. Το τελευταίο 'διπλό' loop εκτελείται $O(n^2)$ φορές, δηλαδή όσα και τα στοιχεία του συνόλου Q. Το σύνολο Q έχει $O(n^2)$ στοιχεία γιατί κάθε πρωτεύουσα μεταβλητή διαμερίζεται με το πολύ n τρόπους (δηλ. όσα και τα σύνολα Σ) και για κάθε μία τέτοια διαμέριση εξετάζεται κάθε κομμάτι αυτής με τα κομμάτια όλων των υπολοίπων (που είναι προφανώς $O(n)$). Άρα το πλήθος των στοιχείων του Q είναι $O(n^2)$. Επίσης όλες οι υπόλοιπες λειτουργίες (ανάθεση, ένωση, έλεγχος κατηγορήματος 'is_non_trivial'), είναι προφανώς σταθερού χρόνου. Σύμφωνα με τα παραπάνω, έπεται ότι το τρίτο βήμα απαιτεί χρόνο $O(n^5)$.

Ολοκληρώνοντας αυτή την παρουσίαση του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, θα πρέπει να σημειώσουμε ότι το τελευταίο βήμα του αλγορίθμου είναι εκείνο που καθορίζει τη συνολική πολυπλοκότητα, η οποία είναι $O(n^5)$.

4. Ο αλγόριθμος Navarrete-Morales-Sciavicco

4.1 Γενική περιγραφή

Σε αυτό το κεφάλαιο γίνεται παρουσίαση του αλγορίθμου των Navarrete-Morales-Sciavicco. Ο συγκεκριμένος αλγόριθμος ουσιαστικά αποτελεί μια παραλλαγή-βελτίωση του προηγούμενου. Το πλεονέκτημα αυτού του αλγορίθμου είναι ότι η χρονική πολυπλοκότητά του είναι $O(n^4)$. Το μειονέκτημά του είναι ότι δεν δουλεύει για μη συνεκτικές περιοχές. Συνεπώς οι μεταβλητές σε αυτή την περίπτωση περιορίζονται στο REG. Ένα άλλο μειονέκτημά του είναι ότι απαιτεί πολύ περισσότερο χώρο στη μνήμη από ότι ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη.

Προκειμένου να αποφανθεί για την συνέπεια ενός συνόλου περιορισμών κατεύθυνσης στο οποίο οι μεταβλητές παίρνουν τιμές από το REG, ο αλγόριθμος Navarrete-Morales-Sciavicco χρησιμοποιεί αυτούσιο το πρώτο και το δεύτερο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη. Πριν από αυτά τα βήματα, εκτελεί ένα βήμα προ επεξεργασίας, με το οποίο μπορεί να ανιχνεύσει κάποιες περιπτώσεις ασυνέπειας. Μετά από αυτά τα βήματα, χρησιμοποιεί μια παραλλαγή του τρίτου βήματος του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη. Όλα αυτά επιτυγχάνονται λόγω του γνωστού θεωρήματος του Helly από τη συνδυαστική γεωμετρία.

Θεώρημα Helly Αν F είναι μια πεπερασμένη οικογένεια από περισσότερα από n φραγμένα και κλειστά σύνολα στον χώρο $\mathbb{R}^n$ και αν κάθε H_n (όπου $H_n = n+1$ είναι ο λεγόμενος αριθμός του Helly) μέλη της F έχουν τουλάχιστον ένα κοινό σημείο, τότε όλα τα μέλη της F έχουν τουλάχιστον ένα κοινό σημείο, δηλαδή ισχύει ότι $\bigcap F \neq \emptyset$.

Μια συνηθισμένη προσέγγιση στην επίλυση προβλημάτων ικανοποίησης περιορισμών είναι η χρήση της σύνθεσης σε διάφορους αλγορίθμους που χρησιμοποιούνται για την διάδοση περιορισμών (π.χ. συνέπεια μονοπατιού). Όπως δείξαμε παραπάνω, στην περίπτωση των περιορισμών κατεύθυνσης, μόνο για την πράξη της αδύναμης σύνθεσης υπάρχουν αλγόριθμοι υπολογισμού της. Συνεπώς μπορούμε να χρησιμοποιήσουμε μια πιο αδύναμη έκδοση της συνέπειας μονοπατιού την οποία ονομάζουμε αλγεβρική κλειστότητα, αντικαθιστώντας απλώς τη σύνθεση με την αδύναμη σύνθεση. Το πρώτο βήμα λοιπόν του αλγορίθμου Navarrete-Morales-Sciavicco είναι η εφαρμογή του αλγορίθμου αλγεβρικής κλειστότητας στο σύνολο των περιορισμών κατεύθυνσης. Προφανώς ένα συνεπές σύνολο περιορισμών είναι αλγεβρικά κλειστό. Επομένως, αν ένα σύνολο περιορισμών δεν είναι αλγεβρικά κλειστό τότε δεν μπορεί να είναι συνεπές. Δυστυχώς η αλγεβρική κλειστότητα δεν είναι ισοδύναμη με την συνέπεια. Για παράδειγμα, το σύνολο $\{$ *a B:SW:W:N:NE b, a B:SW:W:E:SE c, a B:SW:W:E:SE d, b B:SW:W:W:E:SE c, b S:SW d, d B:W:NW:N:NE:E c* $\}$ είναι αλγεβρικά κλειστό αλλά

δεν είναι συνεπές. Παρόλα αυτά η αλγεβρική κλειστότητα εξακολουθεί να είναι μια αρκετά χρήσιμη έννοια, αφού εκτός από τις διάφορες περιπτώσεις ασυνέπειας που ανιχνεύει, χρησιμοποιείται και στο ακόλουθο λήμμα το οποίο είναι ιδιαίτερα σημαντικό για τη συνέχεια. Συγκεκριμένα το λήμμα αυτό χρησιμοποιείται για την απόδειξη του θεωρήματος που ακολουθεί. Το τελευταίο με τη σειρά του, μας παρέχει ουσιαστικά τον τρόπο με τον οποίο πρέπει να αλλάξει το τρίτο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη προκειμένου η χρονική του πολυπλοκότητα να μειωθεί από $O(n^5)$ σε $O(n^4)$.

Λήμμα Αν ένα σύνολο C με περιορισμούς στο D είναι αλγεβρικά κλειστό, τότε για κάθε μεταβλητή a και για κάθε $B \subseteq \{\Sigma_1, .., \Sigma_\mu\}$ με $|B|=2$ ισχύει ότι $\cap B \in REG$.

Θεώρημα Έστω ένα αλγεβρικά κλειστό σύνολο C με περιορισμούς στο D, a μια μεταβλητή περιοχής που εμφανίζεται στο C και $\{\Sigma_1, .., \Sigma_\mu\}$ το σύνολο των αναθέσεων στις συνιστώσες μεταβλητές του a, στις οποίες έγινε η ανάθεση κατά το δεύτερο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη. Αν για κάθε υποσύνολο $B \subseteq \{\Sigma_1, .., \Sigma_\mu\}$ με $|B|=3$ ικανοποιείται ο περιορισμός NTB, τότε ο περιορισμός NTB ικανοποιείται για ολόκληρο το $\{\Sigma_1, .., \Sigma_\mu\}$.

Σύμφωνα με το παραπάνω θεώρημα, δεν χρειάζεται πλέον να εξετάζεται αν για κάθε μεταβλητή όλα τα σύνολα Σ ικανοποιούν ταυτόχρονα τον περιορισμό NTB, αλλά αρκεί ο τελευταίος να ικανοποιείται από κάθε δυνατή τριάδα αυτών των συνόλων. Αυτές οι τριάδες μπορούν εύκολα να απαριθμηθούν και προφανώς υπάρχουν $\binom{m}{3}$ τέτοιες τριάδες για κάθε μεταβλητή a, όπου m το πλήθος των μη ατομικών περιορισμών κατεύθυνσης στους οποίους εμφανίζεται η a ως πρωτεύουσα μεταβλητή.

4.2 Ψευδοκώδικας αλγόριθμου

Πριν δώσουμε τον ψευδοκώδικα για ολόκληρο τον αλγόριθμο Navarrete-Morales-Sciavicco, θα πρέπει να δώσουμε πρώτα τον ψευδοκώδικα για το βήμα προ επεξεργασίας (αλγεβρική κλειστότητα) καθώς επίσης και το τροποποιημένο, λόγω του προηγούμενου θεωρήματος, τρίτο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη.

Ο αλγόριθμος αλγεβρικής κλειστότητας που χρησιμοποιείται, είναι μια μικρή παραλλαγή του κλασικού αλγορίθμου συνέπειας μονοπατιού του Allen, όπως αναφέρεται στο [VB96], για περιορισμούς της άλγεβρας διαστημάτων, όπου χρησιμοποιείται αδύναμη σύνθεση στη θέση της σύνθεσης.

```
Algorithm ALGEBRAICALLY_CLOSED
Input: A set of basic direction constraints C
```

Output: 'True' if C is algebraically closed, else it returns 'False'

Begin
Build the corresponding graph of C using the adjacency matrix representation. Lets assume that n region variables exist in C.

