ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Διάδοσης Περιορισμών και Αναζήτησης στον Προγραμματισμό με Περιορισμούς

Νικόλαος Ι. Ποθητός

Επιβλέπων: Παναγιώτης Σταματόπουλος, Επίκ. Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2009

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Διάδοσης Περιορισμών και Αναζήτησης στον Προγραμματισμό με Περιορισμούς

Νικόλαος Ι. Ποθητός

Α.Μ.: Μ780

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής ΕΚΠΑ
Σταύρος Κολλιόπουλος, Επίκουρος Καθηγητής ΕΚΠΑ

Μάρτιος 2009

Περίληψη

Τα προβλήματα ικανοποίησης περιορισμών (ΠΙΠ) απαντώνται σε πολλούς τομείς, όχι μόνο της Πληροφορικής, αλλά και της καθημερινότητας. Π.χ. ένα δύσκολο τέτοιο πρόβλημα είναι η κατάρτιση ωρολογίου προγράμματος για εκπαιδευτικά ιδρύματα. Ο Προγραμματισμός με Περιορισμούς είναι μία δημοφιλής σύγχρονη μεθοδολογία επίλυσης ΠΙΠ, που εστιάζει στο να κάνει τη φάση εύρεσης λύσεων ενός ΠΙΠ όσο το δυνατόν αποδοτικότερη και ανεξάρτητη από τη φάση διατύπωσής του. Σε αυτήν την εργασία επιλέξαμε τρία βασικά συστατικά του Προγραμματισμού με Περιορισμούς, την απεικόνιση του πεδίου τιμών των μεταβλητών ενός ΠΙΠ, τη διάδοση (επιβολή) των περιορισμών σε αυτές, καθώς και τη μέθοδο αναζήτησης λύσης –συγκεκριμένα τη διχοτόμηση πεδίων τιμών– και προτείναμε καινούργιες τεχνικές, η αποδοτικότητα των οποίων ελέγχθηκε και πειραματικά.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ: διάδοση περιορισμών, συνέπεια ακμών, διχοτόμηση πεδίων τιμών

Abstract

Constraint satisfaction problems (CSPs) appear in many areas not only of Computer Science, but in daily routine too. E.g. timetabling for educational institutes is a (difficult) CSP. Constraint Programming is a popular modern methodology for solving CSPs, that focuses on making the solving phase of a CSP as much efficient and independent of the problem statement phase, as possible. In this work we chose three critical components of Constraint Programming, the representation of the domains of CSP variables, the propagation of the constraints to the variables, and the search (for solution) method—specifically the domain-splitting method—and we proposed new techniques. We also used experiments to show the efficiency of our proposals.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: constraint propagation, arc-consistency, domain-splitting

Στη γιαγιά μου, Βάσω,
και στη μνήμη του φίλου και συμφοιτητή
Γιάννη Βελεσιώτη

Περιεχόμενα

Πρόλογος **15**

1 Εισαγωγή **19**

2 Προγραμματισμός με Περιορισμούς **21**
2.1 Ορισμός 21
2.2 Παράδειγμα 23
2.2.1 Κρυπτάριθμοι 23
2.2.2 Διατύπωση ενός Κρυπταρίθμου ως ΠΙΠ 23
2.3 Αναζήτηση 24
2.3.1 Γέννα-και-Δοκίμαζε 25
2.3.2 Τοπική Αναζήτηση 26
2.3.3 Μέθοδοι με Οπισθοδρόμηση 27
2.3.4 Διάδοση Περιορισμών 29

3 Η Μηχανή Διάδοσης Περιορισμών AC-5+ **35**
3.1 Γνωστοί Αλγόριθμοι Επιβολής Συνέπειας Ακμών 35
3.1.1 Αλγόριθμοι Αδρής Υφής 36
3.1.2 Αλγόριθμοι Λεπτής Υφής 36
3.1.3 Παραμετρικοί Αλγόριθμοι 37
3.2 Ο Αλγόριθμος AC-5 37
3.3 Μία Πρώτη Προσαρμογή του AC-5 39
3.3.1 Περισσότερες Τροποποιήσεις 40
3.3.2 Γενικευμένη Συνέπεια Ακμών 43
3.4 Παραδείγματα Συναρτήσεων ArcCons και LocalArcCons . 45

3.4.1 Ο Περιορισμός $X = Y$ 45
3.4.2 Ο Περιορισμός $X \neq Y$ 46
3.4.3 Μία Παραλλαγή της c_2.LocalArcCons 47
3.5 Κατηγοριοποίηση Μεθόδων LocalArcCons 48
3.5.1 Συμβάντα που Ενεργοποιούν Συναρτήσεις Διάδοσης . . 48
3.5.2 Η Συνάρτηση Διάδοσης LocalArcCons 49
3.5.3 Θεωρητική Ομαδοποίηση των Συμβάντων 50
3.5.4 Υλοποίηση Ομαδοποίησης Συμβάντων 52
3.5.5 Γρήγορη Εισαγωγή στην Ουρά Νέου Τύπου 52
3.6 Ο Αλγόριθμος AC-5+ 54
3.6.1 Ενσωμάτωση c.LocalArcCons $3^{\eta\varsigma}$ Κατηγορίας 54
3.7 Πειραματικά Αποτελέσματα 56

4 Απεικόνιση Πεδίου Τιμών ως Σύνολο Διαστημάτων 61
4.1 Απεικόνιση Συνόλου Διαστημάτων 61
4.2 Αλγόριθμος Διαγραφής/Αναζήτησης 62
4.3 Εφαρμογή σε Πρόβλημα Ακολουθίας DNA 65
4.3.1 Ορισμός Χρωμοσώματος 65
4.3.2 Ένα Απλό Πρόβλημα Ακολουθιών 65
4.3.3 Δυσκολίες κατά την Επίλυση 66
4.3.4 Συμπεράσματα 67

5 Δίκαιη Διχοτόμηση Πεδίων Τιμών 69
5.1 Μία Διαφορετική Οπτική 70
5.2 Ο Περιορισμός $X < Y$ 71
5.3 Άλλοι Τύποι Περιορισμών 73
5.4 Ο Περιορισμός $X = Y^2$ 74
5.5 Περισσότεροι Περιορισμοί 75
5.6 Πειραματικά Αποτελέσματα 76
5.6.1 Υβριδικό Ευριστικό Επιλογής Μεταβλητής 76
5.6.2 Αποτελέσματα Βελτιστοποίησης 79

6 Συμπεράσματα και Μελλοντικές Κατευθύνσεις 81

Α΄ Πρόβλημα Κατάρτισης Ωρολογίου Προγράμματος 83
Α΄.1 Εισαγωγή 84
Α΄.2 Οντότητες 84
Α΄.3 (Αυστηροί) Περιορισμοί 85

Α′.4 Ποιότητα Ωρολογίου Προγράμματος 85

Β′ Πηγαίος Κώδικας για το Πρόβλημα Ακολουθίας DNA της § 4.3.2 **87**
Β′.1 Κώδικας ECL^{i}PS^e 87
Β′.2 Κώδικας Επιλυτή ILOG 88
Β′.3 Κώδικας Επιλυτή NAXOS 89

Βιβλιογραφία **91**

Ευρετήριο **95**

Πρόλογος

Νάξος, Αθήνα, Κόρινθος, Κεχριές, Ξάνθη, Σάπες, ορεινή Ροδόπη, Φέρες, Διδυμότειχο. Είναι αναπόφευκτο να μου έρχονται στο μυαλό τοπία από τα μέρη όπου έτυχε να διαμείνω όλο αυτό το χρονικό διάστημα κατά το οποίο εκπονούσα τούτη την εργασία. Άλλοτε σε διακοπές, αγναντεύοντας το Αιγαίο, άλλοτε στη στρατιωτική μου θητεία, σκοτώνοντας την ώρα μου στη σκοπιά, δεν σταμάτησα να τριβελίζω τη σκέψη μου με τα θέματα της διπλωματικής.

Ευχαριστώ πάρα πολύ τον επιβλέποντα της εργασίας Επίκουρο Καθηγητή Παναγιώτη Σταματόπουλο που βρισκόταν πάντα «online» στο πλευρό μου, από την πρώτη μέχρι και την τελευταία στιγμή. Η πετυχημένη επιστημονική του πορεία και η πανθομολογούμενη διδακτική του δεινότητα όχι μόνο δεν επισκιάζουν, αλλά αναδεικνύουν τον σπάνιο για την εποχή μας ρομαντισμό με τον οποίο αντιμετωπίζει την έρευνα, το εκπαιδευτικό του λειτούργημα και γενικότερα τον συνάνθρωπο.

Ευχαριστώ θερμά τον Επίκουρο Καθηγητή Σταύρο Κολλιόπουλο για τη συμμετοχή του στην επιτροπή εξέτασης της εργασίας. Χωρίς καλά-καλά να το συνειδητοποιήσω, η παρουσία του με οδήγησε στο να δω από μία πιο «θεωρητική» σκοπιά τους αλγορίθμους που σχεδιάσαμε, εξάγοντας κατ΄ αυτόν τον τρόπο πολύτιμα συμπεράσματα.

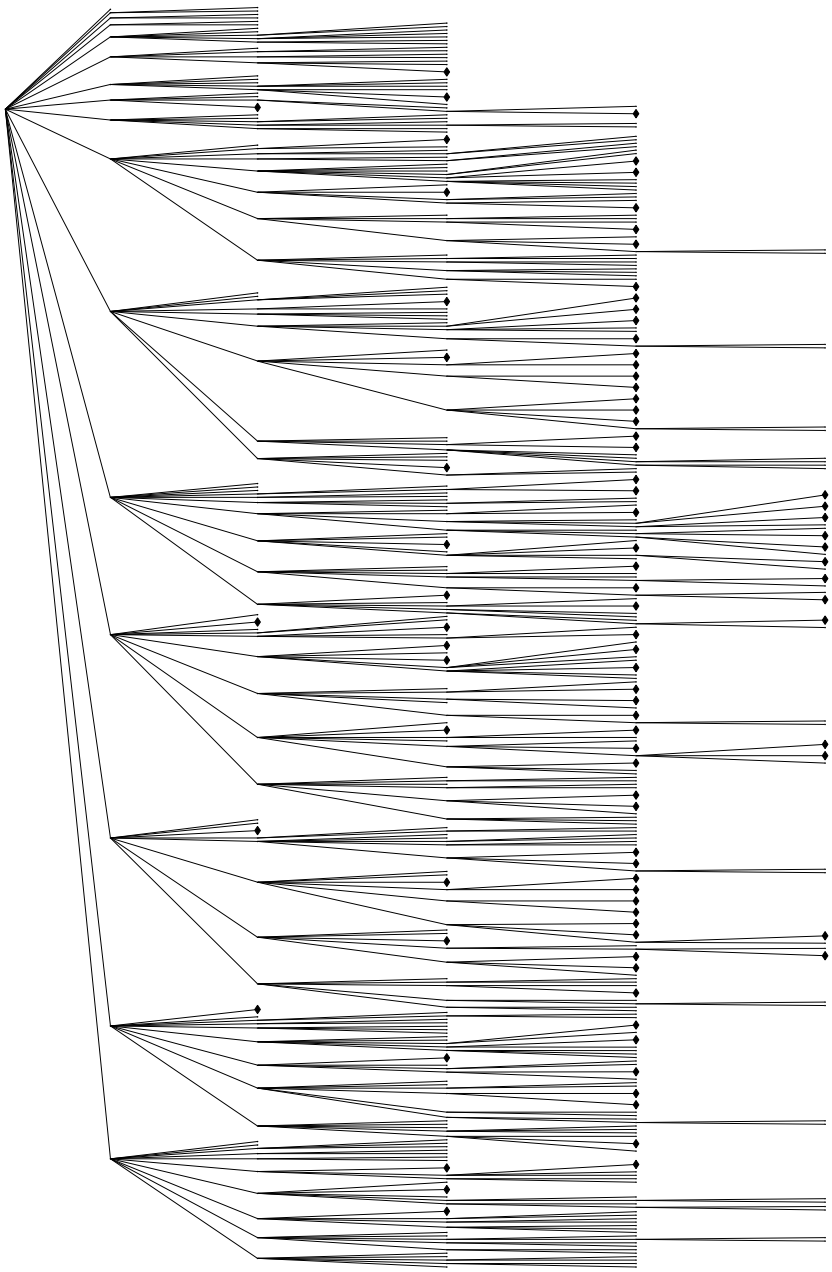
Ευχαριστώ τους Διονύση Κακαβούλη, Γιώργο Καστρίνη και Μιχάλη Σιούτη που μου έκαναν την τιμή να παρευρεθούν στην παρουσίαση της εργασίας. Τους εύχομαι ολόψυχα να παράγουν καλύτερα αποτελέσματα από τα δικά μου στις δικές τους εργασίες –της κοινής μας ερευνητικής περιοχής– και τους υπόσχομαι ότι θα προσπαθήσω να τους βοηθήσω αν με χρειαστούν. Μακάρι να καταφέρουμε να έχουμε την ίδια αμοιβαία συνεργασία, σαν αυτή που είχα με τον Φοίβο Θεοχάρη και τον Αλέξανδρο Παπαγγελή, οι οποίοι είναι πλέον μεταπτυχιακοί

φοιτητές· τους εύχομαι ό,τι καλύτερο για τη συνέχεια.

Ευχαριστώ τους «αφανείς ήρωες» αυτής της εργασίας, τους γονείς μου δηλαδή, οι οποίοι, πέρα από την ηθική και υλική υποστήριξή τους, «έπρεπε» να ακούν και τη γκρίνια μου κάθε φορά που κάτι πήγαινε στραβά.

Ευχαριστώ τον Ανθυπολοχαγό Απόστολο Πουρνάρα για την επιπλέον άδεια που μου χορήγησε –για να προχωρήσω μία εφαρμογή σχετική με τη διπλωματική– όταν υπηρετούσα τη στρατιωτική μου θητεία στο Διδυμότειχο, τον χειμώνα του 2007–2008. Τέλος, ευχαριστώ τον Συνταγματάρχη Γιώργο Πόθο για το ενδιαφέρον του σχετικά με την πορεία της εργασίας, αλλά και για την προτροπή του να συνεχίσω τις σπουδές μου· μακάρι να μην τον απογοητεύσω ☺.

Η εργασία αφιερώνεται ως ελάχιστος φόρος μνήμης και τιμής στον αείμνηστο συμφοιτητή, φίλο και συνεργάτη Γιάννη Βελεσιώτη ο οποίος έφυγε αναπάντεχα από κοντά μας το βράδυ της πρωτοχρονιάς του 2008. Μακάρι να αναπαύεται εκεί ψηλά που είναι βλέποντας ότι η δουλειά που έχει προσφέρει στην ευρύτερη ερευνητική περιοχή αποτελεί ακόμα σημείο αναφοράς για πολλούς από εμάς.



Σχήμα 1: Δένδρο αναζήτησης για το πρόβλημα της τοποθέτησης 8 βασιλισσών σε μία σκακιέρα, έτσι ώστε να μην απειλούνται μεταξύ τους. Διακρίνονται 92 μαύροι ρόμβοι που παριστάνουν τις λύσεις. (Το διάγραμμα παρήχθη αυτόματα από τον επιλυτή Naxos.)

Κεφάλαιο 1

Εισαγωγή

Οι ηλεκτρονικοί υπολογιστές θα γίνουν τα πιο θαυμάσια εργαλεία από τη στιγμή που η συγγραφή και η αποσφαλμάτωση προγραμμάτων θα είναι πλέον περιττή.

— Jean-Louis Laurière, 1976.

Ο Προγραμματισμός με Περιορισμούς είναι ένας από τους βασικούς τομείς της Τεχνητής Νοημοσύνης. Για κάποιον «αμύητο» σίγουρα η προηγούμενη πρόταση δεν θα έβγαζε και πολύ νόημα. Πόσο άμεση μπορεί να είναι η σχέση ενός είδους «προγραμματισμού» με τη «νοημοσύνη» που έχει ένας υπολογιστής;

Για να γίνει κατανοητή αυτή η σύνδεση, θα πρέπει να αναφέρουμε έναν από τους βασικούς στόχους του Προγραμματισμού με Περιορισμούς: να αποσυνδέσει τη φάση διατύπωσης ενός προβλήματος από τον αλγόριθμο επίλυσής του. Κατ' αυτόν τον τρόπο θα μπορεί ένας προγραμματιστής-χρήστης να δηλώνει εύκολα το πρόβλημά του και στη συνέχεια να καλείται ένα σύστημα επίλυσης να κάνει την υπόλοιπη δουλειά. Όσο πιο «έξυπνο» είναι αυτό το σύστημα, τόσο λιγότερη δουλειά θα έχει να κάνει ο χρήστης του [15]. Στο επόμενο κεφάλαιο θα γίνουμε πιο σαφείς αναφορικά με τις θεμελιώδεις ιδέες που εμπεριέχονται στον Προγραμματισμό με Περιορισμούς.

Μία από αυτές αφορά στη μεγαλύτερη εκμετάλλευση των περιορισμών ενός προβλήματος, μέσω της λεγόμενης *διάδοσής* τους. Ένα από τα τρία θέματα με τα οποία ασχολούμαστε σε αυτήν την εργασία είναι το «πάντρεμα» των πρώτων μεθόδων διάδοσης περιορισμών με τις πιο πρόσφατες –πάνω στις οποίες βασίζονται όλα τα σύγχρονα πρακτικά συστήματα επίλυσης. Επίσης προτείνουμε έναν νέο τρόπο αναπαράστασης ενός πεδίου τιμών, για να είμαστε σε θέση να περιγράψουμε αποδοτικά μεγάλα πεδία τιμών των μεταβλητών (συνιστωσών) των

προβλημάτων. Τέλος, επεκτείνουμε μία υπάρχουσα μέθοδο αναζήτησης (που λέγεται *διχοτόμηση πεδίων τιμών*) έτσι ώστε να λαμβάνει υπόψη τις ιδιότητες και τους περιορισμούς του προβλήματος που επιλύουμε.

Κεφάλαιο 2

Προγραμματισμός με Περιορισμούς

Ο *Προγραμματισμός με Περιορισμούς* (Constraint Programming – CP) αποτελεί έναν ευρύτατο τομέα της Τεχνητής Νοημοσύνης και ένα ενεργό πεδίο της σύγχρονης επιστημονικής έρευνας. Εστιάζει στα *Προβλήματα Ικανοποίησης Περιορισμών* (ΠΙΠ – Constraint Satisfaction Problems – CSPs), ο μαθηματικός ορισμός των οποίων δίδεται παρακάτω.

Παραδείγματα τέτοιων προβλημάτων απαντώνται σε όλους σχεδόν τους τομείς της Πληροφορικής. Π.χ. στη Θεωρητική Πληροφορική έχουμε το πρόβλημα ικανοποίησης μιας φόρμουλας που περιέχει λογικές (boolean) μεταβλητές [16], καθώς και άλλα NP-πλήρη προβλήματα· τελευταία, γίνεται επίσης πολλή συζήτηση όσον αφορά τον Προγραμματισμό με Περιορισμούς στην Υπολογιστική Βιολογία [3].

2.1 Ορισμός

Ο ορισμός –με άλλα λόγια η διατύπωση ενός ΠΙΠ– είναι πολύ δηλωτικός. Εξάλλου, είναι ανεξάρτητος από τη διαδικασία επίλυσης του προβλήματος και αυτός είναι ο λόγος για τον οποίον ο Προγραμματισμός με Περιορισμούς είναι τόσο δημοφιλής: οι πολυάριθμες μέθοδοι επίλυσης μπορούν να εφαρμοστούν σε οποιοδήποτε τέτοιο πρόβλημα.

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται μέσα από το εξής τρίπτυχο [41, 33]:

- *Περιορισμένες μεταβλητές* (ή απλά *μεταβλητές*) που αποτελούν το σύνολο $\mathscr{V} = \{V_1, V_2, \ldots, V_n\}$.

- *Πεδία τιμών* των μεταβλητών που αποτελούν το σύνολο $\mathscr{D} = \{D_1, D_2, \ldots, D_n\}$. Πρακτικά, μία μεταβλητή V_i λέμε ότι «παίρνει τιμές» από το D_i, για $1 \leq i \leq n$. Σε αυτή την εργασία θεωρούμε ότι κάθε πεδίο τιμών είναι πεπερασμένο και αποτελείται από διακριτές ακέραιες τιμές.

- *Περιορισμοί* που επιβάλλονται μεταξύ των μεταβλητών και αποτελούν το σύνολο $\mathscr{C} = \{C_1, C_2, \ldots, C_e\}$, όπου το C_i περιέχει μια σχέση μεταξύ των πεδίων τιμών των μεταβλητών ενός συνόλου $S_i \subseteq \mathscr{V}$. Ορίζουμε $C_i = (S_i, T_i)$, με $T_i \subseteq D_{i_1} \times D_{i_2} \times \cdots \times D_{i_q}$ το σύνολο με τους δυνατούς συνδυασμούς για τις τιμές των μεταβλητών του συνόλου $S_i = \{V_{i_1}, V_{i_2}, \ldots, V_{i_q}\}$. Ανάλογα με τον αριθμό των μεταβλητών της σχέσης (ο οποίος ισούται με $|S_i|$), κατατάσσουμε τον περιορισμό C_i στις εξής κατηγορίες/τάξεις περιορισμών:

  - Αν στον περιορισμό εμπλέκεται μία μεταβλητή, τότε είναι *μοναδιαίος* (unary).
  - Αν εμπλέκονται δύο μεταβλητές, τότε είναι *δυαδικός* (binary).
  - Αν εμπλέκονται περισσότερες μεταβλητές, τότε είναι *ανώτερης τάξης* (higher order). Στην ειδική περίπτωση που ένας περιορισμός μπορεί να συνδέσει γενικά έναν οποιονδήποτε αριθμό μεταβλητών n', τότε πρόκειται για έναν *καθολικό περιορισμό* (global constraint).

  Π.χ., για τις μεταβλητές V_1 και V_2 με πεδία τιμών έστω $\{0, 1, 2, 3\}$, ο περιορισμός της ισότητας μπορεί να δηλωθεί σαν $C_1(\{V_1, V_2\}, \{(0, 0), (1, 1), (2, 2), (3, 3)\})$. Παρότι αυτός ο συμβολισμός είναι όσο πιο γενικός γίνεται, στην πράξη χρησιμοποιούμε απλές σχέσεις για να περιγράψουμε τα δίκτυα περιορισμών. Στο παραπάνω παράδειγμα, ο περιορισμός μπορεί να γραφεί απλά σαν $V_1 = V_2$.

Όταν το πεδίο τιμών κάθε μεταβλητής γίνει μονοσύνολο, με άλλα λόγια όταν κάθε μεταβλητή ισούται με μία συγκεκριμένη τιμή, τότε έχουμε μία *ανάθεση* (assignment). (Όταν το πεδίο τιμών μιας μεταβλητής περιέχει μία μόνο τιμή, τότε λέμε ότι είναι *δεσμευμένη* (bound ή singleton).) Όταν μία ανάθεση είναι συνεπής ως προς τους περιορισμούς του προβλήματος, τότε μπορούμε να την ονομάσουμε και *λύση*.

2.2 Παράδειγμα

Σε αυτήν την ενότητα περιγράφεται ένα απλό παράδειγμα προβλήματος ικανοποίησης περιορισμών· συγκεκριμένα, ανήκει στην κατηγορία των κρυπταρίθμων.

2.2.1 Κρυπτάριθμοι

Ένας *κρυπτάριθμος* (cryptarithm, που περιγράφεται και σαν verbal arithmetic) είναι ένα μαθηματικό παίγνιο που περιλαμβάνει μία μαθηματική εξίσωση μεταξύ αγνώστων αριθμών, των οποίων τα ψηφία αναπαρίστανται από γράμματα. Ο στόχος του είναι να βρεθεί η αριθμητική τιμή κάθε γράμματος.

Ένα τετριμμένο παράδειγμα κρυπταρίθμου είναι ο $A + B = C$. Μία λύση είναι η $\{A \leftarrow 1, B \leftarrow 2, C \leftarrow 3\}$. Η ανάθεση $\{A \leftarrow 1, B \leftarrow 1, C \leftarrow 2\}$ δεν μας καλύπτει (δεν είναι λύση), γιατί, παραδοσιακά, καθένα γράμμα αναπαριστά ένα διαφορετικό ψηφίο. Επίσης, ισχύει ότι το πρώτο ψηφίο του κάθε αριθμού είναι μη μηδενικό.

2.2.2 Διατύπωση ενός Κρυπταρίθμου ως ΠΙΠ

Το κυρίως παράδειγμα ΠΙΠ που θα δούμε είναι ο κρυπτάριθμος:

```
    A S
+     A
-------
  M U M
```

Για να ορίσουμε το ΠΙΠ, δηλώνουμε τα τρία σύνολα που το αποτελούν:

- **Μεταβλητές**: A, S, M, U, δηλαδή όλα τα γράμματα που βλέπουμε στον κρυπτάριθμο.

- **Πεδία τιμών** (των αντίστοιχων μεταβλητών): $D_A = [1..9]$, $D_S = [0..9]$, $D_M = [1..9]$, $D_U = [0..9]$. Επειδή τα γράμματα A και M υπάρχουν στην αρχή λέξεων, έχουμε αφαιρέσει το 0 από τα αντίστοιχα πεδία. Με $[a..b]$ συμβολίζουμε το σύνολο ακεραίων $\{a, a+1, \ldots, b\}$.

- **Περιορισμοί**: Ο συνδετικός κρίκος μεταξύ των μεταβλητών είναι η εξίσωση:

```
    A S
+     A
-------
  M U M
```

που μπορεί να «μεταφραστεί» στην

$$10 \cdot A + S + A = 100 \cdot M + 10 \cdot U + M. \tag{2.1}$$

Αυτός είναι και ο πρώτος περιορισμός, ο οποίος πρέπει να ικανοποιείται από τις μεταβλητές. Ο δεύτερος περιορισμός συμπυκνώνεται στη φράση *«οι μεταβλητές πρέπει να είναι διαφορετικές μεταξύ τους»*. Η μαθηματική διατύπωση δίδεται παρακάτω:[1]

$$\begin{array}{lll} A \neq S, & A \neq M, & A \neq U, \\ & S \neq M, & S \neq U, \\ & & M \neq U. \end{array} \tag{2.2}$$

Τέλος, θα μπορούσε κανείς να προσθέσει στο σύνολο των περιορισμών τους

$$A \neq 0 \quad \text{και} \quad M \neq 0,$$

αφού τα A και M αποτελούν τα πρώτα γράμματα των λέξεων AS και MUM αντίστοιχα. Όμως, επειδή πρόκειται για μοναδιαίους περιορισμούς –στον καθέναν εμπλέκεται μία μεταβλητή–, έχουμε ήδη προσαρμόσει τα πεδία τιμών D_A και D_M έτσι ώστε να μην περιέχουν το 0. Κατά συνέπεια, δεν χρειάζεται να ασχοληθούμε πλέον με τους περιορισμούς αυτούς.

2.3 Αναζήτηση

Ο Προγραμματισμός με Περιορισμούς περιλαμβάνει δύο φάσεις, ή, με διαφορετικά λόγια, δύο ρεύματα: α) αυτό της διατύπωσης ενός ΠΙΠ και της μοντελοποίησής του με το οποίο ασχοληθήκαμε στις προηγούμενες ενότητες και β) αυτό

[1] Ο περιορισμός *«οι μεταβλητές ενός συνόλου, έστω P, πρέπει να είναι διαφορετικές μεταξύ τους»* είναι συνηθισμένος στα ΠΙΠ και ονομάζεται AllDifferent, ή AllDiff. Στο παράδειγμά μας δηλαδή, ισχύει:

$$\text{AllDifferent}(P), \quad P = \{A, S, M, U\}.$$

της επίλυσής του, δηλαδή της *αναζήτησης* λύσης (search), με το οποίο θα ασχοληθούμε κυρίως από δω και πέρα. Ωστόσο, δεν πρέπει να παραβλέψουμε ότι το πρώτο ρεύμα είναι εξίσου σημαντικό, καθώς η μεγαλύτερη ευκολία με την οποία διατυπώνει κανείς ένα ΠΙΠ θα κάνει τον Προγραμματισμό με Περιορισμούς ακόμα πιο χρήσιμο και δημοφιλή.

Στη συνέχεια παρουσιάζονται περιληπτικά οι βασικές κατηγορίες των μεθόδων επίλυσης. Ένα από τα κύρια πλεονεκτήματα του Προγραμματισμού με Περιορισμούς έχει να κάνει με τη γενικότητα των αλγορίθμων αναζήτησης που χρησιμοποιεί. Οι μέθοδοι δηλαδή είναι γενικές (generic) και μπορούν να «δουλέψουν» για οποιοδήποτε ΠΙΠ, άλλοτε με περισσότερη και άλλοτε με λιγότερη επιτυχία. Εξάλλου, δεν μπορεί να υπάρχει μία αποδοτική μέθοδος για όλα τα ΠΙΠ, αφού μεταξύ αυτών υπάρχουν προβλήματα που έχουν αποδειχθεί ότι είναι δύσκολα (NP-πλήρη) –όπως ο χρονοπρογραμματισμός– και για τη γρήγορη επίλυση των οποίων αναγκαζόμαστε να προσαρμόζουμε ή και να αλλάξουμε τις μεθοδολογίες αναζήτησης λύσης, επιλέγοντας μέσα από μία πληθώρα γνωστών αλγορίθμων.

2.3.1 Γέννα-και-Δοκίμαζε

Μία τετριμμένη μεθοδολογία επίλυσης ενός ΠΙΠ είναι να πάρουμε όλους τους δυνατούς συνδυασμούς των τιμών που μπορούν να ανατεθούν στις μεταβλητές (δηλαδή το καρτεσιανό γινόμενο $D_1 \times D_2 \times \cdots \times D_n$) και να τους εξετάζουμε έναν προς έναν (ως προς την ικανοποίηση όλων των περιορισμών), μέχρι να βρούμε μία λύση. Η μέθοδος αυτή ονομάζεται *γέννα-και-δοκίμαζε* (generate-and-test).

Δεν είναι δύσκολο να φανταστούμε πόσο αναποτελεσματικός είναι αυτός ο αλγόριθμος, αφού φτάνει στο σημείο να εξετάζει όλα τα στοιχεία του καρτεσιανού γινομένου που είναι εκθετικά πολλά ως προς το n.

Παράδειγμα

Επιστρέφουμε στο παράδειγμα της § 2.2. Για να επιλυθεί ο κρυπτάριθμος $AS + A = MUM$, θα πρέπει να πάρουμε τις αναθέσεις

A	S	M	U
1	0	1	0
1	0	1	1
1	0	1	2
		$\vdots$	
9	9	9	9

και να τις ελέγξουμε μία προς μία αν ικανοποιούν τους περιορισμούς (2.1) και (2.2).

Τελικά, θα φτάσουμε στην 7381^{η} ανάθεση, για να βρούμε τη λύση $\{A \leftarrow 9, S \leftarrow 2, M \leftarrow 1, U \leftarrow 0\}$, ενώ, αν δεν υπήρχε λύση, θα έπρεπε να ελέγξουμε όλους τους συνδυασμούς, ο αριθμός των οποίων είναι ίσος με $|D_A| \cdot |D_S| \cdot |D_M| \cdot |D_U| = 9 \cdot 10 \cdot 9 \cdot 10 = 8100$.

2.3.2 Τοπική Αναζήτηση

Η *τοπική αναζήτηση* (local search) χρησιμοποιείται ευρέως για την επίλυση δύσκολων υπολογιστικών προβλημάτων. Ακόμα και σήμερα που οι μέθοδοι αναζήτησης με οπισθοδρόμηση (οι οποίες ονομάζονται και *συστηματικές*, ή *άμεσες*, όπως θα δούμε και σε επόμενη ενότητα) εξελίχθηκαν σημαντικά, δεν έχουν φτάσει στο επίπεδο να λύνουν αρκετά μεγάλα και πολύπλοκα στιγμιότυπα προβλημάτων, τα οποία γίνεται να επιλυθούν μέσω τοπικής αναζήτησης (η οποία χαρακτηρίζεται και σαν *«έμμεση»* μέθοδος).

Ο αλγόριθμος πίσω από κάθε μέθοδο τοπικής αναζήτησης συνοψίζεται στην εξής πρόταση: Ξεκινάμε από μία (τυχαία) ανάθεση, η οποία πιθανόν να είναι *μερική*, δηλαδή να μην περιλαμβάνει όλες τις μεταβλητές και, επαναληπτικά, βελτιώνουμε αυτή την «υποψήφια λύση» κάνοντας μικρές αλλαγές. Ο ακριβής τύπος των αλλαγών αυτών είναι το σημείο στο οποίο διαφέρουν μεταξύ τους οι μέθοδοι τοπικής αναζήτησης [18].

Γειτονιές Αναθέσεων

Αν εφαρμόσουμε μία αλλαγή σε μία ανάθεση a_1 και προκύψει μία ανάθεση a_2, τότε λέμε ότι οι δύο αναθέσεις είναι *γειτονικές*. Κατά αυτόν τον τρόπο ορίζονται οι *γειτονιές* αναθέσεων. (Π.χ. αν η a_2 ανήκει στη γειτονιά της a_1, τότε γράφουμε $a_2 \in N(a_1)$.) Μία γνωστή γειτονιά είναι η *1-εναλλαγή* (1-exchange neighbourhood), σύμφωνα με την οποία δύο αναθέσεις είναι γειτονικές αν και μόνο αν διαφέρουν στην τιμή που έχει ανατεθεί σε μία μεταβλητή.

Επαναληπτική Βελτίωση

Μία απλή μέθοδος τοπικής αναζήτησης είναι η *επαναληπτική βελτίωση* (iterative improvement), γνωστή και ως *αναρρίχηση λόφου* (hill climbing). Για να μπορούμε να αξιολογούμε καθεμία ανάθεση σε αυτήν τη μέθοδο, χρησιμοποιούμε μία συνάρτηση αποτίμησης g. Όταν λοιπόν σε κάποιο βήμα της συγκεκριμένης μεθόδου έχουμε την ανάθεση a_1, προχωράμε σε μία ανάθεση $a_2 \in N(a_1)$, εφόσον $g(a_2) < g(a_1)$, εφόσον δηλαδή η συνάρτηση αποτίμησης βελτιώνεται. Αν $g(a_1) = 0$, τότε η a_1 είναι λύση και ο αλγόριθμος δεν είναι απαραίτητο να συνεχίσει.

2.3.3 Μέθοδοι με Οπισθοδρόμηση

Η τοπική αναζήτηση, αν και γνωστή για την αποτελεσματικότητά της σε μία ποικιλία σημαντικών προβλημάτων, ιδίως κατά την κλιμάκωσή τους, δεν είναι εν γένει πλήρης μέθοδος, δηλαδή δεν βρίσκει όλες τις λύσεις ενός προβλήματος, πράγμα που είναι εύκολο να φανεί ακόμα και σε μικρά στιγμιότυπα προβλημάτων. Συνεπώς, έχει γίνει πολλή έρευνα πάνω στις πλήρεις μεθόδους, οι οποίες εμπεριέχουν σε μικρότερο βαθμό το στοιχείο της τυχαιότητας και, συνήθως, είναι πιο κατανοητές και συνακόλουθα ελκυστικές.

Ενώ στην τοπική αναζήτηση και στη γέννα-και-δοκίμαζε δίναμε τιμές σε *όλες* τις μεταβλητές και στη συνέχεια ελέγχαμε αν παραβιάζονταν οι περιορισμοί, στην οπισθοδρόμηση δίνουμε τιμή σε *μία* μεταβλητή και έπειτα ελέγχουμε αν ισχύουν οι περιορισμοί για τις ήδη δεσμευμένες (bound) μεταβλητές.

Οι αλγόριθμοι με οπισθοδρόμηση προβλέπουν ότι αν κατά την ανάθεση τιμής σε μία μεταβλητή βρεθεί ότι παραβιάζεται κάποιος περιορισμός –μεταξύ των ήδη δεσμευμένων μεταβλητών πάντα–, τότε επιλέγεται μία άλλη τιμή από το πεδίο τιμών της. Αν έχουμε δοκιμάσει όλες τις τιμές του πεδίου της μεταβλητής, οπισθοδρομούμε (ή, με άλλα λόγια, υπαναχωρούμε), δηλαδή δοκιμάζουμε την επόμενη τιμή της προηγούμενης μεταβλητής που δεσμεύθηκε. Αν αποτύχουν όλες οι αναθέσεις και για αυτήν, τότε οπισθοδρομούμε στην προπροηγούμενη κ.ο.κ. Δέσμευση όλων των μεταβλητών με ικανοποίηση των περιορισμών σημαίνει επιτυχία, ενώ αντίθετα η ανάγκη να οπισθοδρομήσουμε πίσω από την πρώτη μεταβλητή ισοδυναμεί με αποτυχία.

Στον ψευδοκώδικα που ακολουθεί η οπισθοδρόμηση επιτυγχάνεται μέσω αναδρομής:

```
function BackTracking(Variables, Constraints)
    if exists unbound v in Variables then
        for each a in domain(v) do
            v ← a
            if there is no violation of Constraints
            for the Variables that are bound then
                BackTracking(Variables, Constraints)
            end if
        end for
        v ← domain(v)
    else
        return true                                  ▷ Βρέθηκε λύση!
    end if
    return false                                     ▷ Αποτυχία
end function
```

Κατά την εφαρμογή του αλγορίθμου στο παράδειγμα της § 2.2 εμφανίζονται τα εξής βήματα:

A	S	M	U	Περιορισμός που παραβιάζεται	Οπισθοδρόμηση
1					
1	0				
1	0	1		(2.2)	
1	0	2			
1	0	2	0	(2.1)	
1	0	2	1	(2.1)	
				⋮	
1	0	2	9	(2.1)	✓
1	0	3			
1	0	3	0	(2.1)	
1	0	3	1	(2.1)	
				⋮	
1	0	9	9	(2.1)	✓
1	1			(2.2)	
1	2				
				⋮	

Η μέθοδος ξεκινά με την ανάθεση $A \leftarrow 1$. Δεν έχουμε παραβίαση κάποιου περιορισμού, οπότε προχωράμε στην ανάθεση $S \leftarrow 0$. Η μερική ανάθεση $\{A \leftarrow 1, S \leftarrow 0\}$ δεν παραβιάζει τους περιορισμούς μεταξύ των *ήδη δεσμευμένων* μεταβλητών (τους υπόλοιπους περιορισμούς δεν τους εξετάζουμε ούτως ή άλλως), οπότε πάμε στο τρίτο βήμα και παίρνουμε την ανάθεση $\{A \leftarrow 1, S \leftarrow 0, M \leftarrow 1\}$. Όμως ο περιορισμός (2.2) (συγκεκριμένα ο $A \neq M$) δεν τηρείται, οπότε εξετάζουμε κατευθείαν την επόμενη τιμή του D_M (χωρίς να έχουμε φτάσει ακόμα στη U, γλιτώνοντας έτσι 10 ελέγχους).