$L=\{(i,j)|1\le i<j\le n\}$
While L is not empty **Do**
select and delete an element (i,j) of L
For each $k\in\{1,..,n\}\setminus\{i,j\}$ **Do**
$t=C_{ik}\cap(C_{ij}*C_{jk})$
If $t=\emptyset$ **Then Return** 'False'
If $t\neq C_{ik}$ **Then**
$C_{ik}=t$
$C_{ki}=Inv(t)$
$L=L\cup\{(i,k)\}$
EndIf
$t=C_{kj}\cap(C_{ki}*C_{ij})$
If $t=\emptyset$ **Then Return** 'False'
If $t\neq C_{kj}$ **Then**
$C_{kj}=t$
$C_{jk}=Inv(t)$
$L=L\cup\{(k,j)\}$
EndIf
End For each
End While
Return 'True'
End

Το τρίτο βήμα του αλγορίθμου Navarrete-Morales-Sciavicco το οποίο ονομάζουμε NMS_**G**LOBAL**C**HECK**C**ONSTRAINT**NTB**, απλά καλεί τον αλγόριθμο **C**HECK**C**ONSTRAINT**NTB**, όχι για όλα τα $\sum$ σύνολα όπως κάνει ο αλγόριθμος **G**LOBAL**C**HECK**C**ONSTRAINT**NTB**, αλλά για κάθε δυνατή τριάδα αυτών.

Algorithm NMS_**G**LOBAL**C**HECK**C**ONSTRAINT**NTB**
Input: A maximal solution u^0 of O
Output: 'True' if constraint NTB is satisfied, else it returns 'False'

Begin

```
For each a∈{a_1,..,a_n} Do
      Assume that regions Σ_1,..,Σ_m were assigned to the component
      variables that correspond to a by u^0
      Enumerate every possible triad {Σ_i,Σ_j,Σ_k}⊆{Σ_1,..,Σ_m}
      For each {Σ_i,Σ_j,Σ_k}⊆{Σ_1,..,Σ_m} Do
            If CHECKCONSTRAINTNTB( Σ_i,Σ_j,Σ_k ) = 'False'
                  Then Return 'False'
      End For each
End For each
Return 'True'
End
```

Μετά από αυτά, μπορούμε πλέον να δώσουμε τον ψευδοκώδικα για τον συνολικό αλγόριθμο.

```
Algorithm Navarrete-Morales-Sciavicco
Input: A set of basic direction constraints C
Output: 'Consistent' if C is consistent, else it returns
'Inconsistent'

Begin
# Step 0
If ALGEBRAICALLY_CLOSED( C ) returns 'False' Then
      Return 'Inconsistent'

# Step 1
Same as Skiadopoulos-Koubarakis

# Step 2
Same as Skiadopoulos-Koubarakis

# Step 3
Same as Skiadopoulos-Koubarakis, using NMS_GLOBALCHECKCONSTRAINTNTB
instead of GLOBALCHECKCONSTRAINTNTB.
End
```

4.3 Πολυπλοκότητα αλγόριθμου

Όπως αναφέραμε και στην αρχή, ο αλγόριθμος Navarrete-Morales-Sciavicco έχει χρονική πολυπλοκότητά $O(n^4)$. Το βήμα προ επεξεργασίας λόγω της πολυπλοκότητας του γνωστού αλγορίθμου συνέπειας μονοπατιού και του γεγονότος ότι όλοι οι περιορισμοί κατεύθυνσης είναι βασικοί, θέλει χρόνο $O(n^3)$. Το τελευταίο βήμα εύκολα μπορούμε να διαπιστώσουμε ότι απαιτεί χρόνο $O(n^4)$,

αφού ο αλγόριθμος NMS_**G**LOBAL**C**HECK**C**ONSTRAINT**NTB** καλεί τον αλγόριθμο **C**HECK**C**ONSTRAINT**NTB** $O(n^4)$ φορές και ο τελευταίος δουλεύει πλέον σε σταθερό χρόνο.

Σε αυτό το σημείο ολοκληρώνεται η παρουσίαση του αλγορίθμου Navarrete-Morales-Sciavicco.

5. Υλοποίηση των αλγόριθμων

Σε αυτό το κεφάλαιο θα ασχοληθούμε με την υλοποίηση των δύο αλγόριθμων που παρουσιάστηκαν στα προηγούμενα κεφάλαια. Πριν προχωρήσουμε στην περιγραφή των διαφόρων λειτουργικών μονάδων (συναρτήσεων και κλάσεων) που εμφανίζονται στα αρχεία πηγαίου κώδικα, θα πρέπει να αναφέρουμε κάποιες γενικές αρχές, βάση των οποίων δημιουργήθηκε αυτή η υλοποίηση.

5.1 Γενικές αρχές δημιουργίας λογισμικού

- Σταθερότητα: Το λογισμικό θα πρέπει να είναι όσο το δυνατόν περισσότερο σταθερό και αξιόπιστο, δηλαδή θα πρέπει να μην υπάρχουν σφάλματα τα οποία μπορεί να το οδηγήσουν σε κατάρρευση (crash) ή κρέμασμα (hang).

- Απλότητα: Το λογισμικό θα πρέπει να διατηρείται όσο το δυνατόν πιο απλό. Ο στόχος μας είναι ο κώδικας που δημιουργήθηκε να μην είναι ιδιαίτερα πολύπλοκος, αφού η επέμβαση εκ των υστέρων σε ένα τέτοιο κώδικα είναι από εξαιρετικά δύσκολη έως αδύνατη (προγραμματισμός σπαγγέτι). Συνεπώς θα έπρεπε να αποφύγουμε τη χρήση αδικαιολόγητα πολύπλοκων δομών δεδομένων, την επανάληψη του κώδικα (bloat) καθώς επίσης και τις περιττές μετατροπές ανάμεσα στις διάφορες δομές δεδομένων. Τα παραπάνω ήταν αρκετά δύσκολο να επιτευχθούν, αφού όπως είδαμε και στην περιγραφή των αλγορίθμων, η διεπαφή μεταξύ των διαφόρων βημάτων δεν ήταν πάντα ξεκάθαρη. Όπως μπορεί να διαπιστώσει κανείς εξετάζοντας τον κώδικα, υπάρχουν κάποιες μετατροπές που είναι αναγκαίες, αλλά ο ψευδοκώδικας είναι αρκετά υψηλού επιπέδου (όπως θα έπρεπε άλλωστε) και τις κρύβει επιμελώς κάτω από το χαλί. Πιστεύουμε ότι σε μεγάλο βαθμό έχουν επιτευχθεί αυτοί οι στόχοι, πράγμα το οποίο επιβεβαιώνεται από το γεγονός ότι ο ψευδοκώδικας και ο πραγματικός κώδικας μοιάζουν αρκετά.

- Μεταφερσιμότητα: Το λογισμικό θα πρέπει να μπορεί να μεταγλωττιστεί και να εκτελεστεί σε ένα μεγάλο αριθμό από πλατφόρμες (συνδυασμό αρχιτεκτονικής επεξεργαστή και λειτουργικού συστήματος). Για την υλοποίηση επιλέξαμε ως γλώσσα προγραμματισμού τη C++. Η C++ είναι μία από τις πλέον δημοφιλείς γλώσσες προγραμματισμού και χρησιμοποιείται ευρέως για τη δημιουργία διαφόρων ειδών προγραμμάτων. Η ταχύτητά της και η δυνατότητες που παρέχει για προγραμματισμό χαμηλού επιπέδου (bitwise manipulation) ήταν από τα σημαντικότερα κριτήρια που οδήγησαν στην επιλογή της. Συγκεκριμένα χρησιμοποιήσαμε τον μεταγλωττιστή GNU g++, ο οποίος διατίθεται για σχεδόν όλες τις υπολογιστικές πλατφόρμες που υπάρχουν. Το γεγονός ότι δεν χρησιμοποιείται το API κανενός λειτουργικού συστήματος (εκτός από ένα ελάχιστο μέρος που χρησιμοποιεί κάποιες POSIX επεκτάσεις και απομονώνεται από τον υπόλοιπο κώδικα, με τη χρήση μεταγλώττισης υπό συνθήκες) και καμία μη μεταφέρσιμη βιβλιοθήκη, μεγιστοποιεί την μεταφερσιμότητα του προγράμματος. Πρόκειται δηλαδή

για ένα πλήρως ISO C++ πρόγραμμα του οποίου η μόνη εξάρτηση είναι η βιβλιοθήκη BOOST (http://www.boost.org) και μάλιστα μόνο το μέρος που έχει σχέση με τα γραφήματα (για την υλοποίηση του αλγορίθμου `CSPAN`).

5.2 Περιγραφή λειτουργικών μονάδων

Σε αυτή την ενότητα θα περιγράψουμε τις δομές δεδομένων (κυρίως κλάσεις) και τις συναρτήσεις που χρησιμοποιούνται από το πρόγραμμα. Η περιγραφή θα γίνεται με βάση τα αρχεία του πηγαίου κώδικα που υπάρχουν, αφού με αυτό τον τρόπο διευκολύνεται η κατανόηση του προγράμματος.

prof.h

Σε αυτό το αρχείο υπάρχει κώδικας που χρησιμοποιείται για την μέτρηση του χρόνου που απαιτείται μεταξύ των διαφόρων τμημάτων των αλγορίθμων. Για να μετρήσουμε το χρόνο μπορούμε, μεταγλωττίζοντας υπό συνθήκη (HAVE_POSIX), να χρησιμοποιήσουμε δύο δυνατές προσεγγίσεις, οι οποίες έχουν κάποια πλεονεκτήματα αλλά και μειονεκτήματα:

Standard βιβλιοθήκη της C++
+ Μεταφερσιμότητα (είναι μέρος της γλώσσας και υπάρχει παντού)
+ Μετράει χρόνο CPU
- Έχει μικρή ακρίβεια (πολύ μικρά instances που τρέχουν σε λιγότερο από 1 δευτερόλεπτο, πολλές φορές εμφανίζεται ότι έχουν μηδενικό χρόνο εκτέλεσης)

POSIX βιβλιοθήκη
+ Έχει πολύ μεγάλη ακρίβεια (μετράει σε επίπεδο microsecond ή ακόμα και nanosecond ανάλογα με την συνάρτηση που καλείται και την πλατφόρμα)
- Μετράει πραγματικό χρόνο (άρα οι μετρήσεις θα πρέπει να γίνονται σε idle σύστημα)
- Δεν υπάρχει σε μη POSIX συστήματα (π.χ. windows)

Η επιλογή γίνεται κατά το χρόνο μεταγλώττισης χρησιμοποιώντας τα αρχεία Makefile.NOPOSIX και Makefile αντίστοιχα. Για παράδειγμα με την εντολή:

make -f Makefile (ή με σκέτο make)

μεταγλωττίζεται το πρόγραμμα χρησιμοποιώντας τη δεύτερη προσέγγιση.