Μετά την ανάθεση $\{A \leftarrow 1, S \leftarrow 0, M \leftarrow 2, U \leftarrow 9\}$ έχουμε την πρώτη οπισθοδρόμηση, για να πάρουμε τη μερική ανάθεση $\{A \leftarrow 1, S \leftarrow 0, M \leftarrow 3\}$.[2] Μέχρι να φτάσουμε στη λύση, έχουμε πολλά «κλαδέματα»[3] από τα οποία γλιτώνουμε πολλούς ελέγχους περιορισμών, όπως παραπάνω. Και αυτό γιατί δεν είναι απαραίτητο να εξετάζουμε αποκλειστικά *πλήρεις* αναθέσεις· μπορούμε να ελέγχουμε τους περιορισμούς μεταξύ των μεταβλητών ακόμα και για μια *μερική* ανάθεση.

Αποδεικνύεται όμως ότι στη χειρότερη περίπτωση η οπισθοδρόμηση έχει την ίδια αποδοτικότητα με τη γέννα-και-δοκίμαζε. Επινοήθηκαν λοιπόν τεχνικές, όπως η διάδοση περιορισμών που θα δούμε στη συνέχεια, για ακόμα περισσότερη μείωση του χώρου αναζήτησης.

2.3.4 Διάδοση Περιορισμών

Το «μυστικό» της διάδοσης περιορισμών είναι ότι προσπαθεί να εκμεταλλευτεί *όλους* τους περιορισμούς, δηλαδή όχι μόνο τους περιορισμούς που αφορούν τις ήδη δεσμευμένες μεταβλητές, όπως κάνει η μέθοδος της προηγούμενης ενότητας. Έχουμε ένα «κλάδεμα» του δένδρου αναζήτησης, με σκοπό να μην ανατεθούν στο μέλλον λανθασμένες τιμές (a priori pruning). Πριν την παρουσίαση των αλγορίθμων, θα δούμε πώς μπορεί να αποτυπωθεί σαν γράφος ένα ΠΙΠ.

Το *δίκτυο περιορισμών* (constraint network) είναι ένας γράφος με κόμβους που αναπαριστούν τις μεταβλητές. Τους κόμβους συνδέουν ακμές που αναπαριστούν τους περιορισμούς. Μια ακμή ενώνει δύο κόμβους-μεταβλητές και επομένως συμβολίζει έναν δυαδικό περιορισμό. Οι μοναδιαίοι περιορισμοί δεν είναι ανάγκη να αναπαρασταθούν, καθώς μπορούν να ικανοποιηθούν σε μια φάση

[2] Όταν παραβιάζονται περισσότεροι από ένας περιορισμοί, αρκεί να αναφέρουμε έναν μόνο από αυτούς στη στήλη «Περιορισμός που παραβιάζεται».

[3] Τα μονοπάτια τα οποία δεν οδηγούν σε λύση ονομάζονται ασυνεπή και στην αγγλική ορολογία «no-goods».

προεπεξεργασίας μέσω της διαγραφής κάποιων τιμών από τα πεδία των μεταβλητών. Αν είχαμε π.χ. τη μεταβλητή V_i με πεδίο τιμών $D_i = \{-2, -1, 0, 1, 2\}$ και τον μοναδιαίο περιορισμό $V_i \geq 0$, τότε απλά θα θέταμε σαν πεδίο τιμών της V_i το $D'_i = \{-\not{2}, -\not{1}, 0, 1, 2\} = \{0, 1, 2\}$, από την αρχή. Όσον αφορά την αναπαράσταση περιορισμών ανώτερης τάξης στο δίκτυο περιορισμών, δεν είναι απαραίτητη, αφού αποδεικνύεται ότι: *Κάθε πρόβλημα ικανοποίησης περιορισμών μπορεί να μετασχηματιστεί σε ένα ισοδύναμο πρόβλημα, που περιέχει μόνο δυαδικούς περιορισμούς.* Ο μετασχηματισμός αυτός λέγεται *δυαδικοποίηση* (binarization).

Το δίκτυο περιορισμών είναι το μέσο στο οποίο θα διαδίδονται οι περιορισμοί (propagation). Οι αλγόριθμοι που θα δούμε για αυτό το δίκτυο-γράφο, επιχειρούν να τον φέρουν σε μια κατάσταση *συνέπειας ακμών* (arc consistency – AC). Μια ακμή (V_i, V_j) είναι συνεπής, αν για κάθε $x \in D_i$, υπάρχει $y \in D_j$, τέτοιο ώστε το ζεύγος τιμών (x, y) να μην παραβιάζει τον περιορισμό για την ακμή αυτή. Όταν μια ακμή (V_i, V_j) είναι συνεπής, τότε *δεν* ισχύει απαραίτητα ότι και η (V_j, V_i) είναι συνεπής, αφού η συνέπεια είναι κατευθυνόμενη. Π.χ. έστω $V_1 \in \{17, 18\}$ και $V_2 \in \{7, 8, 9\}$ με $V_1 = V_2 + 10$. Ισχύει ότι η (V_1, V_2) συνεπής και η (V_2, V_1) ασυνεπής, αφού δεν υπάρχει $y \in D_1$, με $y = 9 + 10$.

Με τον παρακάτω αλγόριθμο μπορούμε να εξασφαλίσουμε τη συνέπεια μίας ακμής (V_i, V_j), αφαιρώντας τις τιμές του D_i που δεν ικανοποιούν τον περιορισμό για τις δύο μεταβλητές. (Π.χ. για την παραπάνω ακμή (V_2, V_1), αφαιρεί το 9 από το D_2.)

```
function REVISE(V_i, V_j)
    del ← false
    for each x in D_i do
        if there is no y ∈ D_j, with (x, y) not violating the constraint then
            delete x from D_i
            del ← true
        end if
    end for
    return del
end function
```

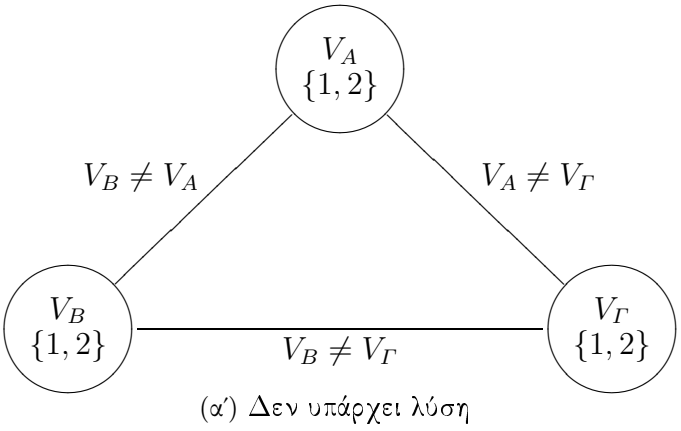
Η μέθοδος αυτή καλείται από κάποιον *αλγόριθμο επιβολής συνέπειας ακμών*. Δεν υπάρχει μόνο ένας τέτοιος αλγόριθμος, αλλά ολόκληρη οικογένεια αλγορίθμων AC. Π.χ. οι AC-1, AC-2, ..., AC-7 και ο πιο πρόσφατος AC-2001. Ας δούμε τον πρώτο αλγόριθμο AC [26]:

procedure AC-1(G)
  $Q \leftarrow \{(V_i, V_j) \mid (V_i, V_j) \in \text{arcs}(G)\}$
  repeat
    changed $\leftarrow$ false
    for each $(V_i, V_j) \in Q$ **do**
      changed $\leftarrow$ *changed* or REVISE(V_i, V_j)
    end for
  until not *changed*
end procedure

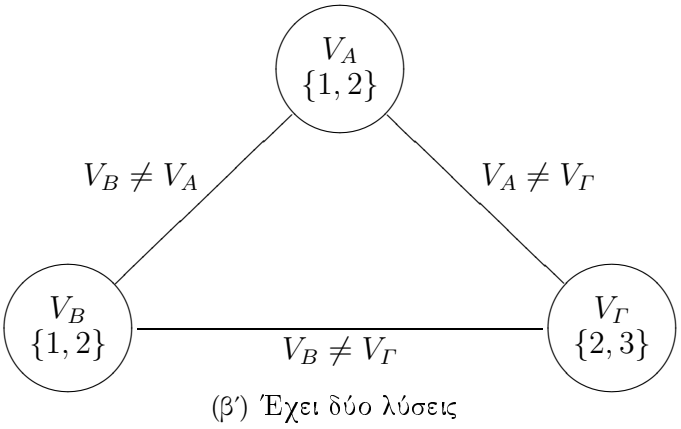
Παρατηρούμε ότι ο αλγόριθμος μετά από κάθε αλλαγή στον γράφο, ελέγχει ξανά τις ακμές του για να διαπιστωθεί αν είναι συνεπείς. Και αυτό γιατί μπορεί να συμβεί το φαινόμενο του ντόμινο: να αφαιρεθεί μια τιμή του πεδίου μιας μεταβλητής V_1 για γίνει συνεπής μια ακμή (V_1, V_2) και αυτό να προκαλέσει ασυνέπεια σε ακμές τύπου (V_x, V_1). Αν προσέξουμε τον AC-1 όμως, θα δούμε ότι δεν θα ελέγξει μόνο τις ακμές τύπου (V_x, V_1), αλλά όλες. Ο AC-3 έρχεται να σταματήσει αυτήν τη σπατάλη στους ελέγχους [26]:

procedure AC-3(G)
  $Q \leftarrow \{(V_i, V_j) \mid (V_i, V_j) \in \text{arcs}(G)\}$
  while $Q \neq \emptyset$ **do**
    select and delete (V_k, V_m) from Q
    if REVISE(V_k, V_m) **then**
      $Q \leftarrow Q \cup \{(V_i, V_k) \mid (V_i, V_k) \in \text{arcs}(G), i \neq m\}$
    end if
  end while
end procedure

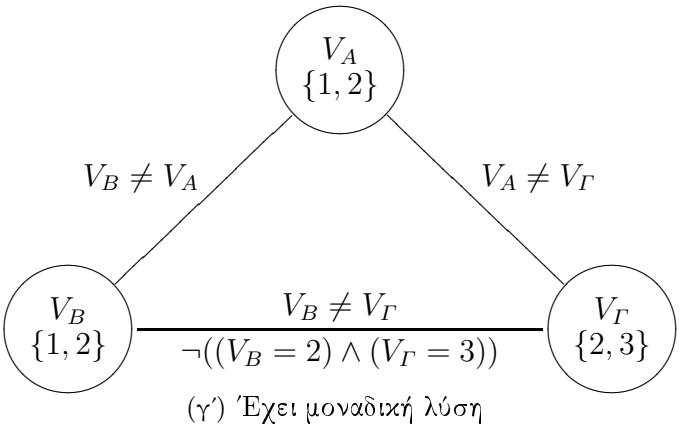
Τονίζεται ότι η συνέπεια των ακμών ενός γράφου δεν σηματοδοτεί ότι υπάρχουν μία ή περισσότερες λύσεις, ούτε καν ότι θα έχουμε λύση. Για του λόγου το αληθές παρουσιάζονται οι τρεις γράφοι του Σχήματος 2.1. Όταν όμως έχουν αφαιρεθεί όλες οι τιμές από κάποια μεταβλητή τότε σίγουρα δεν υπάρχει λύση. Από την άλλη πλευρά, όταν όλες οι μεταβλητές έχουν μονομελή πεδία τιμών σε κατάσταση συνέπειας ακμών, έχουμε μπροστά μας μία λύση.



(α′) Δεν υπάρχει λύση



(β′) Έχει δύο λύσεις



(γ′) Έχει μοναδική λύση

Σχήμα 2.1: Τρεις γράφοι με συνέπεια ως προς τις ακμές

Γενικευμένη Συνέπεια Ακμών

Για να επιβάλλουμε συνέπεια ακμών σε περιορισμούς ανώτερης τάξης (που είναι εκείνοι στους οποίους συμμετέχουν περισσότερες από δύο μεταβλητές), υπάρχει η διαδικασία της δυαδικοποίησης που προαναφέρθηκε. Μπορούμε, δηλαδή, να επιβάλλουμε συνέπεια ακμών στους δυαδικούς περιορισμούς που παράγονται από έναν περιορισμό ανώτερης τάξης.

Ο άλλος δρόμος είναι να επιβάλλουμε απευθείας τη λεγόμενη *γενικευμένη συνέπεια ακμών* (generalized arc consistency – GAC, ή hyper-arc consistency) στον ίδιο τον περιορισμό. Πρόκειται για μία επέκταση της έννοιας της συνέπειας ακμών και αναφέρεται σε μία μεταβλητή και έναν περιορισμό.

Μία μεταβλητή V_i είναι συνεπής ως προς έναν περιορισμό στον οποίο εμπλέκονται οι μεταβλητές $\{V_1, \ldots, V_{i-1}, V_i, V_{i+1}, \ldots, V_k\}$, αν για κάθε τιμή v_i του πεδίου τιμών της V_i υπάρχει μία πλειάδα τιμών $\langle v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k\rangle$ από τα αντίστοιχα πεδία τιμών, η οποία, μαζί με την v_i, δεν θα παραβιάζει τον περιορισμό.

Παράδειγμα. Έστω ότι έχουμε τον περιορισμό $V_1 + V_2 = V_3$, με $D_1 = D_2 = D_3 = [1\,..\,9]$, και έστω ότι θέλουμε να επιβάλλουμε γενικευμένη συνέπεια ακμών για τη μεταβλητή V_1. Παίρνουμε μία-μία τις τιμές του D_1 και ψάχνουμε «στηρίγματα» (ή, πιο επίσημα, «τιμές υποστήριξης» (support values)) στο $D_2 \times D_3$. Για την τιμή $v_1 = 1$ έχουμε την «πλειάδα υποστήριξης» $\langle v_2, v_3\rangle = \langle 1, 2\rangle$, αφού ισχύει $v_1 + v_2 = v_3$.[4] Για $v_1 = 2$ έχουμε την $\langle 1, 3\rangle$, για $v_1 = 3$ την $\langle 1, 4\rangle$, ..., για $v_1 = 8$ την $\langle 1, 9\rangle$. Για $v_1 = 9$ όμως, δεν έχουμε κάποια πλειάδα υποστήριξης, συνεπώς προκαλείται μία ασυνέπεια.

Συνέπεια Ορίων

Η *συνέπεια ορίων* (bounds consistency) είναι μία πιο χαλαρή έννοια σε σχέση με τη συνέπεια ακμών, αλλά συνάμα πολύ χρήσιμη, αφού η ύπαρξή της μπορεί να ελεγχθεί γρηγορότερα. Σε αυτή την εργασία χρησιμοποιούμε πολλούς περιορισμούς ανώτερης τάξης, οπότε βολεύει να περιγράψουμε τη συνέπεια ορίων έτσι ώστε να τους καλύπτει και αυτούς.

Η (γενικευμένη) συνέπεια ορίων, λοιπόν, ανάμεσα σε μία μεταβλητή V_i και έναν περιορισμό στον οποίο εμπλέκονται οι μεταβλητές $\{V_1, \ldots, V_{i-1}, V_i, V_{i+1}, \ldots, V_k\}$ ισχύει αν για καθένα από τα δύο άκρα του D_i, δηλαδή τα $\min(D_i)$

[4] Εκτός από τη $\langle v_2, v_3\rangle = \langle 1, 2\rangle$, έχουμε και τις πλειάδες υποστήριξης $\langle 2, 3\rangle$, $\langle 3, 4\rangle$, ..., $\langle 8, 9\rangle$, αλλά εμείς αρκεί να βρούμε μία μόνο πλειάδα υποστήριξης.

και $\max(D_i)$, *υπάρχει «πλειάδα υποστήριξης»* $\langle v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k \rangle$ *στο* $D_1 \times \cdots \times D_{i-1} \times D_{i+1} \times \cdots \times D_k$.[5]

Είναι σημαντικό ότι η συνέπεια ορίων παρουσιάζει όμοια συμπεριφορά με τη συνέπεια ακμών στα εξής δύο κρίσιμα σημεία: α) όταν κάποια μεταβλητή έχει κενό πεδίο τιμών στην κατάσταση συνέπειας ορίων, τότε το πρόβλημα δεν έχει λύση και β) όταν όλες οι μεταβλητές έχουν μονομελή πεδία στην κατάσταση συνέπειας ορίων, τότε η συγκεκριμένη ανάθεση είναι και λύση του προβλήματος.

Ωστόσο, η συνέπεια ακμών προκαλεί περισσότερα κλαδέματα στο δένδρο αναζήτησης, αφού αφαιρεί περισσότερες τιμές από τα πεδία τιμών των μεταβλητών, έτσι ώστε να τα φέρει σε συνέπεια. Παρόλα αυτά έχει ένα επιπλέον χρονικό κόστος, όσον αφορά την επιβολή της, σε σχέση με τη συνέπεια ορίων. Οπότε ένα θέμα προς συζήτηση είναι να βρεθεί η χρυσή τομή στο εξής *«Θέλουμε μικρότερο δένδρο αναζήτησης ή ταχύτερη εξέταση της συνέπειας ενός κόμβου του;»*. (Η απάντηση σε αυτήν, όπως και σε πολλές τέτοιου είδους ερωτήσεις είναι κοινότυπη: *«Εξαρτάται από το πρόβλημα που θέλουμε να επιλύσουμε»*.)

Παράδειγμα. Ας ξαναδούμε το παράδειγμα της προηγούμενης παραγράφου, δηλαδή τον περιορισμό $V_1 + V_2 = V_3$, με $D_1 = D_2 = D_3 = [1..9]$. Στη γενική συνέπεια ακμών εξετάσαμε όλες τις τιμές του D_1 για να αποδείξουμε την ασυνέπεια, ενώ στη συνέπεια ορίων αρκεί να ελέγξουμε τι γίνεται για την ελάχιστη τιμή $\min(D_1) = 1$ και για τη μέγιστη $\max(D_1) = 9$. Με δύο μόλις ελέγχους θα δείχναμε την ασυνέπεια (που προκαλείται από τη $\max(D_1)$, όπως είδαμε στην προηγούμενη παράγραφο).

[5] Υπάρχει και μια ακόμα –ελαφρώς χαλαρότερη– εκδοχή της συνέπειας ορίων που λέει ότι η πλειάδα υποστήριξης $\langle v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k \rangle$ πρέπει να ανήκει στο $[\underline{V}_1..\overline{V}_1] \times \cdots \times [\underline{V}_{i-1}..\overline{V}_{i-1}] \times [\underline{V}_{i+1}..\overline{V}_{i+1}] \times \cdots \times [\underline{V}_k..\overline{V}_k]$ (με $\underline{V}_i$ και $\overline{V}_i$ συμβολίζουμε την ελάχιστη και τη μέγιστη τιμή του πεδίου τιμών D_i, αντίστοιχα) [7]. Δεν θα ασχοληθούμε στη συνέχεια με αυτήν.