Επιστρέφοντας στην περιγραφή του αρχείου **prof.h,** θα πρέπει να αναφέρουμε ότι ουσιαστικά ορίζονται κάποιοι thin wrappers για τις δύο παραπάνω προσεγγίσεις. Έχουμε ορίσει τον τύπο δεδομένων `time_unit` του οποίου οι μεταβλητές αναπαριστούν κάποια χρονική στιγμή. Με την συνάρτηση `time_measure()` αποθηκεύουμε τη χρονική στιγμή κατά την οποία γίνεται η κλήση αυτής της συνάρτησης. Με τη συνάρτηση `time_diff()` υπολογίζουμε σε δευτερόλεπτα τη διαφορά μεταξύ δύο χρονικών στιγμών.

glob_defs.h

Σε αυτό το αρχείο ορίζουμε τις συμβολικές σταθερές που χρησιμοποιούνται σε ολόκληρο το πρόγραμμα. Όλα τα υπόλοιπα αρχεία θα πρέπει να περιλαμβάνουν αυτό το αρχείο κεφαλίδας. Χρησιμοποιούμε τη γνωστή διάταξη των τμημάτων του επιπέδου, όπως αυτή εμφανίζεται στη βιβλιογραφία και απεικονίζεται στον παρακάτω πίνακα:

SE	E	NE	N	NW	W	SW	S	B
8	7	6	5	4	3	2	1	0

Η αναπαράσταση μιας σχέσης σαν ένα διάνυσμα από bits, όπως αυτό του παραπάνω πίνακα, μας εξασφαλίζει υψηλή ταχύτητα επεξεργασίας για τις διάφορες συνολοθεωρητικές πράξεις. Για παράδειγμα ο έλεγχος για το εάν κάποιο τμήμα του επιπέδου ανήκει σε μια σχέση, γίνεται γρήγορα εκτελώντας διάζευξη με την αντίστοιχη μάσκα και στη συνέχεια ελέγχοντας αν το αποτέλεσμα ταυτίζεται με τη σχέση. Επίσης, με την εκτέλεση διάζευξης και σύζευξης μεταξύ δύο διανυσμάτων από bits παίρνουμε αντίστοιχα την ένωση και την τομή των δύο σχέσεων.

orderCon.h & orderCon.cpp

Σε αυτά τα αρχεία ορίζεται η κλάση των περιορισμών διάταξης `OrderCon` καθώς και οι σχετικές με αυτή μέθοδοι. Από ότι βλέπουμε, ένας περιορισμός διάταξης αποτελείται από δύο συμβολοσειρές που αναπαριστούν τις μεταβλητές του περιορισμού, καθώς επίσης και τη σχέση διάταξης που πρέπει να ικανοποιείται μεταξύ αυτών. Η σχέση διάταξης είναι οποιοδήποτε υποσύνολο του $\{<, =, >\}$. Εκτός από τις αναμενόμενες μεθόδους με τις οποίες αποκτούμε πρόσβαση στα μέλη της κλάσης, έχουμε ορίσει τους γνωστούς τελεστές της C++ για είσοδο (>>) και έξοδο (<<) για αυτό το τύπο δεδομένων. Επίσης ορίζουμε ως λύση, το ζεύγος:
(όνομα μεταβλητής, τιμή μεταβλητής)

basicRel.h & basicRel.cpp

Εδώ έχουμε τον ορισμό της κλάσης των βασικών σχέσεων κατεύθυνσης, η οποία ονομάζεται `BasicRel`. Όπως αναφέραμε και πιο πριν, θεωρούμε ότι μια βασική σχέση κατεύθυνσης είναι ένα διάνυσμα από 9 bits, συνεπώς με τη χρήση δημόσιας κληρονομικότητας έχουμε μια is-a σχέση. Ακόμα, έχουν οριστεί οι αναμενόμενες συνολοθεωρητικές σχέσεις και πράξεις: `includes()`, `isIncluded()`, `setUnion()`, `setInter()`. Επιπλέον, έχουμε ορίσει τους τελεστές εισόδου-εξόδου, αφού θέλουμε η επικοινωνία με το υπόλοιπο πρόγραμμα να γίνεται σε επίπεδο tiles και όχι σε επίπεδο bits.

basiCon.h & basicCon.cpp

Σε αυτά τα αρχεία ορίζεται η κλάση των βασικών περιορισμών κατεύθυνσης `BasicCon` καθώς και οι σχετικές με αυτή μέθοδοι. Προφανώς ένας βασικός περιορισμός κατεύθυνσης αποτελείται από δύο συμβολοσειρές που αναπαριστούν τις μεταβλητές περιοχής, καθώς επίσης και τη βασική σχέση κατεύθυνσης που θα πρέπει να ικανοποιείται μεταξύ αυτών. Όπως και πριν, ορίσαμε τους γνωστούς τελεστές εισόδου-εξόδου και τις μεθόδους με τις οποίες αποκτούμε πρόσβαση στα μέλη της κλάσης. Η πιο ενδιαφέρουσα μέθοδος σε αυτή την κλάση είναι η `mapToOrderCon()` με την οποία μετατρέπουμε μια ατομική σχέση κατεύθυνσης σε ένα ισοδύναμο σύνολο περιορισμών διάταξης (χρησιμοποιώντας τους γνωστούς ορισμούς των ατομικών σχέσεων). Για να επιτευχθεί αυτό, χρησιμοποιείται η βοηθητική συνάρτηση `bStr()` με την οποία γίνεται η κωδικοποίηση μιας μεταβλητής περιοχής σε μια μεταβλητή διάταξης. Με αυτό το τρόπο διευκολύνεται σημαντικά ο προγραμματισμός και ο κώδικας γίνεται πιο κατανοητός.

algoTransform.cpp

Εδώ βρίσκεται το πρώτο βήμα του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, ο αλγόριθμος `TRANSFORM`. Για προγραμματιστική διευκόλυνση, ο αλγόριθμος υλοποιείται σε δύο μέρη και συγκεκριμένα αποτελείται από τι συναρτήσεις `transform()` και `transformBody()`. Η πρώτη απλώς καλεί τη δεύτερη για κάθε περιορισμό κατεύθυνσης. Η δεύτερη συνάρτηση δεν χρειάζεται ιδιαίτερο σχολιασμό, αφού απλά εκτελεί τα 4 βήματα όπως ακριβώς περιγράφονται στον ψευδοκώδικα. Αυτό που ίσως αξίζει να παρατηρήσει κανείς, είναι το ότι ο κώδικας μοιάζει πολύ με τον ψευδοκώδικα, γεγονός που οφείλεται κυρίως στην κωδικοποίηση που γίνεται από τη βοηθητική συνάρτηση `bStr()`.

algoCspan.h & algoCspan.cpp

Σε αυτά τα αρχεία βρίσκονται οι συναρτήσεις που έχουν να κάνουν με τον αλγόριθμο `CSPAN`. Όπως αναφέραμε νωρίτερα, χρησιμοποιήσαμε την βιβλιοθήκη BOOST για τον χειρισμό των γραφημάτων. Η βιβλιοθήκη αυτή παρέχει τις κατάλληλες δομές δεδομένων και τις συναρτήσεις που υλοποιούν τους γνωστούς αλγορίθμους από τη θεωρία γραφημάτων, όπως είναι η τοπολογική διάταξη και η εύρεση ισχυρά συνεκτικών συνιστωσών. Όπως και πριν, για ευκολία χωρίσαμε τον αλγόριθμο σε δύο τμήματα, στις συναρτήσεις `Cspan()` και `CspanCore()`. Η πρώτη συνάρτηση, παίρνει σαν είσοδο ένα σύνολο περιορισμών διάταξης, δημιουργεί το γράφημα που του αντιστοιχεί και στη συνέχεια καλεί τη δεύτερη συνάρτηση. Η τελευταία συνάρτηση είναι ουσιαστικά η υλοποίηση του `CSPAN`.
Αξίζει να παρατηρήσει κανείς, κάποιες απλοποιήσεις που έχουν γίνει στον αλγόριθμο, λόγω της χρήσης της βιβλιοθήκης BOOST και του γεγονότος ότι οι σχέσεις διάταξης που προκύπτουν από τον αλγόριθμο `TRANSFORM`, περιορίζονται σε κάποια από τις $\{<, \leq\}$. Συγκεκριμένα, μετά την εύρεση των ισχυρά συνεκτικών συνιστωσών, μπορούμε να διακρίνουμε δύο περιπτώσεις. Στην περίπτωση που ο αριθμός των ισχυρά συνεκτικών συνιστωσών ταυτίζεται με τον αριθμό των κορυφών του γραφήματος, πράγμα που συμβαίνει όταν αυτό είναι ακυκλικό, τότε δεν χρειάζεται να δημιουργηθεί το συνεπτυγμένο γράφημα, αφού η τοπολογική

διάταξη μπορεί να εκτελεστεί στο αρχικό γράφημα. Στην αντίθετη περίπτωση συνεχίζουμε κανονικά με την κατασκευή του τελευταίου. Και σε αυτή την περίπτωση μπορεί να γίνει η εξής απλοποίηση. Η βιβλιοθήκη μας επιτρέπει να διατρέξουμε τις ακμές και να εξετάσουμε το τι συμβαίνει με τα άκρα τους. Αν τα άκρα μιας ακμής ανήκουν στην ίδια ισχυρά συνεκτική συνιστώσα, τότε θα πρέπει η σχέση μεταξύ τους να μην είναι η $<$, αφού τότε προκύπτει ασυνέπεια όπως είδαμε και στην ανάλυση του `CSPAN`. Αν τα άκρα είναι σε διαφορετικές συνιστώσες, τότε απλά προσθέτουμε την αντίστοιχη ακμή στο συνεπτυγμένο γράφημα.
Η ανάθεση των τιμών στην τελική φάση, γίνεται χρησιμοποιώντας τη βοηθητική συνάρτηση `getValue()`, η οποία σε κάθε κλήση της απλά αυξάνει την τιμή μιας στατικής μεταβλητής.

region.h & region.cpp

Σε αυτά τα αρχεία υπάρχουν οι ορισμοί που έχουν σχέση με τις περιοχές του χώρου. Συγκεκριμένα, ορίζεται μια απλή δομή `mbb` που αναπαριστά το ελάχιστο πλαίσιο το οποίο περικλείει μια περιοχή. Επίσης, ορίζονται οι συναρτήσεις `isNonTrivial()` και `intersection()`. Η πρώτη συνάρτηση χρησιμοποιείται σαν ένα κατηγόρημα με το οποίο μπορούμε να ελέγξουμε αν ένα mbb είναι τετριμμένο, ενώ με τη δεύτερη υπολογίζουμε την τομή δύο πλαισίων. Η τελευταία συνάρτηση εδώ λέγεται `dumpSol()` και απλά τυπώνει στη στάνταρ έξοδο τις περιοχές που έχουν ανατεθεί στις μεταβλητές περιοχής ενός τέτοιου σύνολου.

algoExtend.h & algoExtend.cpp

Εδώ υπάρχουν οι σχετικές συναρτήσεις με την επέκταση της λύσης που βρέθηκε από τον αλγόριθμο `CSPAN`. Οι βασικές συναρτήσεις εδώ ονομάζονται `buildSol()` και `extendSol()`. Η πρώτη συνάρτηση 'χτίζει' στην κατάλληλη μορφή τη λύση του προηγούμενου βήματος. Λέμε ότι 'χτίζει' επειδή ο `CSPAN` αναθέτει για παράδειγμα μια αριθμητική τιμή στην μεταβλητή infx(alpha). Συνεπώς θα πρέπει να συλλέξουμε τις μεταβλητές της μορφής $bound$axis(alpha), όπου $bound είναι inf ή sup και $axis είναι x ή y, ώστε να αντιμετωπίζουμε ως μια οντότητα την περιοχή που ανατέθηκε στην μεταβλητή alpha. Βέβαια, αυτό θα πρέπει να γίνει για όλες τις μεταβλητές περιοχής. Προκειμένου να συμβεί αυτό, χρησιμοποιείται η βοηθητική συνάρτηση `analyseVar()` που κάνει αυτού του είδους την αποκωδικοποίηση.
Στη συνέχεια η συνάρτηση `extendSol()` πριν επεκτείνει, διαμερίζει τη λύση σε δύο σύνολα. Το πρώτο σύνολο περιέχει τις περιοχές που ανατέθηκαν στις συνιστώσες μεταβλητές που υπάρχουν, ενώ το δεύτερο περιέχει αυτές που ανατέθηκαν στις μη συνιστώσες μεταβλητές. Αυτή η διαμέριση έγινε για διάφορους λόγους. Καταρχήν στην περίπτωση που το αρχικό σύνολο των περιορισμών κατεύθυνσης είναι συνεπές, θέλουμε να δώσουμε στο χρήστη, αν βέβαια εκείνος το επιθυμεί, μια λύση την οποία βρήκαμε. Δεύτερον, επειδή οι γνωστές αντικαταστάσεις που γίνονται για να επεκταθεί η λύση, εκτελούνται μόνο στις περιοχές που έχουν ανατεθεί στις συνιστώσες μεταβλητές, είναι πιο αποδοτικό να εξετάζονται μόνο περιοχές αυτού του είδους, από ότι να εξετάζονται όλες. Πέρα από αυτά, θα γίνουν και κάποιες ακόμα μετατροπές στο επόμενο βήμα που είναι απαραίτητες, από τις οποίες θα

φανεί η χρησιμότητα αυτής της διαμέρισης.

algoNTB.h & algoNTB.cpp

Σε αυτά τα αρχεία υπάρχουν οι ορισμοί των δομών δεδομένων και συναρτήσεων, οι οποίες από ότι καταλαβαίνει κανείς έχουν να κάνουν με τον λεγόμενο περιορισμό NTB, δηλαδή με το τρίτο βήμα των αλγορίθμων Σκιαδόπουλου-Κουμπαράκη και Navarrete-Morales-Sciavicco.
Ας περιγράψουμε πρώτα τις δομές δεδομένων που υπάρχουν. Καταρχήν θα πρέπει να θυμηθούμε ότι η είσοδος του αλγορίθμου `CHECKCONSTRAINTNTB` είναι τα λεγόμενα Σ σύνολα. Αυτά αναπαριστώνται από την δομή `SigmaSet`. Ένα Σ σύνολο θα πρέπει να έχει προφανώς ένα όνομα και να περιέχει τα πλαίσια που έχουν ανατεθεί στις αντίστοιχες συνιστώσες μεταβλητές. Λόγω της συνδυαστικής φύσης του προβλήματος και του τρόπου με τον οποίο δουλεύει ο προηγούμενος αλγόριθμος, δεν είναι απαραίτητο να γνωρίζουμε μέσα σε ένα Σ σύνολο, ποιο mbb έχει ανατεθεί σε ποια συνιστώσα μεταβλητή. Το γεγονός αυτό απλοποιεί σημαντικά τα πράγματα. Ας θυμηθούμε τώρα τον τρόπο με τον οποίο δουλεύει ο αλγόριθμος `GLOBALCHECKCONSTRAINTNTB`. Αυτός για κάθε μεταβλητή περιοχής που εμφανίζεται ως πρωτεύουσα μεταβλητή σε κάποιο περιορισμό κατεύθυνσης, καλεί τον `CHECKCONSTRAINTNTB` περνώντας σαν παραμέτρους στον τελευταίο τα Σ σύνολά της (είτε όλα μαζί για την περίπτωση Σκιαδόπουλου-Κουμπαράκη, είτε ανά τριάδες για την περίπτωση Navarrete-Morales-Sciavicco). Συνεπώς θέλουμε μια δομή που να κάνει το binding μιας τέτοιας μεταβλητής με τα Σ σύνολά της. Η δομή αυτή ονομάζεται `RegionVar`. Επειδή είναι δύσκολο να δημιουργήσουμε απευθείας ένα σύνολο τέτοιων μεταβλητών του προηγούμενου τύπου, χρησιμοποιούμε μια ενδιάμεση μορφή την οποία λέμε `Intermediate` (προφανώς!). Η δομή αυτή κάνει το binding μιας μεταβλητής με τις περιοχές που έχουν ανατεθεί σε όλες τις συνιστώσες μεταβλητές της. Συνεπώς έχουμε μια μετατροπή που γίνεται από τη συνάρτηση `totalReconStruct()` σε δύο στάδια, καλώντας διαδοχικά τις συναρτήσεις `reconStruct1()` και `reconStruct2()`. Η μετατροπή αυτή είναι απαραίτητη, λόγω της μορφής της εισόδου που δέχονται οι συναρτήσεις `globalConstraintNTB()` και `NMSglobalConstraintNTB()`. Οι τελευταίες αποτελούν την υλοποίηση του τρίτου βήματος των αλγορίθμων Σκιαδόπουλου-Κουμπαράκη και Navarrete-Morales-Sciavicco αντίστοιχα.

algoMisc.h & algoMisc.cpp

Εδώ υπάρχουν κάποιες βοηθητικές συναρτήσεις οι οποίες χρησιμοποιούνται κυρίως από τον αλγόριθμο Navarrete-Morales-Sciavicco. Πριν αρχίσουμε να περιγράφουμε τη λειτουργία τους, θα πρέπει να θυμίσουμε ότι ο παραπάνω αλγόριθμος χρησιμοποιεί σαν βήμα προ επεξεργασίας τον αλγόριθμο αλγεβρικής κλειστότητας. Ο τελευταίος με τη σειρά του, χρησιμοποιεί την πράξη της αδύναμης σύνθεσης. Επειδή η πράξη αυτή είναι αρκετά χρονοβόρα για να εκτελείται επί τόπου (on-the-fly), κρίναμε σκόπιμο να αποθηκεύσουμε εκ των προτέρων το αποτέλεσμα αυτής της πράξης (όπου παίρνοντας όλες τις δυνατές περιπτώσεις σχηματίζεται ένας πίνακας, ο οποίος λέγεται πίνακας μεταβατικότητας) σε ένα