Κεφάλαιο 3

Η Μηχανή Διάδοσης Περιορισμών AC-5+

Η διάδοση περιορισμών είναι μια μορφή εξαγωγής συμπεράσματος, όχι αναζήτησης, και ως τέτοια είναι περισσότερο «ικανοποιητική», τόσο τεχνικά όσο και αισθητικά.

— Eugene C. Freuder, 2005.

Οι αλγόριθμοι της οικογένειας AC εμφανίζονται συνήθως σε θεωρητικές εργασίες. Στα πρακτικά συστήματα Προγραμματισμού με Περιορισμούς όμως δεν έχουν τόση απήχηση· συνήθως σε αυτά υλοποιούνται οι λεγόμενες *μηχανές διάδοσης περιορισμών*. Σε αυτό το κεφάλαιο θα δούμε πώς ένας νέος αλγόριθμος AC-5+ μπορεί να είναι ταυτόχρονα και μία περίπτωση μηχανής διάδοσης περιορισμών υλοποιημένης σε έναν επιλυτή γενικών προβλημάτων ικανοποίησης περιορισμών.

3.1 Γνωστοί Αλγόριθμοι Επιβολής Συνέπειας Ακμών

Στην § 2.3.4 παρουσιάστηκαν δύο αλγόριθμοι επιβολής συνέπειας ακμών: ο AC-1 και ο AC-3. Προτάθηκαν από τον Mackworth, αρχικά σε μία τεχνική έκθεση το 1975 [25] και πιο επίσημα σε μία δημοσίευση το 1977 [26]. Ο AC-3 αποτελεί ουσιαστικά μία βελτιωμένη έκδοση των AC-1 και AC-2, οι οποίοι αναφέρονται πλέον μόνο για ιστορικούς και διδακτικούς λόγους. Ακόμα και σήμερα –τρεις δεκαετίες μετά–, ενώ έχουν προταθεί και υλοποιηθεί δεκάδες αλγόριθμοι

επιβολής συνέπειας ακμών, τα περισσότερα συστήματα επίλυσης ΠΙΠ χρησιμοποιούν τον AC-3 σαν βάση της *μηχανής διάδοσης περιορισμών* (propagation engine) που ενσωματώνουν, ελλείψει ενός άλλου γενικώς αποδεκτού πλαισίου, όπως ομολογεί ο Bessière σε μία πρόσφατη σχετική επισκόπηση [7].

3.1.1 Αλγόριθμοι Αδρής Υφής

Ο AC-3 κατατάσσεται στη λεγόμενη κατηγορία αλγορίθμων *αδρής υφής* (coarse-grained). Πρόκειται για την κατηγορία αλγορίθμων επιβολής συνέπειας ακμών, στους οποίους κεντρικό ρόλο παίζουν οι έλεγχοι περιορισμών, οι οποίοι «πυροδοτούνται» από μία αφαίρεση τιμής. Με άλλα λόγια, σχηματικά, μπορούμε να πούμε ότι το «καλώδιο» μέσα από το οποίο περνάει το «μήνυμα» της μεταβολής ενός πεδίου τιμών σε αυτή την κατηγορία αλγορίθμων είναι οι *περιορισμοί,* δηλαδή οι ακμές του δικτύου περιορισμών.

Το 2001 οι Bessière και Régin, καθώς και οι Zhang και Yap πρότειναν μέσα από ανεξάρτητες δουλειές [10, 39] μία βελτίωση του αλγορίθμου AC-3. Την ίδια χρονιά, οι τέσσερις ερευνητές δημοσίευσαν μία κοινή εργασία και ονόμασαν αυτόν τον αλγόριθμο AC-2001 [9]. (Ο AC-2001 είχε ονομαστεί προσωρινά και AC-3.1. Αυτός είναι ο λόγος που κάποιες βελτιώσεις του AC-2001 ονομάστηκαν AC-3.2 και AC-3.3 [23].)

3.1.2 Αλγόριθμοι Λεπτής Υφής

Οι λεγόμενοι αλγόριθμοι *λεπτής υφής* (fine-grained) συνιστούν τη δεύτερη –και τελευταία– κατηγορία αλγορίθμων επιβολής συνέπειας ακμών. Σχηματικά, τα «καλώδια» μέσα από τα οποία οι αλγόριθμοι αυτοί μεταφέρουν το «μήνυμα» της μεταβολής ενός πεδίου τιμών, είναι οι ίδιες οι *τιμές* των πεδίων τιμών· εξάλλου, κάθε τιμή ενός πεδίου τιμών συνδέεται μέσω μίας δομής δεδομένων (π.χ. μέσω ενός πίνακα) με όλες τις τιμές των άλλων μεταβλητών στις οποίες στηρίζεται.

Πρακτικά, ενώ οι αλγόριθμοι αδρής υφής εστιάζουν στο δίκτυο περιορισμών, οι αλγόριθμοι λεπτής υφής «προχωρούν ένα βήμα παρακάτω», αφού επεξεργάζονται ένα δίκτυο στο οποίο κάθε τιμή ενός πεδίου τιμών μίας μεταβλητής συσχετίζεται με τις τιμές των υπολοίπων μεταβλητών πάνω στις οποίες στηρίζεται.[1] Οι γνωστότεροι αλγόριθμοι της κατηγορίας είναι ο AC-4 [29], ο AC-6

[1] Π.χ. στον περιορισμό $X = Y + 5$ η τιμή 2 του πεδίου τιμών της Y στηρίζεται στην τιμή 7 του πεδίου τιμών της X (αφού $7 = 2 + 5$). Οι αλγόριθμοι λεπτής υφής εστιάζουν στη χρήση δομών δεδομένων για την αποθήκευση των τιμών υποστήριξης (support values) για καθεμία τιμή κάθε πεδίου τιμών.

[6] και ο AC-7 [8].

3.1.3 Παραμετρικοί Αλγόριθμοι

Κλείνουμε την παρουσίαση των γνωστότερων αλγορίθμων επιβολής συνέπειας ακμών, παρουσιάζοντας τον AC-5 [37]. Ο αλγόριθμος αυτός δεν εντάσσεται απευθείας σε κάποια από τις δύο κατηγορίες αλγορίθμων που αναφέρθηκαν στις προηγούμενες ενότητες. Αυτό συμβαίνει επειδή πρόκειται για έναν παραμετρικό αλγόριθμο: για κάθε περιορισμό ορίζονται –από αυτόν που καλεί τον αλγόριθμο– συγκεκριμένες υπορουτίνες για την αντιμετώπιση του περιορισμού αυτού.

Εφόσον ο αλγόριθμος δέχεται υπορουτίνες που ορίζονται από τον χρήστη, είναι δυνατόν να προσομοιωθούν μέσω αυτού τόσο ο AC-3, όσο και ο AC-4, καθώς και υβρίδια.

3.2 Ο Αλγόριθμος AC-5

Παρακάτω παρουσιάζεται ο αρχικός ορισμός του AC-5 [37].[2]

```
 1: procedure AC-5
 2:     Q ← ∅
 3:     for each (i, j) ∈ arcs(G) do
 4:         ArcCons(i, j, Δ)
 5:         Q ← Q ∪ {((k, i), w) | (k, i) ∈ arcs(G), k ≠ j}
 6:         D_i ← D_i − Δ
 7:     end for
 8:     while Q ≠ ∅ do
 9:         Pick and remove an ((i, j), w) out of Q
10:         LocalArcCons(i, j, w, Δ)
11:         Q ← Q ∪ {((k, i), w) | (k, i) ∈ arcs(G), k ≠ j}
12:         D_i ← D_i − Δ
13:     end while
14: end procedure
```

[2] Ο αρχικός ορισμός έχει υποστεί μία μικρή τροποποίηση-βελτίωση στις γραμμές 5 και 11, όπου έχει μπει η συνθήκη $k \neq j$. Ανάλογη συνθήκη υπάρχει στον ορισμό του AC-3.

Δεν υπάρχει άμεσος ορισμός για τις υπορουτίνες ArcCons και LocalArcCons· ο προγραμματιστής που καλεί τον AC-5 πρέπει να ορίσει ο ίδιος τις δύο υπορουτίνες αυτές για κάθε περιορισμό (ή, καλύτερα, για κάθε τύπο περιορισμού). Αυτός είναι άλλωστε ο λόγος για τον οποίο ο AC-5 χαρακτηρίζεται ως παραμετρικός.

Παρόλα αυτά, είναι απαραίτητο να ξεκαθαριστούν οι προδιαγραφές που οφείλουν να πληρούν οι δύο υπορουτίνες:

- Η ArcCons(i, j, Δ), θα πρέπει να φέρνει σε συνέπεια την ακμή (i, j). Κατά τη διαδικασία αυτή, όποιες τιμές αφαιρεθούν από το πεδίο D_i, θα πρέπει να μπουν στο σύνολο Δ –το οποίο κατά την έναρξη της εκτέλεσης της υπορουτίνας είναι κενό.

- Η LocalArcCons(i, j, w, Δ) θα πρέπει να φέρνει σε συνέπεια την ακμή (i, j) –αφαιρώντας τιμές από το D_i και βάζοντάς τις στο κενό αρχικά σύνολο Δ– *δεδομένης της αφαίρεσης της τιμής* w *από το πεδίο* D_j. Το σημείο που θα πρέπει να προσεχθεί εδώ, είναι ότι η LocalArcCons δεν απαιτείται να εξασφαλίσει τη συνέπεια της ακμής (i, j). Αυτό που πρέπει να κάνει είναι να αφαιρέσει τις τιμές εκείνες του D_i, οι οποίες είχαν μοναδικό στήριγμα στη μεταβλητή j, την τιμή w.

Είναι δυνατόν, να ορίσουμε τη LocalArcCons έτσι:

procedure LocalArcCons(i, j, w, Δ)
  ArcCons(i, j, Δ)
end procedure

Αν χρησιμοποιήσουμε τον παραπάνω ορισμό, κατ΄ ουσίαν αγνοούμε το w και τελικά ο AC-5 ταυτίζεται με τον AC-3.

Πράγματι, αν έχουμε για μία ακμή (i, j) «αφηρημένους» περιορισμούς του τύπου $C_{ij} = \{(1, 25), (10, 7), (201, 99), (5, 4)\}$, αυτό το w δεν θα είναι ιδιαίτερα χρήσιμο. Για τέτοιες περιπτώσεις περιορισμών, ο ιδανικός αλγόριθμος θα ήταν ο AC-2001.

Στην πράξη, όμως, οι περιορισμοί που συνήθως τίθενται μεταξύ δύο ή περισσοτέρων μεταβλητών είναι «σαφέστεροι». Π.χ. είναι της μορφής $C_{ij} = \{(x, y) \mid x = y + 1, x \in D_i, y \in D_j\}$. Συνηθισμένοι περιορισμοί που διατυπώνονται σε έναν επιλυτή, είναι οι *συναρτησιακοί* (functional), όπως μία σχέση $V_i = 4V_j + 7$, μεταξύ των μεταβλητών V_i και V_j, *αντι-συναρτησιακοί* (anti-functional), όπως η ανισότητα $V_i \neq V_j$ και *μονότονοι* (monotonic) όπως το $V_i < 8V_j$.

Για τις παραπάνω κατηγορίες περιορισμών –καθώς και για αρκετές άλλες– μπορούμε να ορίσουμε την LocalArcCons κατά τέτοιο τρόπο, ώστε να έχει σταθερή πολυπλοκότητα. Π.χ. για τον περιορισμό $V_i = 2V_j + 1$, αν κληθεί η LocalArcCons$(i, j, 5, \Delta)$ –δηλαδή αν έχουμε $w = 5$– μπορούμε να δούμε σε $O(1)$ βήματα αν $(2 \cdot 5 + 1) \in D_i$, ενώ η αντίστοιχη κλήση ArcCons(i, j, Δ), θα έπαιρνε χρόνο $O(|D_i|)$.

Και ενώ η λογική του αλγορίθμου AC-5 είναι ίδια με αυτή του AC-3, *η χρήση στον AC-5 προσαρμοσμένων συναρτήσεων (*ArcCons *και* LocalArcCons*) για τον έλεγχο της συνέπειας ενός περιορισμού αποδεικνύεται καλύτερη από τη χρήση της γενικής συνάρτησης (*Revise*) του AC-3.*

Το τίμημα ωστόσο για αυτή τη βελτίωση είναι ότι για κάθε τύπο περιορισμού οφείλουμε να ορίζουμε τις μεθόδους του ArcCons και LocalArcCons, όπως θα κάνουμε ενδεικτικά για κάποιους περιορισμούς παρακάτω, σε αυτό το κεφάλαιο. Δηλαδή, ναι μεν εκμεταλλευόμαστε τα χαρακτηριστικά και τις ιδιαιτερότητες συγκεκριμένων κατηγοριών περιορισμών, έτσι ώστε να τους αντιμετωπίζουμε γρηγορότερα, αλλά η αλγοριθμική υλοποίηση αυτής της γνώσης απαιτεί κάποιον επιπλέον κόπο.

3.3 Μία Πρώτη Προσαρμογή του AC-5

Στη συνέχεια παρουσιάζουμε μία πρώτη τροποποίηση του AC-5:

procedure AC-5-NewQueue
  $Q \leftarrow \emptyset$
  for each $(i, j) \in \text{arcs}(G)$ **do**
    ArcCons(i, j, Δ)
    $Q \leftarrow Q \cup \{(i, (i, j), w) \mid w \in \Delta\}$
    $D_i \leftarrow D_i - \Delta$
  end for
  while $Q \neq \emptyset$ **do**
    Pick and remove an $(i, (i, j), w)$ out of Q
    for each $(k, i) \in \text{arcs}(G)$, with $k \neq j$ **do**
      LocalArcCons(k, i, w, Δ)
      $Q \leftarrow Q \cup \{(k, (k, i), w) \mid w \in \Delta\}$
      $D_k \leftarrow D_k - \Delta$
    end for
  end while

end procedure

Παρατηρούμε ότι το Q περιέχει τώρα άλλου τύπου στοιχεία. Ενώ στον AC-5 το Q αποτελούνταν από στοιχεία $((i,j),w)$, που σήμαινε ότι έπρεπε να ελεγχθεί η συνέπεια της ακμής (i,j), δεδομένης της απουσίας του w από το D_j, στον AC-5-NewQueue το Q έχει στοιχεία $(i,(i,j),w)$, που σημαίνει ότι θα πρέπει να ελεγχθεί η συνέπεια όλων των ακμών, που κατευθύνονται προς τη μεταβλητή i, δεδομένης της απουσίας του w από D_i [28, 13]. Η ακμή (j,i) δεν χρειάζεται να ελεγχθεί. Ο νέος τύπος λοιπόν των στοιχείων του Q, περιέχει περισσότερες πληροφορίες.[3]

Συνοπτικά, έχουμε την εξής αλλαγή στον τύπο των στοιχείων της ουράς.

Αλγόριθμος	**Τύπος Στοιχείων Q**	**Ερμηνεία**
AC-5	$((i,j),w)$	Πρέπει να κληθεί η LocalArcCons(i,j,w,Δ).
AC-5-NewQueue	$(i,(i,j),w)$	Πρέπει να κληθεί η LocalArcCons(k,i,w,Δ) για κάθε $(k,i) \in \text{arcs}(G)$, με $k \neq j$.

3.3.1 Περισσότερες Τροποποιήσεις

Ο αλγόριθμος AC-5-NewQueue που παρουσιάστηκε προηγουμένως, «περιστρέφεται» γύρω από τις ακμές (i,j) του γράφου που αναπαριστά το δίκτυο των περιορισμών. Τώρα θα τον μετασχηματίσουμε έτσι ώστε να εστιάζει απευθείας στους ίδιους τους περιορισμούς. Έστω ότι ένας περιορισμός $c \in \mathscr{C}$ συνδέει τις μεταβλητές i και j. Συνεπώς οι δύο ακμές (i,j) και (j,i) μπορούν να αντιπροσωπευθούν από τον περιορισμό c.

```
1: procedure AC-5-Final
2:     Q ← ∅
3:     for each c ∈ 𝒞 do
4:         c.ArcCons(Q)
5:     end for
6:     while Q ≠ ∅ do
7:         Pick and remove an (i,(i,j),w) out of Q
8:         for each c ∈ 𝒞, with i ∈ vars(c) and c ≠ (i,j) do
9:             c.LocalArcCons(i,w,Q)
```

[3] Ο παράγοντας εξοικονόμησης μνήμης για την αποθήκευση της ουράς Q ισούται με τον μέσο όρο των βαθμών των κόμβων του G.

10: **end for**
11: **end while**
12: **end procedure**

Για να δείξουμε ότι ο παραπάνω αλγόριθμος είναι ισοδύναμος με τον AC-5-NewQueue, θα πρέπει να ορίσουμε τις δύο παραμετρικές μεθόδους. Με c.ArcCons(Q) και c.LocalArcCons(v, w, Q) συμβολίζουμε αντίστοιχα τις ArcCons(c, Q) και LocalArcCons(c, v, w, Q). Προτιμήσαμε όμως να βάλουμε το πρόθεμα «c.» πριν το όνομα των μεθόδων, για να τονιστεί ότι οι δύο μέθοδοι πρέπει να οριστούν για κάθε περιορισμό c. Επίσης, θελήσαμε έτσι να επισημάνουμε ότι ο αλγόριθμος είναι κατά κάποιον τρόπο αντικειμενοστρεφής και ότι τα αντικείμενα στα οποία εστιάζει είναι όλοι οι περιορισμοί στο $\mathscr{C}$.

Έστω ότι οι ακμές που περιλαμβάνονται στον περιορισμό c εντάσσονται στο σύνολο arcs(c). Η c.ArcCons(Q) θα πρέπει λοιπόν να συμπεριλάβει τόσες κλήσεις της ArcCons (όπως αυτή χρησιμοποιείται στον AC-5-NewQueue), όσες είναι οι ακμές του arcs(c):

procedure c.ArcCons(Q)
  for each $(i, j) \in \text{arcs}(c)$ **do**
    ArcCons(i, j, Δ)
    $Q \leftarrow Q \cup \{(i, (i, j), w) \mid w \in \Delta\}$
    $D_i \leftarrow D_i - \Delta$
  end for
end procedure

Ή, πιο απλά, για arcs$(c) = \{(p, q), (q, p)\}$ έχουμε:

procedure c.ArcCons(Q)
  ArcCons(p, q, Δ)
  $Q \leftarrow Q \cup \{(p, (p, q), w) \mid w \in \Delta\}$
  $D_p \leftarrow D_p - \Delta$
  ArcCons(q, p, Δ)
  $Q \leftarrow Q \cup \{(q, (q, p), w) \mid w \in \Delta\}$
  $D_q \leftarrow D_q - \Delta$
end procedure

Σημειώνουμε πάλι, προς αποφυγή παρεξηγήσεων, ότι η c.ArcCons είναι προφανώς διαφορετική μέθοδος από την ArcCons που χρησιμοποιείται στον AC-5-NewQueue, αλλά μπορεί να εκφραστεί βάσει αυτής.

Ενώ με τη c.ArcCons αναφερόμαστε και στις *δύο* ακμές του c (όταν ο c είναι δυαδικός περιορισμός), δηλαδή τις (p, q) και (q, p), με τη c.LocalArcCons αναφερόμαστε σε *μία* από τις δύο ακμές. Ως εκ τούτου, το πρώτο όρισμα της c.LocalArcCons έχει να κάνει με τη μεταβλητή v στην οποία καταλήγει η ακμή στην οποία επιθυμούμε να επιβάλλουμε συνέπεια.

procedure c.LocalArcCons(v, w, Q)
  for each $(i, j) \in \text{arcs}(c)$, with $i \neq v$ **do**
    LocalArcCons(i, j, w, Δ)
    $Q \leftarrow Q \cup \{(i, (i, j), w) \mid w \in \Delta\}$
    $D_i \leftarrow D_i - \Delta$
  end for
end procedure

Το οποίο για $\text{arcs}(c) = \{(p, q), (q, p)\}$ γίνεται:

procedure c.LocalArcCons(v, w, Q)
  if $v = p$ **then**
    LocalArcCons(q, p, w, Δ)
    $Q \leftarrow Q \cup \{(q, (q, p), w) \mid w \in \Delta\}$
    $D_q \leftarrow D_q - \Delta$
  else ▷ $v = q$
    LocalArcCons(p, q, w, Δ)
    $Q \leftarrow Q \cup \{(p, (p, q), w) \mid w \in \Delta\}$
    $D_p \leftarrow D_p - \Delta$
  end if
end procedure

Στην πράξη, δεν υλοποιούμε τις δύο παραπάνω μεθόδους κατά τον τρόπο με τον οποίο παρουσιάστηκαν. Αυτή η παρουσίαση ήταν θεωρητική επί το πλείστον, για να αποδειχθεί η ισοδυναμία του AC-5-Final με τον AC-5.

Ωστόσο, αυτοί οι θεωρητικοί ορισμοί μάς καθοδηγούν στο να περιγράψουμε τις ιδιότητες που οφείλουν οι δύο μέθοδοι να πληρούν.

- Η c.ArcCons οφείλει να επιβάλλει *διαδοχικά* συνέπεια ακμών για τις ακμές του περιορισμού c.[4]

[4] Υπάρχει μία λεπτομέρεια που θα πρέπει να προσέξουμε εδώ. Ας πάρουμε το σενάριο στο οποίο $\text{vars}(c) = \{p, q\}$. α) Η c.ArcCons αρχικά επιβάλλει συνέπεια στην ακμή (p, q). β) Στη συνέχεια θα επιβάλει συνέπεια στην (q, p). Και εκεί τερματίζει. *Αυτό* δεν *σημαίνει*

- Η c.LocalArcCons(v, w, Q) οφείλει να φέρνει σε συνέπεια τις ακμές εκείνες του c που δεν ξεκινάνε από τη μεταβλητή v (δηλαδή τις ακμές εκείνες που *κατευθύνονται* προς το v), δεδομένης της απουσίας της τιμής w από το D_v, όπως ακριβώς και η LocalArcCons του AC-5.[5]

Επισημαίνεται ότι και στις δύο μεθόδους έχουν ανατεθεί πλέον «αρμοδιότητες» να αφαιρούν οι ίδιες τις ασυνεπείς τιμές από τα πεδία τιμών –και όχι να τις βάζουν απλά στο ενδιάμεσο σύνολο Δ. Επίσης, είναι «υπεύθυνες» για την εισαγωγή στοιχείων στο σύνολο Q. Η λειτουργία της εισαγωγής στοιχείων στην ουρά είναι δυνατόν για λόγους απλότητας αλλά και «ομοιομορφίας» της υλοποίησης να ενσωματωθεί στη μέθοδο αφαίρεσης τιμών από το πεδίο τιμών μίας μεταβλητής v (το όνομα της μεθόδου θα μπορούσε να είναι π.χ. «v.remove»). Μία τέτοια μέθοδος θα είναι επίσης δυνατόν να εγείρει μία σημαία ασυνέπειας, όταν το πεδίο τιμών της μεταβλητής στην οποία αναφέρεται γίνει κενό (γεγονός που σημαίνει ότι όλο το δίκτυο περιορισμών είναι ασυνεπές).

3.3.2 Γενικευμένη Συνέπεια Ακμών

Στην § 2.3.4 αναφερθήκαμε στην έννοια της γενικευμένης συνέπειας ακμών (GAC), που είναι χρήσιμη όταν έχουμε να ασχοληθούμε –πέρα από τους δυαδικούς– και με καθολικούς περιορισμούς.

Το 1977 ο Mackworth επέκτεινε με έναν πολύ φυσικό και απλό τρόπο τον αλγόριθμό του AC-3 ορίζοντας τον GAC-3, καταφέρνοντας έτσι να εκμεταλλευτεί τους καθολικούς περιορισμούς [27]. Πριν επιχειρήσουμε και εμείς το ίδιο για τον AC-5-Final, θα επισημάνουμε πρώτα μία «ασάφειά» του στη γραμμή 8: εκεί υπάρχει η ανισότητα $c \neq (i, j)$. Με αυτή την ανισότητα θέλουμε να εξασφαλίσουμε ότι $(p, q) \neq (j, i)$ και $(q, p) \neq (j, i)$, αν $\text{arcs}(c) = \{(p, q), (q, p)\}$. (Ο λόγος που κάνουμε αυτόν τον έλεγχο είναι επειδή δεν χρειάζεται να ελέγξουμε τον περιορισμό που προκάλεσε τη διάδοση περιορισμών –όπως γίνεται

ότι μετά τον τερματισμό όλες οι ακμές του c βρίσκονται σε κατάσταση συνέπειας.

Π.χ. η (p, q) που βρισκόταν σε κατάσταση συνέπειας μετά το βήμα α), δεν είναι σίγουρο ότι θα βρίσκεται στην ίδια κατάσταση μετά το βήμα β), αφού το τελευταίο βήμα μπορεί να προκάλεσε αλλαγές στο πεδίο τιμών της q, οι οποίες να έφεραν την (p, q) σε κατάσταση ασυνέπειας.

Συμπερασματικά, αυτό που η c.ArcCons οφείλει να κάνει, είναι να επιβάλλει *διαδοχικά* συνέπεια στις ακμές του c.

[5] Χρησιμοποιήσαμε πληθυντικό (στη φράση «φέρνει σε συνέπεια *τις* ακμές»), γιατί παρακάτω θα δούμε την εφαρμογή του AC-5 σε καθολικούς περιορισμούς. Όταν έχουμε να κάνουμε με δυαδικούς περιορισμούς, τότε η c.LocalArcCons φέρνει σε σε συνέπεια μία από τις δύο ακμές του c, δηλαδή είτε την (p, q), είτε την (q, p).

και στον AC-3. Η ακμή που προκαλεί τη διάδοση περιορισμών είναι η (i, j). Συνεπώς ούτε η ακμή (j, i) χρειάζεται να ελεγχθεί.)

Αν θέσουμε $\text{arcs}(c_f) = \{(i, j), (j, i)\}$,[6] μπορούμε την ανισότητα της γραμμής 8 να την ξαναγράψουμε ισοδύναμα σαν $c \neq c_f$. Αυτό μπορούμε να το ισχυριστούμε και στην περίπτωση που υπάρχουν καθολικοί περιορισμοί.

Μετά και από την παραπάνω αλλαγή είμαστε σε θέση να τροποποιήσουμε και τον τύπο των στοιχείων της ουράς Q, έτσι ώστε να υποστηρίζει και αυτός τους καθολικούς περιορισμούς. (Όλος ο υπόλοιπος αλγόριθμος AC-5-Final, καθώς και οι προδιαγραφές των μεθόδων c.ArcCons και c.LocalArcCons τους υποστηρίζουν ήδη.) Ο τρέχων τύπος είναι ο $(i, (i, j), w)$. Τον τροποποιούμε τώρα σε (i, c_f, w), με $(i, j) \in \text{arcs}(c_f)$. Έχουμε μία γενίκευση δηλαδή, αλλά παρόλα αυτά δεν χάνουμε κάποια χρήσιμη πληροφορία, αφού ξέρουμε ότι η ακμή που προκάλεσε την αφαίρεση της τιμής w από το D_i, ξεκινάει από το i και ανήκει στο $\text{arcs}(c_f)$.[7]

Δεδομένων των προδιαγραφών των δύο μεθόδων c.ArcCons και c.LocalArcCons, η διατύπωση του αλγορίθμου επιβολής *γενικευμένης* συνέπειας ακμών γίνεται απρόσμενα απλή:

```
 1: procedure GAC-5
 2:     Q ← ∅
 3:     for each c ∈ 𝒞 do
 4:         c.ArcCons(Q)
 5:     end for
 6:     while Q ≠ ∅ do
 7:         Pick and remove an (i, c_f, w) out of Q
 8:         for each c ∈ 𝒞, with i ∈ vars(c) and c ≠ c_f do
 9:             c.LocalArcCons(i, w, Q)
10:         end for
11:     end while
12: end procedure
```

[6] Ο δείκτης «f» προκύπτει από το «fired», δηλαδή «c_f» σημαίνει «ο περιορισμός που "πυροδότησε" τη διάδοση περιορισμών».

[7] Θεωρούμε ότι δύο ακμές που ενώνουν τους ίδιους κόμβους-μεταβλητές και ξεκινούν από τον ίδιο κόμβο-αφετηρία ταυτίζονται. Π.χ. $(V_1, V_2, V_3) \equiv (V_1, V_3, V_2)$. Η επιβολή συνέπειας, εξάλλου, στην πρώτη ακμή ισοδυναμεί με την επιβολή συνέπειας και στη δεύτερη.

3.4 Παραδείγματα Συναρτήσεων ArcCons και LocalArcCons

Οι αναφορές μας στην οικογένεια αλγορίθμων AC-5 περιστράφηκαν γύρω από τη χρήση και τις ιδιότητες των μεθόδων c.ArcCons και c.LocalArcCons. Δεν έχουμε ορίσει ωστόσο ακόμα κάποια από αυτές. Για κάθε περιορισμό εξάλλου οι μέθοδοι αυτές ορίζονται διαφορετικά.

Πριν παρουσιάσουμε τους αλγορίθμους για συγκεκριμένους περιορισμούς, ορίζουμε τη μέθοδο αφαίρεσης μίας τιμής val από το πεδίο τιμών μίας μεταβλητής V_f, η οποία προκλήθηκε από μία μέθοδο c_f.ArcCons ή c_f.LocalArcCons. Η μέθοδος αυτή φροντίζει να ενημερωθεί κατάλληλα και η ουρά Q.

procedure V_f.remove(val, Q, c_f)
  $D_{V_f} \leftarrow D_{V_f} - \{val\}$ ▷ Αν $D_{V_f} = \emptyset$, το πρόβλημα δεν έχει λύση.
  $Q \leftarrow Q \cup \{(V_f, c_f, val)\}$
end procedure

3.4.1 Ο Περιορισμός $X = Y$

Παρακάτω ορίζουμε τις μεθόδους c_1.ArcCons και c_1.LocalArcCons για έναν περιορισμό c_1 ο οποίος εκφράζει την ισότητα μεταξύ δύο περιορισμένων μεταβλητών X και Y. Ο περιορισμός αυτός συμβολίζεται και σαν «$X = Y$».

procedure c_1.ArcCons(Q)
  for each $val \in D_X$ **do**
    if $val \notin D_Y$ **then**
      X.remove(val, Q, c_1)
    end if
  end for
  for each $val \in D_Y$ **do**
    if $val \notin D_X$ **then**
      Y.remove(val, Q, c_1)
    end if
  end for
end procedure

Οι προδιαγραφές μίας οποιασδήποτε c_1.ArcCons –όπως διατυπώθηκαν– απαιτούν τη διαδοχική εφαρμογή συνέπειας ακμών για κάθε ακμή στο arcs(c_1). Ως εκ τούτου, στον παραπάνω αλγόριθμο βλέπουμε ότι στον πρώτο βρόχο (**for**)

έρχεται σε συνέπεια η ακμή (X, Y), ενώ μετά τον δεύτερο βρόχο η ακμή (Y, X) έρχεται σε συνέπεια.

Η c_1.LocalArcCons είναι πιο εξειδικευμένη συνάρτηση, αφού παίρνει δύο επιπλέον ορίσματα: το V_f[8] και το w. Συνεπώς, καλείται για να φέρει σε συνέπεια την/τις ακμή/ές του συνόλου arcs(c_1) που κατευθύνεται/ονται προς τη V_f, *δεδομένης της αφαίρεσης της τιμής* w *από το πεδίο τιμών* D_{V_f}.

procedure c_1.LocalArcCons(V_f, w, Q)
  if $X \equiv V_f$ **then**
    if $w \in D_Y$ **then**
      Y.remove(w, Q, c_1)
    end if
  else ▷ $Y \equiv V_f$
    if $w \in D_X$ **then**
      X.remove(w, Q, c_1)
    end if
  end if
end procedure

Με τη c_1.LocalArcCons, εξασφαλίζουμε ότι μετά την αφαίρεση μίας τιμής w από το πεδίο της μίας μεταβλητής (δηλαδή της V_f), θα αφαιρεθεί η ίδια τιμή από το πεδίο της άλλης μεταβλητής, εφόσον βέβαια ανήκει σε αυτό.

Σε αυτόν τον περιορισμό φαίνεται ξεκάθαρα η διαφορά στην τάξη πολυπλοκότητας, μεταξύ των c_1.ArcCons και c_1.LocalArcCons. Ουσιαστικά, δρέπουμε τους καρπούς της απόφασής μας να χρησιμοποιήσουμε τον AC-5.

3.4.2 Ο Περιορισμός $X \neq Y$

Σε αυτήν την παράγραφο εξετάζουμε τις μεθόδους για τον περιορισμό c_2 της ανισότητας μεταξύ δύο μεταβλητών X και Y, που συμβολίζεται και σαν «$X \neq Y$». Ο περιορισμός αυτός απαγορεύει την ανάθεση της ίδιας τιμής και στις δύο μεταβλητές.

Όταν μία μεταβλητή V είναι δεσμευμένη, τότε τη μοναδική τιμή του πεδίου τιμών D_V τη συμβολίζουμε με «V.value» (και ισχύει ότι $D_V = \{V.\text{value}\}$).

procedure c_2.ArcCons(Q)
  if $|D_Y| = 1$ **and** Y.value $\in D_X$ **then**

[8] Όπως και στο «c_f», ο δείκτης «f» στη μεταβλητή «V_f» υποδηλώνει ότι η μεταβλητή αυτή «πυροδοτεί» («fires») τη διάδοση περιορισμών.

```
        X.REMOVE(Y.value, Q, c_2)
    end if
    if |D_X| = 1 and X.value ∈ D_Y then
        Y.REMOVE(X.value, Q, c_2)
    end if
end procedure
```

Η c_2.LocalArcCons θα μπορούσε εδώ να ταυτιστεί με την c_2.ArcCons, αφού η ασυμπτωτική πολυπλοκότητα τής τελευταίας είναι ήδη βέλτιστη (και ίση με $O(1)$).

```
procedure c_2.LocalArcCons(V_f, w, Q)
    c_2.ArcCons(Q)
end procedure
```

3.4.3 Μία Παραλλαγή της c_2.LocalArcCons

Έχουμε ήδη ορίσει τη c_2.LocalArcCons για τον περιορισμό της ανισότητας $X \neq Y$, βασιζόμενοι στη μέθοδο c_2.ArcCons. Παρόλα αυτά, θα κάνουμε μία προσπάθεια να εκμεταλλευτούμε την παράμετρο V_f (μεταβλητή «πυροδότησης» διάδοσης περιορισμών), έτσι ώστε να εκτελούνται μόνο οι απαραίτητες εντολές.

```
procedure c_3.LocalArcCons(V_f, w, Q)
    if X ≡ V_f then
        if |D_X| = 1 and X.value ∈ D_Y then
            Y.REMOVE(X.value, Q, c_2)
        end if
    else                                              ▷ Y ≡ V_f
        if |D_Y| = 1 and Y.value ∈ D_X then
            X.REMOVE(Y.value, Q, c_2)
        end if
    end if
end procedure
```

3.5 Κατηγοριοποίηση Μεθόδων LocalArcCons

Στην προηγούμενη ενότητα παρουσιάστηκαν τρία διαφορετικά ζεύγη μεθόδων ArcCons και LocalArcCons για τους περιορισμούς της ισότητας (c_1) και της ανισότητας (c_2 και c_3: ο c_2 διαφέρει από τον c_3 μόνο στη μέθοδο LocalArcCons με την οποία αντιμετωπίζεται).

Με τις μεθόδους ArcCons δεν θα ασχοληθούμε, καθώς καλούνται μία μόνο φορά. Αντίθετα, θα εξετάσουμε πιο προσεκτικά τις ιδιότητες των μεθόδων LocalArcCons, αφού αυτές είναι που καλούνται τις περισσότερες φορές.

Ένα ποσοστό μάλιστα των κλήσεων αυτών γίνεται άσκοπα και έτσι πολλοί ερευνητές έχουν ασχοληθεί με τη μείωσή του. Π.χ. σε αυτήν την κατεύθυνση κινούνται οι –απαιτητικοί σε μνήμη– αλγόριθμοι επιβολής συνέπειας λεπτής υφής.

3.5.1 Συμβάντα που Ενεργοποιούν Συναρτήσεις Διάδοσης

Ένα άλλο –λιγότερο απαιτητικό– γενικό πλαίσιο αλγορίθμων επιβολής συνέπειας ακμών είναι η μηχανή διάδοσης περιορισμών, έτσι όπως ορίζεται από τους Schulte και Carlsson σε μία επισκόπηση περιβαλλόντων Προγραμματισμού με Περιορισμούς [35]. Η επισκόπησή τους εστιάζει σε *συναρτήσεις διάδοσης* (propagators). (Μία τέτοια συνάρτηση είναι και η LocalArcCons.) Μία συνάρτηση διάδοσης καλείται[9] όταν λάβει χώρα κάποιο συγκεκριμένο *συμβάν* (event). Αναφέρονται τρία ήδη συμβάντων (που το ένα είναι υποπερίπτωση του άλλου):

any(X). Το συμβάν αυτό μας ενημερώνει ότι το πεδίο τιμών της περιορισμένης μεταβλητής X έχει υποστεί μία (οποιαδήποτε) μεταβολή. Πρακτικά, έχουμε μία αφαίρεση τιμής από το D_X. Σε μερικές περιπτώσεις, αντί του συμβάντος any(X) χρησιμοποιείται το πιο συγκεκριμένο συμβάν any(X, w) που υποδεικνύει ότι η τιμή w έχει αφαιρεθεί από το D_X [40].

bounds(X). Συμβάν που υποδεικνύει ότι έχει αλλάξει είτε η ελάχιστη τιμή του πεδίου τιμών D_X (το συμβάν τότε ονομάζεται και «minc(X)»), είτε η μέγιστη τιμή του (το συμβάν αυτό λέγεται και «maxc(X)»).

[9] Τη λέξη «καλείται» μπορούμε να την αντικαταστήσουμε με την «πυροδοτείται», ή με την «ενεργοποιείται».

fix(X). Συμβάν που εμφανίζεται μόλις μία μεταβλητή δεσμευτεί (μόλις δηλαδή το πεδίο τιμών της περιέχει μία μόνο τιμή).