αρχείο, που φορτώνουμε στη μνήμη κάθε φορά που θέλουμε να εκτελέσουμε τον αλγόριθμο με την μέγιστη δυνατή ταχύτητα. Εναλλακτικά, η σύνθεση μπορεί να υπολογίζεται επί τόπου, χρησιμοποιώντας τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη. Με τον τρόπο αυτό όμως, υπάρχει μια πολύ μεγάλη μείωση της ταχύτητας εκτέλεσης.
Ο αριθμός των βασικών σχέσεων κατεύθυνσης είναι 511, συνεπώς μια σχέση κατεύθυνσης μπορεί να αναπαρασταθεί από ένα διάνυσμα 511 ψηφίων. Ωστόσο, επειδή οι διάφορες λειτουργίες στους υπολογιστές εκτελούνται πολύ πιο γρήγορα όταν δουλεύουν με αριθμούς που είναι δυνάμεις του 2, για λόγους βελτιστοποίησης της εκτέλεσης θεωρήσαμε διάνυσμα 512 ψηφίων. Ο πίνακας μεταβατικότητας απαιτεί χώρο $512^3=16$ MBytes στην μνήμη, αν και σαν αρχείο στον δίσκο λόγω της μετατροπής του bit σε byte χρειάζεται περίπου 128 MBytes. Αξίζει να σημειώσουμε ότι ο πίνακας μεταβατικότητας αποθηκεύεται σαν ASCII αρχείο και όχι σαν binary αρχείο, ώστε να μπορεί να χρησιμοποιηθεί και με τους δύο τύπους αρχιτεκτονικής επεξεργαστών (little endian VS big endian). Επειδή η κατασκευή αυτού του πίνακα είναι μια χρονοβόρα διαδικασία (λόγω της χρήσης του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη), πραγματοποιείται τμηματικά με τις συναρτήσεις `genTTFs()` οι οποίες κατασκευάζουν είτε μια μεμονωμένη γραμμή, είτε ένα διάστημα από γραμμές (πχ. 50 – 100). Αφού δημιουργηθούν όλα τα αρχεία που αντιστοιχούν στις γραμμές, στο τέλος με μια παράθεση αυτών παίρνουμε ολόκληρο τον πίνακα. Ο τελευταίος φορτώνεται στη μνήμη χρησιμοποιώντας τη συνάρτηση `getTTF()`. Αυτές οι συναρτήσεις εισόδου-εξόδου, χρησιμοποιούν τις βοηθητικές συναρτήσεις `bail()`, `buildRow()`, `readRow()`, `writeRow()`.
Μια εναλλακτική προσέγγιση με την οποία θα είχαμε υψηλή ταχύτητα, αλλά και μείωση του απαιτούμενου χώρου στη μνήμη, είναι η χρήση 2-επιπέδων για προσπέλαση στη μνήμη. Δηλαδή θα μπορούσαμε με την πρώτη προσπέλαση να παίρνουμε έναν δείκτη στα δεδομένα και με τη δεύτερη τα ίδια τα δεδομένα. Σε μια τέτοια περίπτωση αντί για 512^2*64 bytes, παρατηρούμε ότι θα χρειαζόμασταν μόνο $512^2*\$psize+\$diffelem*64$ bytes, όπου $\$psize$ είναι το μέγεθος του δείκτη (συνήθως 4 ή 8 bytes) και $\$diffelem$ το πλήθος των διαφορετικών στοιχείων του πίνακα μεταβατικότητας. Επειδή στον πίνακα μεταβατικότητας υπάρχουν πολλές επαναλήψεις στοιχείων, με αυτό το τρόπο θεωρούμε ότι μπορεί να μειωθεί σημαντικά η απαιτούμενη μνήμη.
Επιστρέφοντας τώρα στην περιγραφή των συναρτήσεων, θα πρέπει να αναφέρουμε ότι οι συναρτήσεις `compRels()`, `compRelSets()`, `wrapComp()` χρησιμοποιούνται για τη σύνθεση δύο σχέσεων, ενώ με τις συναρτήσεις `invRel()`, `invRelSet()` υπολογίζουμε την αντίστροφη σχέση.
Οι συναρτήσεις `printComp()`, `printInv()` τυπώνουν στη στάνταρ έξοδο το αποτέλεσμα της σύνθεσης και της αντιστροφής αντίστοιχα.
Τέλος, για τον αλγόριθμο αλγεβρικής κλειστότητας ακολουθήσαμε τη γνωστή τακτική, δηλαδή όπως και σε προηγούμενες περιπτώσεις αλγορίθμων, για ευκολία χωρίσαμε τον αλγόριθμο σε δύο τμήματα, στις συναρτήσεις `algebClosure()` και `algebClosureCore()`. Η πρώτη συνάρτηση, παίρνει σαν είσοδο ένα σύνολο βασικών περιορισμών κατεύθυνσης, δημιουργεί το γράφημα που του αντιστοιχεί και στη συνέχεια καλεί τη δεύτερη συνάρτηση. Η τελευταία συνάρτηση είναι

ουσιαστικά η υλοποίηση του αλγορίθμου αλγεβρικής κλειστότητας.

algoCons.h & algoCons.cpp

Σε αυτά τα αρχεία υπάρχουν οι ορισμοί των συναρτήσεων `SKconsist()` και `NMSconsist()`, οι οποίες είναι οι υλοποιήσεις ων αλγορίθμων Σκιαδόπουλου-Κουμπαράκη και Navarrete-Morales-Sciavicco αντίστοιχα. Οι συναρτήσεις αυτές απλώς καλούν εκείνες που αντιστοιχούν στα διάφορα βήματα των δύο αλγορίθμων και που αναφέραμε πιο πάνω. Έχουν οριστεί 5 verbosity modes για αυτές τις συναρτήσεις. Στο μηδενικό verbosity mode, απλώς επιστρέφουν **true** ή **false** ανάλογα με το εάν το σύνολο των περιορισμών κατεύθυνσης που δέχονται ως είσοδο είναι συνεπές ή όχι. Στο πρώτο verbosity mode τυπώνουν επιπλέον την παραπάνω πληροφορία στη στάνταρ έξοδο. Στο δεύτερο verbosity mode τυπώνεται επιπλέον ο συνολικός χρόνος εκτέλεσης (σε δευτερόλεπτα) που απαιτήθηκε. Στο τρίτο verbosity mode τυπώνεται ότι και στο δεύτερο, καθώς επίσης και ο χρόνος εκτέλεσης των διαφόρων βημάτων. Στο τέταρτο (και τελευταίο) verbosity mode τυπώνεται ότι και στο τρίτο και επιπλέον αν το σύνολο των περιορισμών είναι συνεπές, τυπώνεται η λύση που βρέθηκε από τον αλγόριθμο `CSPAN`.

main.h & main.cpp

Εδώ υπάρχει η κύρια συνάρτηση του προγράμματος `main()`, καθώς και δύο ακόμα βασικές συναρτήσεις που ονομάζονται `parseParams()` και `execMode()`, οι οποίες καλούνται από την `main()`. Η πρώτη από αυτές κάνει το parsing της γραμμής διαταγών και θέτει τον αντίστοιχο τρόπο λειτουργίας του προγράμματος, ενώ η δεύτερη απλά κάνει το πρόγραμμα να εκτελεστεί με τον τρόπο που τέθηκε πριν. Το πρόγραμμα αναγνωρίζει τις ακόλουθες παραμέτρους και λειτουργεί ως εξής:

-SK
Θεωρώντας ότι η στάνταρ είσοδος είναι ένα σύνολο βασικών περιορισμών κατεύθυνσης, το πρόγραμμα αποφασίζει για την συνέπεια αυτού του συνόλου χρησιμοποιώντας τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη.

-NMS
Όπως και πριν, μόνο που αυτή τη φορά χρησιμοποιείται ο αλγόριθμος Navarrete-Morales-Sciavicco.

-C
Το πρόγραμμα υποθέτει ότι η στάνταρ είσοδος είναι μια ακολουθία βασικών σχέσεων κατεύθυνσης και υπολογίζει τη σύνθεσή τους (πρώτη-δεύτερη, τρίτη-τέταρτη κτλ.)

-I
Όπως και πριν, μόνο που αυτή τη φορά υπολογίζονται οι αντίστροφες σχέσεις εκείνων που δίνονται από τη στάνταρ είσοδο.

-GTTF $lower $upper
Παράγονται τα αρχεία που αντιστοιχούν στις γραμμές από $lower μέχρι $upper του πίνακα μεταβατικότητας (προφανώς $0 < \$lower \leq \$upper < 512$).

-GTTR $relation
Παράγεται το αρχείο που αντιστοιχεί στη γραμμή της σχέσης $relaton του πίνακα μεταβατικότητας.

-TTF $file
Το πρόγραμμα φορτώνει τον πίνακα μεταβατικότητας από το αρχείο $file. Αν αυτή η παράμετρος δεν έχει οριστεί, η σύνθεση θα γίνεται χρησιμοποιώντας τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη. Αν δεν έχει δοθεί μία από τις παραμέτρους -NMS ή -C, τότε η παράμετρος -TTF αγνοείται σιωπηλά από το πρόγραμμα.

-v -vv -vvv
Αυτές οι παράμετροι ορίζουν το verbosity mode του προγράμματος, δηλαδή την ποσότητα της πληροφορίας που τυπώνεται κατά τη διάρκεια εκτέλεσης, κάνοντας το πρόγραμμα φλύαρο, πιο φλύαρο και ακόμα πιο φλύαρο αντίστοιχα.

5.3 Συνοψίζοντας

Σε αυτό το κεφάλαιο αναφέραμε τα σχετικά θέματα με την υλοποίηση που κάναμε. Αρχικά κάναμε μια μελέτη σχετικά με τις προδιαγραφές που έπρεπε να πληρεί το λογισμικό και στη συνέχεια περιγράψαμε τις διάφορες δομές δεδομένων και τις συναρτήσεις που χρησιμοποιούνται από το πρόγραμμα. Το πρόγραμμα αυτό χρησιμοποιήθηκε στη συνέχεια για να μελετήσουμε την απόδοση των δύο αλγορίθμων που εξετάζουμε. Χρησιμοποιώντας αυτές τις μετρήσεις, όπως θα δούμε στο επόμενο κεφάλαιο, μπορούμε να κάνουμε μια εμπειρική σύγκριση των δύο αλγορίθμων.