Κάθε συμβάν έχει συνδεθεί εκ των προτέρων με τις αντίστοιχες συναρτήσεις διάδοσης. Έτσι, κάθε φορά που λαμβάνει χώρα ένα συμβάν, αυτομάτως μπαίνουν σε μία ουρά αναμονής οι συναρτήσεις διάδοσης περιορισμών με τις οποίες αυτό συνδέεται.

Δηλαδή η μηχανή διάδοσης περιορισμών των Schulte και Carlsson βασίζεται σε μία ουρά από συναρτήσεις διάδοσης. Θεωρητικά, όταν προκύψει κάποιο συμβάν, τότε είναι πιθανόν να έχουμε $O(e)$ εισαγωγές στην ουρά των συναρτήσεων διάδοσης. Μπορεί δηλαδή να χρειαστεί –στη χειρότερη περίπτωση– να κληθούν για τους e περιορισμούς ενός ΠΙΠ, οι αντίστοιχες e συναρτήσεις διάδοσης.

3.5.2 Η Συνάρτηση Διάδοσης LocalArcCons

Στον αλγόριθμο GAC-5 που περιγράφηκε στην παρούσα εργασία, η ουρά Q δεν περιέχει συναρτήσεις διάδοσης. Μπορούμε να πούμε ότι η ουρά του GAC-5 περιέχει *συμβάντα*. Συνεπώς, τα συμβάντα αυτά έχουν τη μορφή (V_f, c_f, w) και υποδεικνύουν ότι συνέβη η διαγραφή της τιμής w από το πεδίο τιμών D_{V_f} (η οποία προκλήθηκε από κάποια κλήση c_f.ArcCons ή c_f.LocalArcCons).

Ένα τέτοιο συμβάν μοιάζει με το any(V_f, w). Η διαφορά έγκειται στο ότι το τελευταίο προκαλεί $O(e)$ εισαγωγές στην ουρά συναρτήσεων διάδοσης της μηχανής διάδοσης περιορισμών όπως αναλύθηκε στην προηγούμενη παράγραφο, ενώ στον GAC-5 το ίδιο συμβάν προκαλεί μία μόνο εισαγωγή στην ουρά συμβάντων του αλγορίθμου.

Επειδή έχουμε (προς το παρόν) έναν τύπο συμβάντων στον GAC-5, δεν θα κατηγοριοποιήσουμε τα ίδια τα συμβάντα, αλλά τις συναρτήσεις διάδοσης που απαντώνται στον GAC-5 και συγκεκριμένα τις μεθόδους LocalArcCons.

1η Κατηγορία c.LocalArcCons: Χρήση παραμέτρων V_f και w. Κάθε c.LocalArcCons έχει σαν παραμέτρους εισόδου τις V_f και w. (Υπάρχει και η παράμετρος Q με την οποία όμως δεν θα ασχοληθούμε, γιατί ουσιαστικά χρησιμοποιείται μόνο σαν παράμετρος εξόδου.) Αλλά, όπως είδαμε και στα παραδείγματα της § 3.4, δεν εκμεταλλεύονται όλες οι μέθοδοι τις παραμέτρους τους. Συγκεκριμένα, το w το χρησιμοποιεί μόνο η μέθοδος c_1.LocalArcCons της § 3.4. Οπότε η μέθοδος αυτή εντάσσεται στην *1η κατηγορία*.

2η Κατηγορία c.LocalArcCons: **Χρήση παραμέτρου** V_f**.** Σε αυτήν την κατηγορία εντάσσονται οι μέθοδοι που κάνουν χρήση της παραμέτρου V_f, αλλά όχι και της παραμέτρου w. Π.χ. αυτό κάνει η μέθοδος c_3.LocalArcCons της § 3.4. Κατά αυτόν τον τρόπο επιβάλλεται συνέπεια ως προς την ακμή εκείνη του arcs(c) που ξεκινάει από τη V_f.

Ενώ η 1η κατηγορία c.LocalArcCons θα μπορούσαμε να πούμε ότι αφορά τα συμβάντα τύπου any(V_f, w), η 2η θα λέγαμε ότι αφορά γενικά τα συμβάντα any(V_f). Ωστόσο εμείς στην παρούσα εργασία θα την αντιστοιχήσουμε στο συμβάν bounds(V_f), πράγμα που σημαίνει ότι οι μέθοδοι αυτής της κατηγορίας θα καλούνται μόνο όταν η ελάχιστη ή η μέγιστη τιμή του πεδίου τιμών μίας μεταβλητής αλλάξει. Κάνουμε αυτή την αντιστοίχηση επειδή στα πραγματικά συστήματα επίλυσης οι περισσότερες συναρτήσεις διάδοσης αντιστοιχίζονται σε ένα τέτοιο συμβάν (bounds(V_f)) και επιβάλλουν συνέπεια *ορίων*.

3η Κατηγορία c.LocalArcCons**: Καμία χρήση των παραμέτρων** V_f **και** w**.** Σε αυτή την τελευταία κατηγορία εντάσσονται οι μέθοδοι εκείνες που επιβάλλουν συνέπεια ορίων, οι οποίες δεν εκμεταλλεύονται καμία από τις παραμέτρους εισόδου τους, όπως η c_2.LocalArcCons της § 3.4. Αυτό έχει σαν αποτέλεσμα να επιβάλλουν συνέπεια ορίων σε όλες τις ακμές (του συνόλου arcs(c)) του περιορισμού στον οποίο αναφέρονται. Αφού ο «κόμβος-αφετηρία» V_f ουσιαστικά δεν υπάρχει, η συνέπεια επιβάλλεται *προς όλες τις κατευθύνσεις*.

Όσον αφορά την αντιστοίχηση των συναρτήσεων διάδοσης με τα συμβάντα (όπως περιγράφηκαν στην προηγούμενη ενότητα), μπορούμε να πούμε ότι οι μέθοδοι c.LocalArcCons αυτής της κατηγορίας καλούνται όταν υπάρχει ένα ή περισσότερα συμβάντα bounds(V_f), με $V_f \in$ vars(c).

3.5.3 Θεωρητική Ομαδοποίηση των Συμβάντων

Η 1η κατηγορία c.LocalArcCons έχει ήδη ενταχθεί αρμονικά στον GAC-5. Αναφορικά με τη 2η κατηγορία, είναι προφανές ότι δεν χρησιμοποιεί τη μεταβλητή w του GAC-5. Δηλαδή τα στοιχεία (V_f, c_1, w_1), (V_f, c_2, w_2), …, (V_f, c_n, w_n) της ουράς Q του αλγορίθμου αντιστοιχούν σε ένα μόνο στοιχείο (V_f, c_n, w), όπου w τυχαίος αριθμός και όπου c_1, c_2, …, c_n οι περιορισμοί που (οι μέθοδοι ελέγχου συνέπειας τους) αφαίρεσαν τις αντίστοιχες τιμές w_1, w_2, …, w_n από το D_{V_f}, σε χρονολογική σειρά αφαίρεσης.

Απόδειξη. Την τελευταία πρόταση μπορούμε να την αποδείξουμε αν παρατηρήσουμε ποιες κλήσεις μεθόδων θα προκαλούσε το καθένα από τα n παραπάνω στοιχεία της ουράς του GAC-5.

Στοιχείο της Q	**Κλήσεις Συναρτήσεων $2^{\eta\varsigma}$ Κατηγορίας που προκαλεί**
(V_f, c_1, w_1)	c.LocalArcCons(V_f, w_1, Q), $\forall c \neq c_1$, με $V_f \in \text{vars}(c)$.
$\vdots$	$\vdots$
(V_f, c_n, w_n)	c.LocalArcCons(V_f, w_n, Q), $\forall c \neq c_n$, με $V_f \in \text{vars}(c)$.
Σύνολο Κλήσεων	c.LocalArcCons(V_f, w, Q), $\forall c$, με $V_f \in \text{vars}(c)$.

Από τον παραπάνω πίνακα συμπεραίνουμε λοιπόν ότι, λόγω της δυνατότητας που έχουμε να αλλάξουμε τη σειρά εκτέλεσης των συναρτήσεων διάδοσης και επειδή κάθε w_i μπορεί να αντικατασταθεί με το –τυχαίο– w, τα συμβάντα (V_f, c_1, w_1), ..., (V_f, c_n, w_n) που πιθανόν να υπάρχουν –ανάμεσα στα άλλα– στην ουρά του GAC-5 προκαλούν τις εξής κλήσεις: c.LocalArcCons με ορίσματα τα V_f, w, Q, για κάθε c.LocalArcCons $2^{\eta\varsigma}$ κατηγορίας, με τον περιορισμό c να συνδέεται με τη μεταβλητή V_f (ήτοι, $V_f \in \text{vars}(c)$).

Επειδή, όπως φαίνεται από τα στοιχεία της ουράς, η τελευταία κλήση συνάρτησης διάδοσης –πριν αρχίσουμε εμείς τις δικές μας κλήσεις– ήταν η c_n.LocalArcCons (από την οποία προέκυψε το συμβάν (V_f, c_n, w_n)), δεν υπάρχει λόγος να την ξανακαλέσουμε. Τελικά για τα n συμβάντα θα έχουμε τις εξής κλήσεις: c.LocalArcCons με ορίσματα τα V_f, w, Q, για κάθε c.LocalArcCons $2^{\eta\varsigma}$ κατηγορίας, με $c \neq c_n$ και $V_f \in \text{vars}(c)$. *Συνεπώς, τα συμβάντα* (V_f, c_1, w_1), ..., (V_f, c_n, w_n) *προκαλούν ισοδύναμες κλήσεις συναρτήσεων διάδοσης* $2^{\eta\varsigma}$ *κατηγορίας με αυτές που θα προκαλούσε το συμβάν* (V_f, c_n, w), *όπου* w *τυχαίος αριθμός.*

□

Επειδή οι c.LocalArcCons $2^{\eta\varsigma}$ κατηγορίας αφορούν την επιβολή συνέπειας *ορίων*, το πακετάρισμα των στοιχείων της ουράς (V_f, c_1, w_1), ..., (V_f, c_n, w_n) στο ισοδύναμο στοιχείο (V_f, c_n, w) έχει νόημα μόνο αν κάποιο από τα w_i ήταν πρώην ελάχιστο ή μέγιστο του D_{V_f}. Εφόσον δεν έχει πειραχτεί το ελάχιστο ή το μέγιστο του D_{V_f}, δεν υπάρχει λόγος να εφαρμόσουμε συνέπεια ορίων.

3.5.4 Υλοποίηση Ομαδοποίησης Συμβάντων

Έστω ότι έχουμε στην ουρά του GAC-5 τα συμβάντα (V_f, c_1, w_1), ..., (V_f, c_n, w_n). Εφόσον υπάρχουν μέθοδοι c.LocalArcCons $1^{\eta\varsigma}$ κατηγορίας με τις οποίες συνδέεται η μεταβλητή V_f (μέσω της σχέσης $V_f \in \text{vars}(c)$), δεν θα πρέπει να χάσουμε την πληροφορία που δίνουν οι τιμές $w_1, \ldots, w_n$.

Θα πρέπει όμως να λάβουμε υπόψη ότι αν υπάρχουν c.LocalArcCons $2^{\eta\varsigma}$ κατηγορίας με τις οποίες *επίσης* συνδέεται η μεταβλητή V_f (μέσω της σχέσης $V_f \in \text{vars}(c)$), τα συμβάντα (V_f, c_1, w_1), ..., (V_f, c_n, w_n) θα προκαλέσουν n περίπου άσκοπες κλήσεις της καθεμίας από αυτές. Επομένως μπορούμε να «πακετάρουμε» τα συμβάντα αυτά σε ένα της μορφής (V_f, W), όπου $W = \{(c_1, w_1), \ldots, (c_n, w_n)\}$. Μπορούμε επιπλέον να επαυξήσουμε αυτόν τον τύπο συμβάντος για να είμαστε σε θέση να γνωρίζουμε αν έχει αλλαχτεί είτε το ελάχιστο, είτε το μέγιστο του D_{V_f} και τον περιορισμό που έκανε την αλλαγή αυτή. Έτσι προκύπτει το συμβάν $(V_f, W, (c_b, boundc))$.

Η *boundc* (ονομάστηκε έτσι από τις λέξεις «**bound c**hanged») παίρνει την τιμή true ή false. Αν είναι true, υποδεικνύει ότι τα άκρα του D_{V_f} έχουν αλλάξει και ότι η τελευταία συνάρτηση διάδοσης που τα άλλαξε αφορούσε τον περιορισμό c_b.[10] Στον παρακάτω πίνακα φαίνεται το πώς μπορούμε να εκμεταλλευτούμε τον νέο τύπο συμβάντος $(V_f, W, (c_b, boundc))$ –τον οποίο θα ενσωματώσουμε αργότερα στη νέα ουρά του GAC-5– για να μειώσουμε τις κλήσεις των συναρτήσεων $2^{\eta\varsigma}$ κατηγορίας.

Κατηγορία c.LocalArcCons	**Κλήσεις της που προκαλεί ένα συμβάν** $(V_f, W, (c_b, boundc))$	**Αριθμός Κλήσεων**
1^{η}	$\forall w_i$ με $(c_i, w_i) \in W$ και $c_i \neq c$ c.LocalArcCons(V_f, w_i, Q).	$O(\|W\|)$
2^{η}	Αν $boundc =$ true και $c_b \neq c$ c.LocalArcCons(V_f, w, Q) (w τυχαίο).	$O(1)$

3.5.5 Γρήγορη Εισαγωγή στην Ουρά Νέου Τύπου

Ο νέος τύπος στοιχείων της ουράς του GAC-5 είναι ο $(V_f, W, (c_b, boundc))$, όπως προκύπτει από την προηγούμενη παράγραφο. Συνεπώς, ένα στιγμιότυπο της ουράς θα μπορούσε να είναι το εξής:

$$Q = \{(V_1, W_1, (c_{b_1}, boundc_1)), \ldots, (V_n, W_n, (c_{b_n}, boundc_n))\}.$$

[10] Δηλαδή η τελευταία συνάρτηση που αφαίρεσε το ελάχιστο ή το μέγιστο του D_{V_f} ήταν η c_b.ArcCons ή η c_b.LocalArcCons.

Αν η ουρά περιέχει αυτά τα στοιχεία-συμβάντα, ανακύπτει το παρακάτω ερώτημα: *Αν αφαιρεθεί η τιμή w_i από μία μεταβλητή V_i, με ποιον τρόπο θα ενημερωθεί η Q;* Η απάντηση είναι ότι αν υπάρχει το στοιχείο $(V_i, W_i, (c_{b_i}, boundc_i))$ εντός της ουράς, θα πρέπει να ενημερωθεί κατάλληλα. Αν δεν υπάρχει, τότε θα πρέπει να το δημιουργήσουμε.

Το «αναγνωριστικό» με το οποίο ξεχωρίζουμε ένα στοιχείο (V_i, W_i, (c_{b_i}, $boundc_i$)) της Q είναι η μεταβλητή V_i, αφού για κάθε μεταβλητή ορίζεται ένα το πολύ στοιχείο της ουράς. Επομένως, το επόμενο ερώτημα που προκύπτει είναι το εξής: *Πώς εντοπίζω ένα στοιχείο στην ουρά (αν υπάρχει);*

Αν κάναμε σειριακή αναζήτηση, θα κόστιζε $O(n)$ βήματα, τα οποία είναι πολλά για μία απλή πράξη εισαγωγής. Για να μειώσουμε το κόστος αυτό, εκμεταλλευόμαστε την ένα προς ένα αντιστοιχία κάθε μεταβλητής με ένα στοιχείο της ουράς. Έτσι ορίζουμε για κάθε μεταβλητή V_i έναν δείκτη V_i.q_item προς το τρέχον στοιχείο που την αντιπροσωπεύει στην ουρά. Αν δεν υπάρχει τέτοιο στοιχείο, τότε V_i.q_item $= NIL$.

Για να κατανοήσουμε τη χρησιμότητα αυτού του δείκτη, αναθεωρούμε τη μέθοδο αφαίρεσης τιμών από μία μεταβλητή V που περιγράφηκε στην § 3.4. Η μέθοδος αυτή αφαιρεί το διάστημα τιμών $[a\,..\,b]$ από το D_V, ενώ παράλληλα ενημερώνει την ουρά Q προσθέτοντας/ενημερώνοντας τα κατάλληλα συμβάντα.

```
1:  procedure V.REMOVE(a, b, Q, c_f)
2:      if V.q_item = NIL then
3:          Q ← Q ∪ {(V, ∅, (NIL, false))}
4:          V.q_item points to the new item of Q (end of Q)
5:      end if
6:      Get the item (V, W, (c_b, boundc)) that V.q_item points to
7:      if min(D_V) ∈ [a..b] or max(D_V) ∈ [a..b] then
8:          Replace (V, W, (c_b, boundc)) by (V, W, (c_f, true))
9:      end if
10:     if ∃ c with V ∈ vars(c) and c.LocalArcCons category is 1st then
11:         Replace (V, W, (c_b, boundc))
12:                 by (V, W ∪ {(c_f, w) | w ∈ [a..b] ∩ D_V}, (c_b, boundc)))
13:     end if
14:     D_V ← D_V − [a..b]          ▷ Αν D_V = ∅, το πρόβλημα δεν έχει λύση.
15: end procedure
```

Όταν η V δεν συνδέεται (μέσω της σχέσης $V \in \text{vars}(c)$) με κανέναν περιο-

ρισμό $1^{\eta\varsigma}$ κατηγορίας, η πολυπλοκότητα της παραπάνω μεθόδου είναι σταθερή. (Το αν μία μεταβλητή συνδέεται με έναν περιορισμό $1^{\eta\varsigma}$ κατηγορίας μπορούμε να το γνωρίζουμε εκ των προτέρων, οπότε ο έλεγχος που γίνεται στη γραμμή 10 της μεθόδου κοστίζει $O(1)$ βήματα και όχι $O(|\{c \mid V \in \text{vars}(c)\}|)$.) Αλλιώς, αν υπάρχει έστω και ένας τέτοιος περιορισμός $1^{\eta\varsigma}$ κατηγορίας, τότε το κόστος της μεθόδου είναι $O(|[a\,..\,b]|)$ –καθώς για κάθε αφαίρεση τιμής προκύπτει και από ένα συμβάν για το οποίο πρέπει να ενημερώσουμε την ουρά.

3.6 Ο Αλγόριθμος AC-5+

Έως τώρα παρουσιάστηκαν οι τροποποιήσεις που κάναμε γύρω από τον αλγόριθμο GAC-5: Στην § 3.5.4 αλλάξαμε τον τύπο της ουράς του, στην § 3.5.5 αναθεωρήσαμε τον τρόπο με τον οποίο η μέθοδος αφαίρεσης τιμής προσθέτει στοιχεία στη (νέα) ουρά και για όλα αυτά βασιστήκαμε στην § 3.5.2 όπου κάναμε έναν απλό διαχωρισμό των συναρτήσεων c.LocalArcCons σε τρεις κατηγορίες. Αφήσαμε για το τέλος τον κεντρικό αλγόριθμο επιβολής συνέπειας, τον οποίο ονομάσαμε AC-5+ και τον παρουσιάζουμε στην επόμενη σελίδα. (Θα μπορούσαμε να τον είχαμε ονομάσει και GAC-5+, αφού υποστηρίζει πλήρως τη γενικευμένη συνέπεια ακμών, αλλά δεν το κάναμε για να υποδείξουμε ότι αποτελεί μετεξέλιξη του αλγορίθμου AC-5.)

Ο AC-5+ δεν έχει την απλότητα του GAC-5, αλλά η υλοποίησή του δεν απαιτεί τη χρήση κάποιων πολύπλοκων δομών δεδομένων (η τάξη πολυπλοκότητας της μνήμης παρέμεινε σταθερή), ούτε και πρόσθεσε κάτι στην πολυπλοκότητα χρόνου. Αντιθέτως, φαίνεται καθαρά ότι προσπαθήσαμε να αποφύγουμε τον εσωτερικό βρόχο των γραμμών 12–14, επισημαίνοντας δύο νέες κατηγορίες c.LocalArcCons –τη 2^{η} και την 3^{η}– τις οποίες μπορούμε να επεξεργαστούμε με ταχύτερο τρόπο. Οι c.LocalArcCons των δύο τελευταίων κατηγοριών είναι αυτές που απαντώνται κατά συντριπτική πλειοψηφία στα σύγχρονα συστήματα Προγραμματισμού με Περιορισμούς, καθώς τα περισσότερα από αυτά εστιάζουν στη *συνέπεια ορίων* και στους *καθολικούς περιορισμούς*.

3.6.1 Ενσωμάτωση c.LocalArcCons $3^{\eta\varsigma}$ Κατηγορίας

Αφήσαμε για το τέλος την επεξήγηση της ενσωμάτωσης των συναρτήσεων c.LocalArcCons(V, w, Q) $3^{\eta\varsigma}$ κατηγορίας στον AC-5+. Υπενθυμίζουμε ότι η κατηγορία αυτή αποτελεί υποπερίπτωση της $2^{\eta\varsigma}$ κατηγορίας. Οι συναρτήσεις

```
 1: procedure AC-5+
 2:     time ← 0
 3:     Q ← ∅
 4:     for each c ∈ 𝒞 do
 5:         c.ArcCons(Q)
 6:     end for
 7:     while Q ≠ ∅ do
 8:         Pick and remove a (V, W, (c_b, boundc, timec)) out of Q
 9:         V.q_item ← NIL
10:         for each c ∈ 𝒞, with V ∈ vars(c) do
11:             if c.LocalArcCons category is 1st then
12:                 for each (c_f, w) ∈ W, with c ≠ c_f do
13:                     c.LocalArcCons(V, w, Q)
14:                 end for
15:             else if c.LocalArcCons category is 2nd then
16:                 if boundc = true and c ≠ c_b then
17:                     c.LocalArcCons(V, w, Q)              ▷ w τυχαίο.
18:                 end if
19:             else                          ▷ c.LocalArcCons category is 3rd
20:                 if boundc = true and c.last_check_time < timec then
21:                     c.LocalArcCons(V, w, Q)
22:                                  ▷ Τα V και w δεν χρησιμοποιούνται.
23:                 end if
24:             end if
25:             c.last_check_time ← time
26:             time ← time + 1
27:         end for
28:     end while
29: end procedure
```

διάδοσης που υπάγονται σε αυτήν δεν εκμεταλλεύονται ούτε το όρισμα V ούτε το w. Δηλαδή, για την 3^{η} κατηγορία ισχύει καταρχήν ό,τι και για την 2^{η} –λόγω της μη εκμετάλλευσης του w.

Όσον αφορά τη μη εκμετάλλευση του V, αυτή συνεπάγεται ότι δύο συμβάντα $(V_1, W_1, (c_{b_1}, boundc_1, t_1))$ και $(V_2, W_2, (c_{b_2}, boundc_2, t_2))$ που βρίσκονται ταυτόχρονα στην ουρά Q θα έχουν το ίδιο αποτέλεσμα όσον αφορά τη c.LocalArcCons (εφόσον $V_1, V_2 \in \text{vars}(c)$): θα κληθεί (άσκοπα) δύο φορές –αν αντιμετωπιστεί σαν συνάρτηση $2^{\eta\varsigma}$ κατηγορίας. Εξάλλου, εφόσον αυτή η c.LocalArcCons δεν διαβάζει τις παραμέτρους εισόδου της, οι κλήσεις c.LocalArcCons(V_1, w, Q) και c.LocalArcCons(V_2, w, Q) ταυτίζονται.

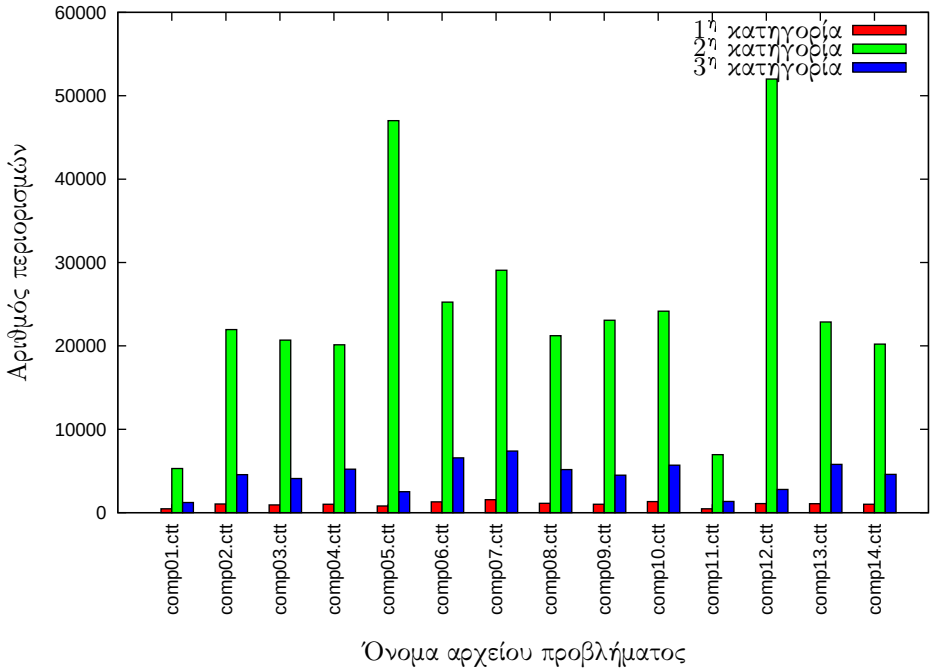
Γι′ αυτόν τον λόγο έχουμε ήδη προσθέσει στον τύπο στοιχείου-συμβάντος της ουράς τη μεταβλητή t (ή $timec$) στην οποία σημειώνεται ο χρόνος που συνέβη η τελευταία μεταβολή των ορίων του D_V. Μέσω αυτής ο AC-5+ αποφεύγει άσκοπες κλήσεις των c.LocalArcCons $3^{\eta\varsigma}$ κατηγορίας· τις καλεί μία φορά για όλες τις μεταβολές των ορίων του D_V, με $V \in \text{vars}(c)$, που έχουν σημειωθεί στην Q.

3.7 Πειραματικά Αποτελέσματα

Ο AC-5+ έχει υλοποιηθεί σε μία νέα έκδοση του επιλυτή προβλημάτων ικανοποίησης περιορισμών Naxos [42]. Στον επιλυτή αυτόν είναι υλοποιημένη μία ποικιλία περιορισμών. Έχουμε δηλαδή περιορισμούς όλων των κατηγοριών.

Το ζητούμενο λοιπόν ήταν να βρούμε και να υλοποιήσουμε ένα ΠΙΠ που θα εκμεταλλευόταν περιορισμούς και των τριών κατηγοριών με τις οποίες έχουμε ασχοληθεί σε αυτό το κεφάλαιο. Τα απλά και γνωστότερα ΠΙΠ εστιάζουν συνήθως σε μία μικρή γκάμα περιορισμών. Για αυτόν τον λόγο ασχοληθήκαμε με ένα μεγάλο «πολυσυλλεκτικό» ΠΙΠ, το οποίο περιέχει περιορισμούς πολλών ειδών. Πρόκειται για το πρόβλημα κατάρτισης ωρολογίων προγραμμάτων για εκπαιδευτικά ιδρύματα με τη μορφή που είχε στον δεύτερο αντίστοιχο διεθνή διαγωνισμό (International Timetabling Competition 2007). Εκτενής περιγραφή του προβλήματος υπάρχει στο Παράρτημα Α′.

Στο Σχήμα 3.1 φαίνεται ο αριθμός των περιορισμών των 14 στιγμιότυπων του προβλήματος τα οποία αποτέλεσαν το επίκεντρο του διαγωνισμού. Είναι εμφανής η κατηγοριοποίησή τους στις τρεις κατηγορίες της § 3.5.2. Ωστόσο, οφείλουμε να επισημάνουμε ότι δεν μπορούμε να συμπεράνουμε από το διάγραμμα την ουσιαστική επίδραση της κάθε κατηγορίας στο πρόβλημα, εφόσον στην καταμέτρηση δεν ξεχωρίσαμε π.χ. τους δυαδικούς από τους καθολικούς

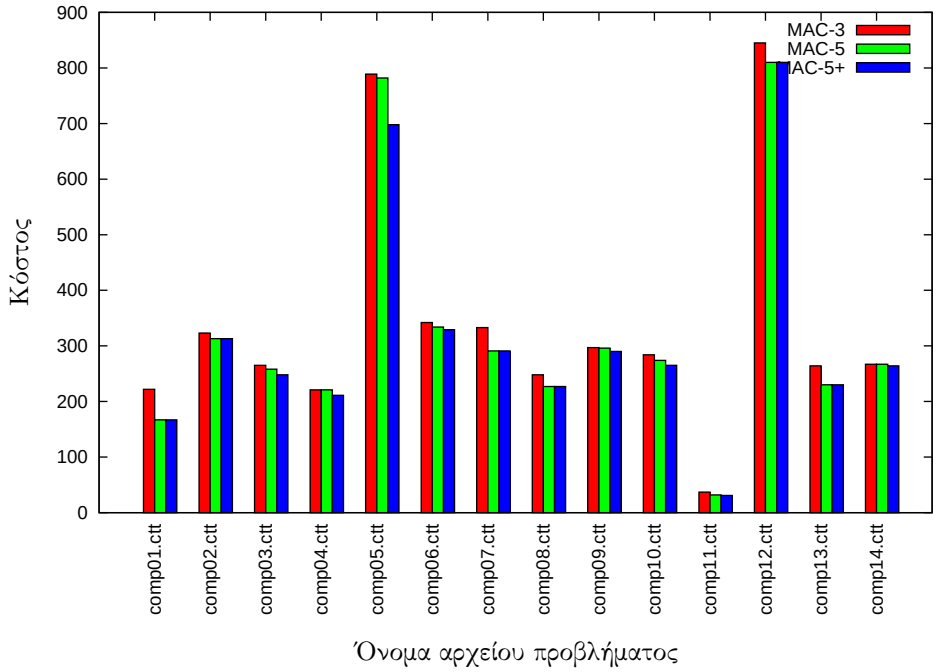


Σχήμα 3.1: Οι κατηγορίες των περιορισμών που απαντώνται στα 14 προβλήματα κατάρτισης ωρολογίου προγράμματος

περιορισμούς που διατίθενται στον επιλυτή Naxos. Είναι εμφανές πάντως ότι κυριαρχούν –όπως συνηθίζεται στα πρακτικά συστήματα– περιορισμοί στους οποίους επιβάλλεται *συνέπεια ορίων* (δηλαδή περιορισμοί της $2^{\eta\varsigma}$ και της $3^{\eta\varsigma}$ κατηγορίας).

Επιχειρήσαμε λοιπόν να συγκρίνουμε τον AC-5+ με τον «προκάτοχό» του, τον AC-5, ο οποίος αποτελεί εξέλιξη του AC-3. Στα σχήματα που ακολουθούν υπάρχουν γραφικές παραστάσεις των αποτελεσμάτων που πήραμε προσπαθώντας να επιλύσουμε τα 14 στιγμιότυπα με τη βοήθεια των αλγορίθμων αυτών. Λέμε ότι η επίλυση έγινε *«με τη βοήθεια»* των αλγορίθμων επιβολής συνέπειας ακμών/ορίων, γιατί από μόνοι τους αυτοί δεν αρκούν πάντα για να επιλύσουν ένα ΠΙΠ (βλ. κ. § 2.3.4). Ως εκ τούτου, χρειάστηκε να τους συνδυάσουμε με μία μέθοδο αναζήτησης.

Χρησιμοποιήσαμε τη μέθοδο *αναζήτησης με φραγμένη κατά βάθος ασυμφωνία* (depth-bounded discrepancy search – DDS) [38]. Πρόκειται για μία μέθοδο αναζήτησης με οπισθοδρόμηση (βλ. κ. § 2.3.3). Σε κάθε βήμα των μεθόδων αυτών (δηλαδή σε κάθε ανάθεση τιμής σε μία μεταβλητή του προβλήματος) ε-

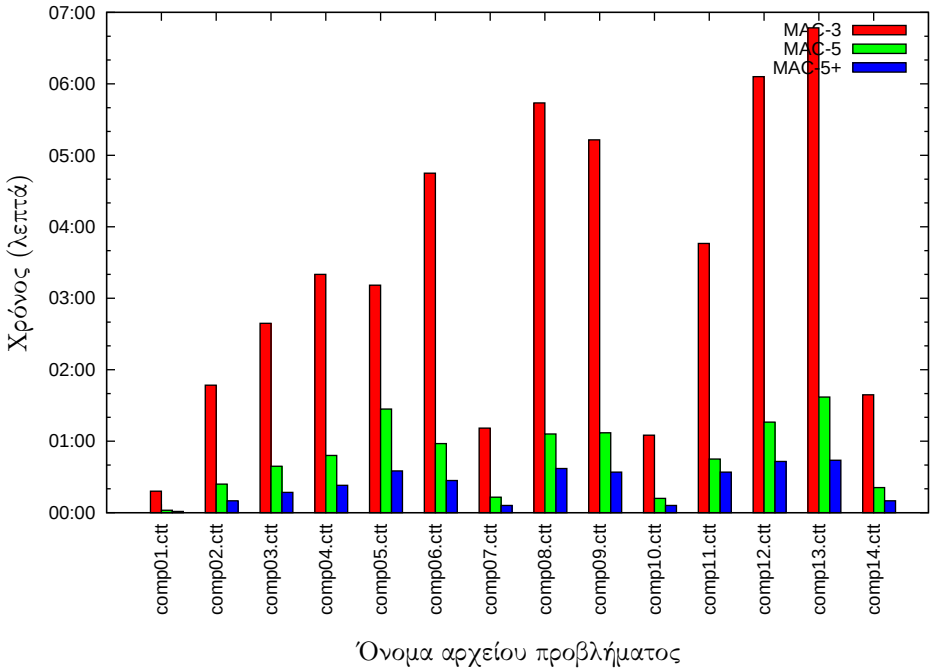


Σχήμα 3.2: Τα κόστη των λύσεων που βρέθηκαν από τις τρεις διαφορετικές μεθοδολογίες που εφαρμόστηκαν (σε καθένα από τα 14 προβλήματα κατάρτισης ωρολογίου προγράμματος)

πιβάλλουμε συνέπεια ακμών –έτσι ώστε να μειώσουμε τον χώρο αναζήτησης. Η μεθοδολογία αυτή ονομάζεται *διατήρηση συνέπειας ακμών* (maintaining arc consistency – MAC) [34] και χρησιμοποιείται κατά κόρον στους επιλυτές, όπως και στον Naxos. Αυτός είναι ο λόγος για τον οποίον στα παρακάτω σχήματα χρησιμοποιούνται οι ονομασίες MAC-3, MAC-5 και MAC-5+.

Για κάθε δυνατό συνδυασμό μεθοδολογίας και στιγμιότυπου προβλήματος αφήσαμε τον Naxos να το επιλύει βρίσκοντας διαδοχικά λύσεις, με τη μία λύση να είναι υποχρεωτικά καλύτερη από την προηγούμενή της, σύμφωνα με τη μεθοδολογία *διακλάδωσε-και-φράξε* (branch-and-bound) [22]. Ο χρόνος που έτρεχε ο Naxos για κάθε ΠΠΠ ήταν προκαθορισμένος από τον διαγωνισμό μέσω ενός προγράμματος (benchmark) που ανέλυε τις δυνατότητες κάθε υπολογιστή. (Π.χ. σε ένα μηχάνημα με διπύρηνο επεξεργαστή Intel Core στα 2.8 GHz, το χρονικό όριο που έπρεπε να τεθεί ήταν 7 λεπτά και 20 δευτερόλεπτα.)

Στο Σχήμα 3.2 παρατηρούμε ότι μέσα στα πλαίσια αυτά η MAC-5+ προλαβαίνει να δώσει ελαφρώς καλύτερες λύσεις από τη MAC-5, η οποία επίσης δίνει

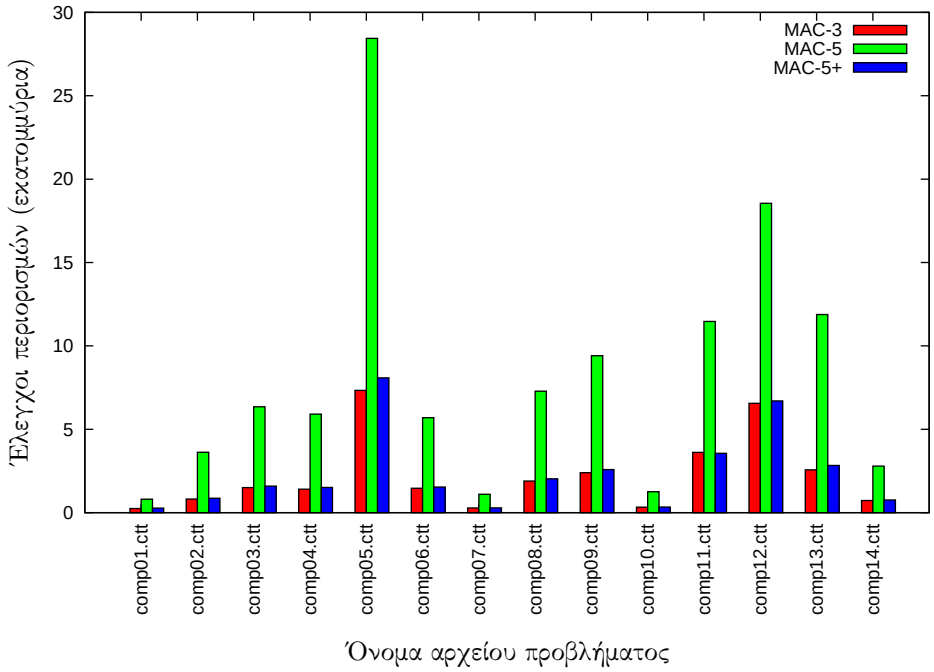


Σχήμα 3.3: Ο χρόνος που χρειάστηκαν οι τρεις διαφορετικές μεθοδολογίες για να φτάσουν στην καλύτερη *κοινή* τους λύση (σε καθένα από τα 14 προβλήματα κατάρτισης ωρολογίου προγράμματος)

ελαφρώς καλύτερες λύσεις από τη MAC-3.