6. Μετρήσεις – Σύγκριση των αλγορίθμων

Σε αυτό το κεφάλαιο παρουσιάζουμε τις μετρήσεις που έγιναν σχετικά με την ταχύτητα των αλγορίθμων Σκιαδόπουλου-Κουμπαράκη και Navarrete-Morales-Sciavicco. Το κεφάλαιο αυτό χωρίζεται σε δύο ενότητες. Στην πρώτη ενότητα παρουσιάζουμε τη μεθοδολογία που χρησιμοποιήσαμε για την πραγματοποίηση και την καταγραφή των μετρήσεων, δηλαδή τον τρόπο με τον οποίο έγινε η παραγωγή των τυχαίων συνόλων περιορισμών κατεύθυνσης και στη συνέχεια τη μέθοδο με την οποία έγινε η εκτέλεση του προγράμματος και η συλλογή των αποτελεσμάτων. Στη δεύτερη ενότητα γίνεται η παρουσίαση των αποτελεσμάτων, η οποία βέβαια συνοδεύεται από ανάλογο σχολιασμό.

6.1 Μεθοδολογία μετρήσεων

Στην ενότητα αυτή παρουσιάζουμε τη μέθοδο παραγωγής και εκτέλεσης των πειραμάτων. Καταρχήν θα πρέπει να αναφέρουμε ότι όλες οι μετρήσεις έγιναν σε σύστημα Apple® Mac Mini με επεξεργαστή Intel® Core Duo 1.66 Mhz και μνήμη 512 MB RAM, που έτρεχε λειτουργικό σύστημα Mac OS® X Tiger. Για τη διεξαγωγή αυτών των πειραμάτων χρησιμοποιήθηκε η scripting γλώσσα προγραμματισμού PERL. Όπως αναφέραμε και στο προηγούμενο κεφάλαιο, ανάλογα με το τρόπο μεταγλώττισης του προγράμματος, μπορούμε να επιλέξουμε να μετράμε είτε χρόνο CPU είτε πραγματικό χρόνο (ο οποίος αναφέρεται συνήθως στη γλώσσα των υπολογιστών ως 'wall time'). Ο χρόνος που τελικά επιλέχτηκε για τις μετρήσεις, ήταν ο πραγματικός χρόνος εκτέλεσης, εξαιτίας της πολύ μεγάλης ακρίβειας (σε επίπεδο μικρο δευτερολέπτου) που παρέχει αυτή η προσέγγιση. Για να εξασφαλιστεί η αξιοπιστία των μετρήσεων, τα πειράματα πραγματοποιήθηκαν σε ένα idle σύστημα, όπου η μόνη εφαρμογή χρήστη που έτρεχε ήταν το πρόγραμμα που έκανε τον έλεγχο συνέπειας. Για κάθε στιγμιότυπο το πρόγραμμα έτρεχε τρεις φορές για την κάθε περίπτωση (δηλ. τους δύο υπό εξέταση αλγόριθμους) και στη συνέχεια παίρναμε σαν χρόνο εκτέλεσης τον μέσο όρο που προέκυπτε από αυτές τις τρεις εκτελέσεις.

Η παραγωγή των συνόλων περιορισμών έγινε χρησιμοποιώντας τα scripts `gen.pl` και `genall.pl`. Οι παράμετροι για την παραγωγή ενός τέτοιου συνόλου είναι το πλήθος των μεταβλητών που εμφανίζονται στους περιορισμούς και η πυκνότητα αυτού του συνόλου. Λέγοντας πυκνότητα εννοούμε το λόγο του αριθμού των περιορισμών που υπάρχουν στο σύνολο προς τον αριθμό όλων των δυνατών περιορισμών που υπάρχουν για αυτό τον αριθμό των μεταβλητών. Για παράδειγμα, αν έχουμε n μεταβλητές, τότε θεωρούμε ότι όλοι οι δυνατοί περιορισμοί είναι n(n -1) το πλήθος, αφού πιστεύουμε ότι δεν έχει ιδιαίτερο νόημα η συσχέτιση μιας περιοχής με τον εαυτό της, με την έννοια ότι ο μοναδικός περιορισμός που ικανοποιείται σε μια τέτοια περίπτωση είναι προφανώς ο περιορισμός του εγκλεισμού (δηλαδή αυτός που περιέχει τη σχέση B) και μόνον αυτός.

Το πρώτο script δέχεται τις δύο παραπάνω παραμέτρους και τυπώνει στη στάνταρ έξοδο ένα τέτοιο σύνολο. Αρχικά αποθηκεύει ένα διάνυσμα από n * n bits, έτσι ώστε αν η i-ιοστή μεταβλητή έχει σχετιστεί με την j-ιοστή μεταβλητή, να θέτεται το (i * n + j)-ιοστό bit. Με τον τρόπο αυτό αποφεύγουμε να σχετίσουμε για

δεύτερη φορά ένα ήδη σχετιζόμενο ζεύγος μεταβλητών. Επίσης, επειδή οι ατομικές σχέσεις είναι μόνο 9, επιλέγοντας ομοιόμορφα μια σχέση από τις σχέσεις του D (που είναι 218 το πλήθος) παρατηρήσαμε ότι πολύ σπάνια εμφανιζόταν κάποια ατομική σχέση στο σύνολο των περιορισμών που παράγονταν. Συνεπώς, για να υπάρχει κάποια σχετική ισορροπία μεταξύ των ατομικών περιορισμών και των μη ατομικών, για κάθε περιορισμό που παραγόταν θεωρούσαμε ισοπίθανα τα ενδεχόμενα εκείνος να είναι ατομικός ή μη ατομικός. Για να εκτελείται γρήγορα το script θεωρήσαμε ότι θα πρέπει να ισχύει $n \leq 1000$ και $d \leq 0.5$.

Το δεύτερο script εκτελεί το προηγούμενο για τις διάφορες τιμές των n και d. Συγκεκριμένα, θεωρήσαμε τις περιπτώσεις d = 0.01, 0.02, 0.05, 0.07, 0.1 και n = 10, 20,..,100. Επιλέξαμε σχετικά μικρό μέγεθος παραμέτρων, ώστε κάποια από τα σύνολα περιορισμών που θα παράγονταν να είναι συνεπή. Άλλωστε, όπως εύκολα μπορεί να αντιληφθεί κανείς λόγω της φύσεως του προβλήματος, είναι εξαιρετικά απίθανο να παραχθεί με τον παραπάνω τυχαίο τρόπο ένα πολυπληθές σύνολο περιορισμών που να είναι συνεπές. Το γεγονός αυτό **επηρεάζει ξεκάθαρα** τα αποτελέσματα των μετρήσεων **υπέρ** του αλγορίθμου **Navarrete-Morales-Sciavicco**, όπως θα διαπιστώσουμε στη συνέχεια.

Για κάθε περίπτωση, δηλαδή για κάθε ζεύγος παραμέτρων (n, d), παρήχθησαν 30 σύνολα περιορισμών. Επομένως, μετά την εκτέλεση του script `genall.pl` δημιουργήθηκαν 5 * 10 * 30 = 1500 αρχεία. Τα αρχεία αυτά έχουν όνομα της μορφής:

$d_$n_$i_i.txt

όπου $d η πυκνότητα, $n ο αριθμός των μεταβλητών και $i είναι ο αριθμός του συνόλου περιορισμών (από 1 έως 30) για αυτό το ζεύγος ($d, $n) παραμέτρων.

Πριν προχωρήσουμε στην εκτέλεση του προγράμματος για την αποθήκευση των χρόνων εκτέλεσης χρησιμοποιώντας τα παραπάνω αρχεία, θεωρήσαμε σκόπιμο να κάνουμε έναν έλεγχο εγκυρότητας του προγράμματος. Προφανώς για κάθε στιγμιότυπο οι δύο αλγόριθμοι θα πρέπει να βγάζουν σαν έξοδο το αυτό αποτέλεσμα. Τούτο επιβεβαιώθηκε με το script `validate.pl` το οποίο έκανε επιτυχώς αυτό τον έλεγχο στα παραπάνω 1500 αρχεία. Εκτός από αυτό, με την εκτέλεση του εν λόγω script, διαπιστώσαμε ότι από τα 1500 σύνολα περιορισμών που παρήχθησαν, τα 239 είναι συνεπή ενώ τα 1261 είναι ασυνεπή. Το γεγονός αυτό επιβεβαιώνει τον ισχυρισμό που έγινε παραπάνω, ότι δηλαδή καθώς το πλήθος των περιορισμών αυξάνεται, η πιθανότητα το σύνολο αυτό να είναι συνεπές ουσιαστικά εκμηδενίζεται, στην περίπτωση που χρησιμοποιούμε μια τέτοια τυχαία κατανομή.

Η εκτέλεση του προγράμματος και η συλλογή των αποτελεσμάτων έγινε από το script `runall.pl`. Το script αυτό εκτελεί διαδοχικά για τους δύο αλγόριθμους το πρόγραμμα conscheck και αποθηκεύει τους χρόνους εκτέλεσης σε αρχεία τα οποία έχουν όνομα της μορφής:

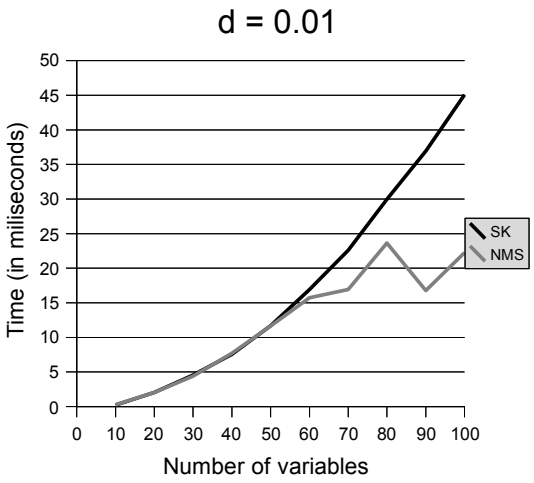
$algo_$d_o.txt

όπου $algo είναι 'sk' ή 'nms' για τους δύο αλγόριθμους αντίστοιχα και $d όπως πριν η

πυκνότητα. Κάθε τέτοιο αρχείο περιέχει 10 χρόνους (για n = 10, 20,.., 100) οι οποίοι είναι οι μέσοι όροι των αντίστοιχων 30 στιγμιοτύπων. Έχοντας πλέον δημιουργήσει τα παραπάνω αρχεία, στη συνέχεια με το βοηθητικό script `tomili.pl`, μετατρέψαμε τους χρόνους σε χιλιοστά του δευτερολέπτου, ώστε να είναι ελκυστικότερη η παρουσίαση των διαγραμμάτων που ακολουθούν.

6.2 Αποτελέσματα μετρήσεων

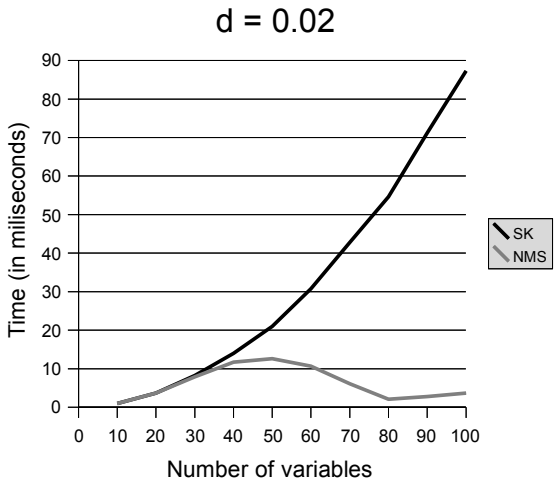
d = 0.01	SK	NMS
0	0.0	0.0
10	0.23	0.26
20	2.06	2.06
30	4.56	4.41
40	7.58	7.72
50	11.69	11.66
60	16.94	15.75
70	22.58	16.93
80	29.96	23.64
90	36.93	16.76
100	45.06	22.25



Διάγραμμα 1: d = 0.01

Σε αυτό το διάγραμμα βλέπουμε τα αποτελέσματα των μετρήσεων, όπου τα σύνολα των περιορισμών που εξετάζονται έχουν πολύ λίγα στοιχεία και επομένως τα περισσότερα από αυτά είναι συνεπή. Όπως βλέπουμε, οι δύο αλγόριθμοι για τιμές του n μέχρι 60, χρειάζονται περίπου τον δύο χρόνο εκτέλεσης. Αξίζει να παρατηρήσει κανείς την αυξομείωση στο χρόνο του αλγορίθμου Navarrete-Morales-Sciavicco για τιμές του n από 70 έως 100. Αυτή η αυξομείωση δικαιολογείται λόγω της ύπαρξης κάποιων ασυνεπών συνόλων, ιδιαίτερα για n = 70 και n = 90, όπου προκύπτει σχεδόν ο ίδιος χρόνος εκτέλεσης. Σε αυτές τις περιπτώσεις η ασυνέπεια εντοπίζεται πολύ γρήγορα από τον αλγόριθμο αλγεβρικής κλειστότητας και συνεπώς επηρεάζεται σημαντικά ο μέσος όρος.
Η ταχύτητα του τελευταίου είναι ο λόγος που κάνει τον αλγόριθμο Navarrete-Morales-Sciavicco να εκτελείται πάρα πολύ γρήγορα, όταν δέχεται σαν είσοδο ασυνεπή σύνολα περιορισμών. Αυτή η παρατήρηση θα γίνει πιο εμφανής βλέποντας τα επόμενα διαγράμματα που ακολουθούν.

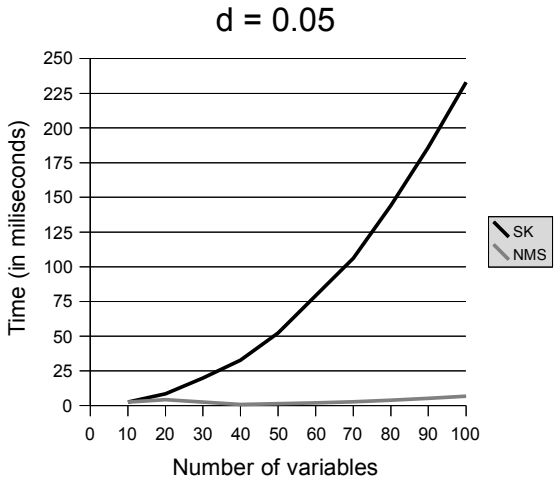
d = 0.02	SK	NMS
0	0.0	0.0
10	0.97	0.96
20	3.66	3.69
30	8.26	7.88
40	14.02	11.74
50	20.97	12.64
60	30.77	10.68
70	42.78	6.14
80	54.65	2.12
90	71.19	2.81
100	87.29	3.68



Διάγραμμα 2: d = 0.02

Εδώ πάλι έχουμε σχετικά μικρό αριθμό στοιχείων στα σύνολα των περιορισμών. Παρατηρούμε ότι για n = 50, έχουμε ένα μέγιστο σημείο για τον αλγόριθμο Navarrete-Morales-Sciavicco. Αυτό συμβαίνει γιατί καθώς αυξάνεται το πλήθος των μεταβλητών, ελαττώνεται το πλήθος των συνεπών συνόλων, με αποτέλεσμα να μειώνεται ο μέσος χρόνος εκτέλεσης. Αντίθετα, ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη φαίνεται να μην επηρεάζεται πολύ από αυτό το γεγονός, αφού βλέπουμε ότι η καμπύλη του χρόνου του συνεχίζει να αυξάνεται ομαλά.

d = 0.05	SK	NMS
0	0.0	0.0
10	2.43	2.42
20	8.39	4.19
30	19.77	2.47
40	32.73	0.74
50	52.27	1.23
60	79.23	1.86
70	106.15	2.7
80	144.11	3.74
90	186.12	5.14
100	233.07	6.69

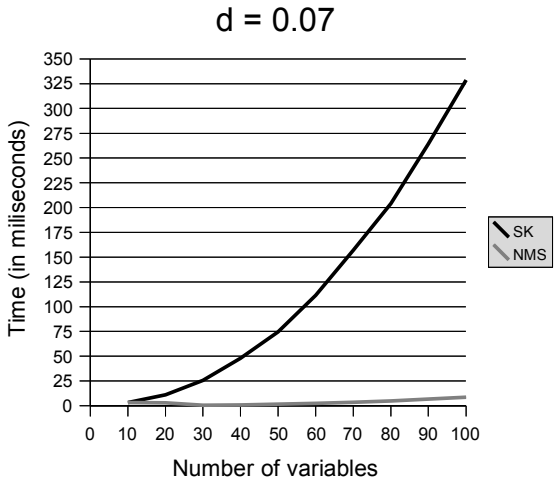


Διάγραμμα 3: d = 0.05

Σε αυτή την περίπτωση έχουμε πλέον ένα αρκετά μεγάλο αριθμό στοιχείων στα

σύνολα των περιορισμών, με αποτέλεσμα τα περισσότερα από αυτά (αν όχι όλα) να είναι ασυνεπή. Για n = 20 έχουμε ένα τοπικό μέγιστο για τον αλγόριθμο Navarrete-Morales-Sciavicco, όπως δηλαδή έγινε και στην προηγούμενη περίπτωση, μόνο που εδώ συμβαίνει πιο νωρίς. Από εκεί και μετά έχουμε μια μείωση του χρόνου μέχρις ότου έχουμε n = 40 και στη συνέχεια ο χρόνος αυξάνεται με πολύ μικρό ρυθμό.

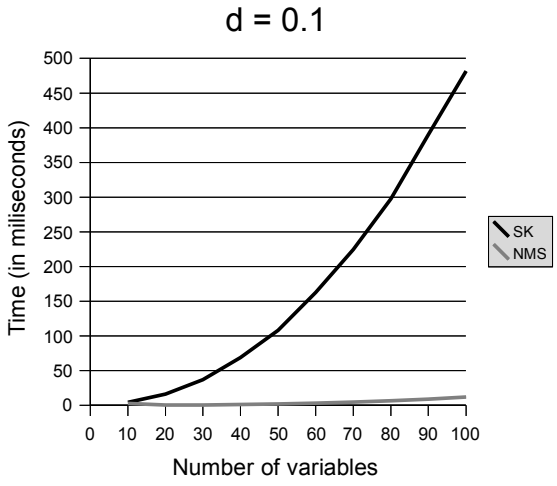
d = 0.07	SK	NMS
0	0.0	0.0
10	3.39	3.25
20	11.09	2.98
30	25.61	0.48
40	47.99	0.9
50	74.66	1.53
60	111.53	2.35
70	156.95	3.44
80	203.99	4.83
90	264.11	6.64
100	328.74	8.76



Διάγραμμα 4: d = 0.07

Στα διαγράμματα 4 και 5, έχουμε πλέον πολυπληθή σύνολα περιορισμών και η εικόνα είναι ουσιαστικά η ίδια με πριν, λόγω του αλγορίθμου αλγεβρικής κλειστότητας που όπως αναφέραμε, ανιχνεύει τις ασυνέπειες εξαιρετικά γρήγορα.

d = 0.1	SK	NMS
0	0.0	0.0
10	4.35	2.65
20	16.29	0.3
30	36.84	0.6
40	68.74	1.12
50	108.18	1.96
60	162.55	3.07
70	224.66	4.56
80	297.43	6.46
90	389.74	8.93
100	481.68	11.88



Διάγραμμα 5: d = 0.1

Βλέποντας κανείς τα παραπάνω αποτελέσματα μπορεί να κάνει τις εξής παρατηρήσεις:

- ➔ Επιβεβαιώνεται και εμπειρικά η βελτίωση στην ταχύτητα εκτέλεσης του αλγορίθμου Navarrete-Morales-Sciavicco σε σχέση με τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη. Ο τελευταίος σε λίγες μόνο περιπτώσεις και συγκεκριμένα για d = 0.01 και n = 10, 30 και για d = 0.02 και n = 20, κατάφερε να είναι ελαφρά πιο γρήγορος από τον πρώτο. Ωστόσο μπορούμε να πούμε ότι στις περιπτώσεις όπου έχουμε λίγους περιορισμούς (στα δύο πρώτα διαγράμματα) και κατά συνέπεια εκεί όπου τα περισσότερα σύνολα που εξετάζονται είναι συνεπή, οι χρόνοι εκτέλεσης των δύο αλγορίθμων βρίσκονται πολύ κοντά.

- ➔ Η καμπύλη του χρόνου του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη είναι ομαλή σε όλες τις περιπτώσεις που εξετάσαμε. Αυτό σημαίνει ότι η συνέπεια ή μη του συνόλου εισόδου, ουσιαστικά δεν επηρεάζει πολύ την ταχύτητα εκτέλεσης.

- ➔ Αντίθετα η καμπύλη του χρόνου του αλγορίθμου Navarrete-Morales-Sciavicco, ιδιαίτερα στα δύο πρώτα διαγράμματα, φαίνεται ότι επηρεάζεται πάρα πολύ από τη συνέπεια ή μη του συνόλου εισόδου.

- ➔ Συνδυάζοντας τις δύο προηγούμενες παρατηρήσεις με την εικόνα των τριών τελευταίων διαγραμμάτων, αντιλαμβανόμαστε ότι η διαφορά στο χρόνο εκτέλεσης είναι δραματική, στην περίπτωση όπου έχουμε ως είσοδο ένα πολυπληθές ασυνεπές σύνολο περιορισμών. Μάλιστα, εκτός από τα παραπάνω πειράματα, δοκιμάσαμε ενδεικτικά ένα τεράστιο σύνολο με 50.000 περιορισμούς. Είναι χαρακτηριστικό ότι ο αλγόριθμος Σκιαδόπουλου-Κουμπαράκη χρειάστηκε περίπου 54 δευτερόλεπτα για να εντοπίσει την ασυνέπεια, ενώ με τον αλγόριθμο Navarrete-Morales-Sciavicco η ασυνέπεια εντοπίστηκε σε μόλις 6 δευτερόλεπτα, δηλαδή περίπου 9 φορές πιο γρήγορα. Το γεγονός αυτό όπως αναφέραμε και πριν, οφείλεται στον αλγόριθμο αλγεβρικής κλειστότητας, ο οποίος από ότι φαίνεται στην πράξη δουλεύει εξαιρετικά γρήγορα και κάνει τον αλγόριθμο Navarrete-Morales-Sciavicco να αποφεύγει την εκτέλεση των υπόλοιπων χρονοβόρων βημάτων. Βέβαια, ο αλγόριθμος αλγεβρικής κλειστότητας θα μπορούσε να χρησιμοποιηθεί και από τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη, ωστόσο ο στόχος μας ήταν να συγκρίνουμε τις πρωτότυπες εκδοχές των δύο αλγορίθμων, όπως δηλαδή εμφανίζονται αυτοί στη βιβλιογραφία και όχι κάποια παραλλαγή αυτών που θα προέκυπτε εκ του αποτελέσματος.

- ➔ Σύμφωνα με τα παραπάνω, καταλήγουμε στο συμπέρασμα ότι ο αλγόριθμος Navarrete-Morales-Sciavicco αποτελεί ουσιαστική βελτίωση του αλγορίθμου Σκιαδόπουλου-Κουμπαράκη, ιδιαίτερα για πολυπληθή ασυνεπή σύνολα και προτείνουμε τη χρήση του στις περιπτώσεις όπου μπορεί εφαρμοστεί, δηλαδή όταν οι περιοχές του χώρου είναι συνεκτικές. Ωστόσο, διατηρούμε τις **επιφυλάξεις** μας για την περίπτωση των πολυπληθών συνεπών

συνόλων περιορισμών, αφού λόγω χρονικών περιορισμών που υπήρχαν κατά την εκπόνηση αυτής της διπλωματικής εργασίας, δεν κατέστη δυνατό να εξετάσουμε τέτοια σύνολα.

7. Συμπεράσματα

Με το κεφάλαιο αυτό ολοκληρώνεται ουσιαστικά η παρούσα διπλωματική εργασία. Συνοψίζοντας τη δουλειά που έγινε, θα πρέπει να αναφέρουμε ότι αρχικά παρουσιάστηκε το μοντέλο των χωρικών περιορισμών κατεύθυνσης και στη συνέχεια οι δύο αλγόριθμοι των Σκιαδόπουλου-Κουμπαράκη και των Navarrete-Morales-Sciavicco, οι οποίοι ορίστηκαν με βάση αυτό το μοντέλο. Κατόπιν, οι δύο αυτοί αλγόριθμοι υλοποιήθηκαν σε ένα πρόγραμμα που γράφτηκε σε C++ και ονομάστηκε conscheck και στη συνέχεια συγκρίθηκαν μεταξύ τους, όσον αφορά το χρόνο εκτέλεσης, χρησιμοποιώντας το προηγούμενο πρόγραμμα και τα κατάλληλα scripts που γράφτηκαν σε PERL. Το βασικό συμπέρασμα αυτής της σύγκρισης είναι ότι επιβεβαιώνεται και πειραματικά η θεωρητική ανωτερότητα του αλγορίθμου Navarrete-Morales-Sciavicco σε σχέση με τον αλγόριθμο Σκιαδόπουλου-Κουμπαράκη, ιδιαίτερα στην περίπτωση όπου το σύνολο περιορισμών που εξετάζεται είναι ασυνεπές.

Ένα ενδιαφέρον στοιχείο που θα μπορούσε να εξεταστεί σε μελλοντικές μετρήσεις, είναι η χρήση κάποιας γενικής μεθόδου παραγωγής πολυπληθών συνόλων περιορισμών κατεύθυνσης τα οποία είναι συνεπή. Δυστυχώς λόγω χρονικών περιορισμών που υπήρχαν κατά την εκπόνηση αυτής της εργασίας, δεν προλάβαμε να χρησιμοποιήσουμε τέτοια πειραματικά δεδομένα, δηλαδή πολυπληθή συνεπή σύνολα περιορισμών. Με τα είδη των συνόλων που δοκιμάσαμε, δηλαδή ολιγομελή (και συνήθως συνεπή) και πολυμελή (και ασυνεπή από τον τρόπο παραγωγής τους), προέκυψαν αρκετά συμπεράσματα σχετικά με την ταχύτητα εκτέλεσης κάποιων βημάτων. Διαπιστώσαμε λόγου χάρη, πόσο αποτελεσματικός είναι ο αλγόριθμος αλγεβρικής κλειστότητας για τις περιπτώσεις πολυπληθών ασυνεπών συνόλων περιορισμών. Ωστόσο θεωρούμε σημαντικό να γίνουν μελλοντικά μετρήσεις με σύνολα περιορισμών όλων των ειδών, ώστε να έχουμε μια ολοκληρωμένη εικόνα για την απόδοση των δύο αλγορίθμων, αφού οι μετρήσεις που κάναμε **σαφέστατα ευνοούν** τον αλγόριθμο **Navarrete-Morales-Sciavicco**.

Κάτι που επίσης θεωρούμε αρκετά ενδιαφέρον και μπορεί να γίνει μελλοντικά, είναι η δημιουργία μιας εφαρμογής διεπαφής (GUI frontend) για την εφαρμογή conscheck. Ένα τέτοιο πρόγραμμα εκτός από τη φιλικότητα προς το χρήστη, θα μπορούσε να προσφέρει και μια οπτική αναπαράσταση της λύσης, εφόσον βέβαια θα ήταν συνεπές το σύνολο των περιορισμών που θα εξεταζόταν.

8. Βιβλιογραφία

[ALLEN83]: J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM, 26(11):832–843, 1983

[GE00]: R. Goyal and M. Egenhofer, Consistent Queries over Cardinal Directions across Different Levels of Detail, Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications, 2000

[NMS07]: Isabel Navarrete, Antonio Morales and Guido Sciavicco, Consistency Checking of Basic Cardinal Constraint over Connected Regions, Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI2007), 495 – 500, 2007

[RL05]: J. Renz and G. Ligozat, Weak Composition for Qualitative Spatial and Temporal Reasoning, Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP'05), 2005

[SK04]: S. Skiadopoulos and M. Koubarakis, Composing cardinal direction relations, Artificial Intelligence, 152(2):143–171, 2004

[SK05]: S. Skiadopoulos and M. Koubarakis, On the consistency of cardinal direction constraints, Artificial Intelligence, 163(1):91–135, 2005

[VB92]: Peter van Beek, Reasoning About Qualitative Temporal Information, Artificial Intelligence, 58:297–326, 1992

[VB96]: Peter van Beek and Dennis W. Manchak, The design and experimental analysis of algorithms for temporal reasoning, Journal of Artificial Intelligence Research 4:1-18, 1996