Πριν φτάσουν όμως στις βέλτιστες λύσεις, οι μεθοδολογίες αυτές έχουν βρει κάποιες κοινές λύσεις, σαν ενδιάμεσες. (Δεν αποκλείεται βέβαια η βέλτιστη λύση της μίας μεθοδολογίας να είναι ενδιάμεση κάποιας –καλύτερης– μεθοδολογίας.) Για να κατανοήσουμε καλύτερα τη φιλοσοφία που κρύβεται πίσω από τους τρεις αλγορίθμους, στα τελευταία δύο σχήματα εστιάσαμε στη *βέλτιστη κοινή λύση* των τριών μεθοδολογιών που προέκυψε για κάθε στιγμιότυπο και το *πώς* φτάσαμε σε αυτήν.

Το *πότε* φτάσαμε στη βέλτιστη κοινή λύση φαίνεται στο Σχήμα 3.3· η υπεροχή της MAC-5+ είναι προφανής. Η φιλοσοφία της όμως αναδεικνύεται από το Σχήμα 3.4. Από το διάγραμμα αυτό φαίνεται ότι χρήση του AC-5 αυξάνει κατακόρυφα τους ελέγχους περιορισμών. Προηγουμένως όμως είδαμε ότι είναι αποδοτικότερος του AC-3. Το οξύμωρο αυτό εξηγείται από το γεγονός ότι ένας έλεγχος περιορισμών του AC-5 παίρνει λιγότερο χρόνο από τους αντίστοιχους του AC-3. Για να γίνει όμως αυτό, ο AC-5 αυξάνει τον αριθμό των –ταχύτερων



Σχήμα 3.4: Ο αριθμός των ελέγχων περιορισμών που έκαναν οι τρεις διαφορετικές μεθοδολογίες, μέχρι να φτάσουν στην καλύτερη *κοινή* τους λύση (σε καθένα από τα 14 προβλήματα κατάρτισης ωρολογίου προγράμματος)

κατά τα άλλα– ελέγχων περιορισμών. Ο AC-5+ μειώνει τις σπατάλες του AC-5 εκμεταλλευόμενος καλύτερα τα συμβάντα που προκύπτουν κατά την εκτέλεσή του.

Κεφάλαιο 4

Απεικόνιση Πεδίου Τιμών ως Σύνολο Διαστημάτων

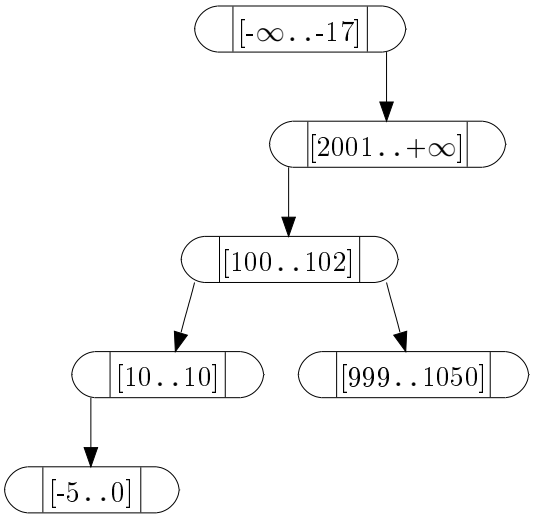
Στη βιβλιογραφία δεν γίνεται συνήθως λόγος για τη δομή δεδομένων στην οποία αποθηκεύουμε ένα πεδίο τιμών αποτελούμενο από ακέραιες τιμές. Είθισται να χρησιμοποιείται ένας πίνακας από λογικές μεταβλητές (οι οποίες παριστάνονται από bit 0 ή 1).

Ας πάρουμε την απλή περίπτωση στην οποία το πεδίο τιμών περιλαμβάνει μόνο *θετικές* ακέραιες τιμές και έστω ότι ο πίνακας για ένα πεδίο τιμών D ονομάζεται a. Τότε έχουμε ότι μία τιμή v ανήκει στο D, αν και μόνο αν $a[v] = 1$. Άλλες υλοποιήσεις εκμεταλλεύονται υπάρχουσες δομές δεδομένων που περιγράφουν σύνολα ακεραίων.

4.1 Απεικόνιση Συνόλου Διαστημάτων

Η καινοτομία της δικής μας προσέγγισης έχει να κάνει με τον *τύπο των στοιχείων* που περιέχει το σύνολο το οποίο υλοποιήσαμε. Δεν φτιάξαμε ένα σύνολο από αριθμούς, αλλά ένα σύνολο από *διαστήματα* ακεραίων αριθμών.

Για την ακρίβεια, θεωρήσαμε σκόπιμο τα στοιχεία του συνόλου να είναι τα *κενά διαστήματα* του πεδίου τιμών που περιγράφουμε. Η λογική πίσω από αυτή την επιλογή μας έχει να κάνει με το γεγονός ότι κατά την πορεία επίλυσης ενός προβλήματος ικανοποίησης περιορισμών το μέγεθος των πεδίων τιμών των μεταβλητών ελαττώνεται, καθώς γίνονται αναθέσεις τιμών και διάδοση περιορισμών. Δηλαδή δεν γίνεται να έχουμε προσθήκες τιμών σε ένα πεδίο τιμών, παρά μόνο αφαιρέσεις. Κατά την αφαίρεση λοιπόν μίας τιμής ή ενός διαστή-



Σχήμα 4.1: Δένδρο με τα κενά διαστήματα του πεδίου τιμών [-16..-6 1..9 11..99 103..998 1051..2000]

ματος τιμών από ένα πεδίο, εμείς εισάγουμε στο σύνολο κενών διαστημάτων ένα ακόμα στοιχείο του. Το σύνολο με τα κενά υλοποιήθηκε σαν ένα δυαδικό δένδρο αναζήτησης.

Για παράδειγμα, το πεδίο τιμών [9..17 44..101] περιγράφεται από τα τρία κενά $[-\infty..8]$, [18..43] και $[102..+\infty]$. Στο Σχήμα 4.1 έχουμε ένα άλλο παράδειγμα απεικόνισης (των κενών) ενός πεδίου τιμών. Τα κενά είναι μάλιστα διατεταγμένα σύμφωνα με τη δομή ενός δυαδικού δένδρου. Προφανώς, ένας κόμβος του δένδρου περιέχει τον πρώτο και τον τελευταίο αριθμό ενός κενού διαστήματος, καθώς και δείκτες στο αριστερό και στο δεξί παιδί του κόμβου.

4.2 Αλγόριθμος Διαγραφής/Αναζήτησης

Δύο είναι οι βασικές μέθοδοι οι οποίες αλληλεπιδρούν με ένα πεδίο τιμών: η μέθοδος διαγραφής ενός διαστήματος τιμών και η μέθοδος (αναζήτησης) που ελέγχει αν ένα διάστημα τιμών ανήκει στο πεδίο τιμών.

Το πλεονέκτημα της προσέγγισής μας είναι ότι οι δύο βασικές μέθοδοι διαγραφής/αναζήτησης υλοποιούνται ουσιαστικά από την ίδια συνάρτηση ονόματι SearchGap, που παρουσιάζουμε παρακάτω. Η SearchGap δέχεται τέσσερα ορίσματα $(gapNode, newStartVal, newEndVal, removeInterval)$. Αν το

removeInterval είναι 1, τότε ο αλγόριθμος *διαγράφει* το διάστημα [*newStartVal*..*newEndVal*] από το πεδίο τιμών, που παριστάνεται από ένα δένδρο με ρίζα τον *gapNode*.

Αν το *removeInterval* είναι 0, τότε η συνάρτηση επιστρέφει έναν κόμβο του δένδρου ο οποίος περιέχει έστω και ένα στοιχείο του [*newStartVal*.. *newEndVal*]. Αν δεν υπάρχει κάποιος κόμβος που να πληροί αυτό το κριτήριο, τότε επιστρέφεται ένας κενός κόμβος. Έτσι, για να ελέγξουμε π.χ. αν ένα διάστημα [*a*..*b*] ανήκει στο πεδίο τιμών D (που απεικονίζεται από τη δομή δεδομένων), καλούμε τη SearchGap(ρίζα_δένδρου, $a, b, 0$) και εάν ο επιστρεφόμενος κόμβος είναι κενός, τότε ισχύει $[a..b] \subseteq D$, αλλιώς ισχύει $[a..b] \nsubseteq D$.

Ακολουθεί ο εν λόγω αλγόριθμος.

```
 1: function SearchGap (gapNode, newStartVal, newEndVal,
 2:                      removeInterval)
 3: while true do
 4:     if gapNode is empty then
 5:         if removeInterval = 1 then
 6:             Insert [newStartVal..newEndVal] into gapNode
 7:         end if
 8:         return gapNode
 9:     else
10:         gapStartVal ← min_of_gap_in(gapNode)
11:         gapEndVal ← max_of_gap_in(gapNode)
12:         if newEndVal + removeInterval < gapStartVal then
13:             gapNode ← left_child(gapNode)
14:         else if newStartVal − removeInterval > gapEndVal then
15:             gapNode ← right_child(gapNode)
16:         else
17:             if removeInterval = 1 and (newStartVal < gapStartVal
18:                                        or newEndVal > gapEndVal) then
19:                 if newStartVal ≥ gapStartVal then
20:                     newStartVal ← gapStartVal
21:                 else
22:                     extensionNode ← SearchGap(left_child(gapNode),
23:                                               newStartVal − 1, newStartVal − 1, 0)
24:                     if extensionNode not empty then
25:                         newStartVal ← min_of_gap_in(extensionNode)
26:                     end if
```

```
27:                 end if
28:                 if newEndVal ≤ gapEndVal then
29:                     newEndVal ← gapEndVal
30:                 else
31:                     extensionNode ← SearchGap(right_child(gapNode),
32:                                               newEndVal + 1, newEndVal + 1, 0)
33:                     if extensionNode not empty then
34:                         newEndVal ← max_of_gap_in(extensionNode)
35:                     end if
36:                 end if
37:                 Insert [newStartVal..newEndVal] into gapNode
38:             end if
39:             return gapNode
40:         end if
41:     end if
42: end while
43: end function
```

Ο αλγόριθμος «περικλείεται» από έναν ατέρμονα βρόχο, σε κάθε επανάληψη του οποίου διασχίζεται ο κόμβος του δένδρου $gapNode$ (ο οποίος αρχικά είναι η ρίζα του δένδρου, ενώ στο τέλος κάθε επανάληψης έχει αντικατασταθεί από κάποιο παιδί του).

Προχωρώντας στη γραμμή 12 (και αντίστοιχα στη γραμμή 14) της συνάρτησης, παρατηρούμε τη χρήση της μεταβλητής $removeInterval$ μέσα σε συνθήκη. Η χρησιμότητά της εδώ έχει να κάνει με την περίπτωση εισαγωγής (ή, καλύτερα, προσκόλλησης) ενός κενού διαστήματος $[newStartVal..newEndVal] = [a..b]$, όταν ο τρέχων κόμβος περιλαμβάνει το διάστημα $[gapStartVal..gapEndVal] = [b+1..c]$ (και συνεπώς μπορεί να επεκταθεί σε $[gapStartVal'..gapEndVal'] = [a..c]$). Το «+1» εδώ αναπαρίσταται από τη $removeInterval$.

Η συνθήκη στις γραμμές 17–18 υπάρχει για να ελεγχθεί αν $[newStartVal..newEndVal] \subseteq [gapStartVal..gapEndVal]$. Αν ισχύει αυτό, τότε δεν έχει νόημα να εισάγουμε κάποιο νέο κενό διάστημα.

Στις γραμμές 22–23 (και αντίστοιχα στις γραμμές 31–32) αναζητούμε αν υπάρχει κάποιο κενό διάστημα $[a..b]$ στο δένδρο, με $newStartVal-1 \in [a..b]$. Αν υπάρχει, τότε το νέο κενό διάστημα $[newStartVal..newEndVal]$ μπορεί να συγχωνευθεί με το $[a..b]$.

Σημειώνεται τέλος ότι στην πραγματικότητα κάθε κόμβος του δένδρου δεν περιέχει μόνο ένα κενό, αλλά μία στοίβα από κενά (το καθένα από αυτά συνο-

δευόμενο από μία «χρονική ένδειξη» για το πότε προστέθηκε) η οποία γεμίζει καθώς ένα πρόβλημα ικανοποίησης περιορισμών επιλύεται και αδειάζει όποτε οπισθοδρομούμε.

4.3 Εφαρμογή σε Πρόβλημα Ακολουθίας DNA

Η χρήση της δομής δεδομένων που εξετάζουμε ενδείκνυται όταν έχουμε μεγάλα μη συνεχή πεδία τιμών. Σε αυτή την ενότητα παρουσιάζουμε ένα απλό πρόβλημα με μεταβλητές με μεγάλα μη συνεχή πεδία τιμών και συγκρίνουμε τη δική μας προσέγγιση με αυτή άλλων γνωστών επιλυτών.

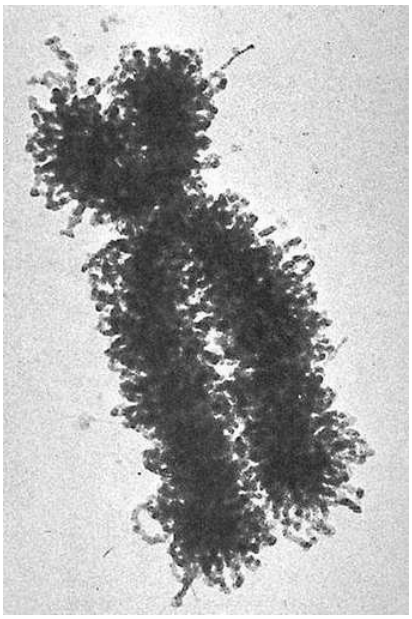
4.3.1 Ορισμός Χρωμοσώματος [24]

Κάθε κύτταρο ενός ανθρώπου περιέχει 46 χρωμοσώματα, κάποια από τα οποία μοιάζουν με αυτό του Σχήματος 4.2. Ένα χρωμόσωμα αποτελεί μέρος του γενετικού μας υλικού, αφού δεν είναι τίποτα άλλο από μία ακολουθία νουκλεοτιδίων DNA. Υπάρχουν τέσσερις τύποι νουκλεοτιδίων που συμβολίζονται με A, T, G και C. (A = αδενίνη, T = θυμίνη, G = γουανίνη, C = κυτοσίνη.) Σε κάθε χρωμόσωμά μας υπάρχει μία ακολουθία 247.2 περίπου εκατομμυρίων τέτοιων νουκλεοτιδίων. Π.χ.

$$\underbrace{\texttt{C A G G C A T ... G A A A C A A}}_{\text{247,200,000 νουκλεοτίδια}}$$

4.3.2 Ένα Απλό Πρόβλημα Ακολουθιών

Έστω ότι θέλουμε να «χωρέσουμε» σε ένα χρωμόσωμα μία ακολουθία από τέσσερις κυτοσίνες C_1, C_2, C_3, C_4 και μία ακολουθία από τέσσερις γουανίνες G_1, G_2, G_3, G_4. Έστω ότι η πρώτη ακολουθία μοιάζει με γεωμετρική πρόοδο, με $C_i = \lfloor C_{i+1}/99 \rfloor$ και η δεύτερη ακολουθία είναι αριθμητική πρόοδος με $G_{i+1} = G_i + 99$. Με C_i και G_i συμβολίσαμε τις θέσεις των αντίστοιχων νουκλεοτιδίων στην αλυσίδα DNA· το πεδίο τιμών για μία θέση είναι το $[1..247200000]$. Το ζητούμενο είναι να βρούμε καθεμία από τις θέσεις αυτές. Ο πηγαίος κώδικας που το επιλύει σε διάφορα συστήματα προγραμματισμού με περιορισμούς είναι διαθέσιμος στο Παράρτημα Β′.



Σχήμα 4.2: Ένα χρωμόσωμα ανθρώπου ιδωμένο μέσα από ηλεκτρονικό μικροσκόπιο [36]

4.3.3 Δυσκολίες κατά την Επίλυση

Ένα απλό και «αθώο» πρόβλημα, λοιπόν, οκτώ περιορισμένων μεταβλητών το οποίο γίνεται... δύσκολο αν δεν διαχειριστούμε σωστά τα οκτώ πεδία τιμών τα οποία περιέχουν εκατομμύρια τιμές.

Για την επίλυση του προβλήματος χρησιμοποιήσαμε αρχικά τη δομή δεδομένων για τα πεδία τιμών, που περιγράφηκε σε αυτό το κεφάλαιο. Ο επιλυτής μέσα στον οποίο ενσωματώσαμε τη δομή σαν επέκταση είναι ο Naxos [42]. Πήραμε ακαριαία τη λύση[1] στο πρόβλημα χρησιμοποιώντας 3 MB μνήμη.

Εντούτοις, στο ίδιο μηχάνημα[2] το σύστημα λογικού προγραμματισμού με περιορισμούς ECL^{i}PS^e [1, 2] έκανε τρία δευτερόλεπτα για να επιλύσει το πρόβλημα, αλλά το χειρότερο είναι ότι χρησιμοποίησε 124 MB μνήμη. Βρεθήκαμε δηλαδή λίγο πριν από το προκαθορισμένο όριο χρήσης μνήμης από την ECL^{i}PS^e! Αν δοκιμάσουμε να προσθέσουμε ένα ακόμη νουκλεοτίδιο, η επίλυση θα στα-

[1] Οι θέσεις για τα νουκλεοτίδια όπως προκύπτουν από την πρώτη λύση που βρέθηκε είναι οι $C_1 = 1$, $C_2 = 99$, $C_3 = 9801$, $C_4 = 970299$, $G_1 = 2$, $G_2 = 101$, $G_3 = 200$ και $G_4 = 299$.

[2] Ο υπολογιστής στον οποίο κάναμε τα πειράματα είναι ένας Sun Blade με επεξεργαστή SPARC ταχύτητας 1.5 GHz και μνήμη μεγέθους 8 GB.

ματήσει λόγω έλλειψης μνήμης.

Στο ίδιο μηχάνημα πάλι, ο επιλυτής ILOG [21, 20] κάνει τριπλάσιο χρόνο σε σχέση με την ECL^{i}PS^e για να βρει λύση (περίπου δέκα δευτερόλεπτα), ενώ δαπανά σχεδόν την ίδια μνήμη. Όσον αφορά την κλιμάκωση του προβλήματος, ο ILOG μάς επιτρέπει να εκμεταλλευτούμε όλη σχεδόν τη μνήμη του υπολογιστή και συνεπώς είναι δυνατόν να προσθέσουμε ακόμα μέχρι 80 περίπου νουκλεοτίδια στην αλυσίδα DNA. Μπορούμε π.χ. να βάλουμε άλλες 80 γουανίνες στην αντίστοιχη αριθμητική πρόοδο, θα σπαταλήσουμε όμως γύρω στα 5 GB μνήμη (καθώς και μία ώρα από τον χρόνο μας μέχρι να εμφανιστούν τα αποτελέσματα)!

Από την άλλη πλευρά, ο NAXOS κλιμακώνεται ομαλά και μπορεί να συμπεριλάβει χιλιάδες περιορισμένες μεταβλητές-νουκλεοτίδια με μεγάλα πεδία τιμών. Π.χ. μπορεί σε τρία λεπτά και χρησιμοποιώντας μόλις 6 MB μνήμη να επιλύσει το ίδιο πρόβλημα, με την αριθμητική πρόοδο να περιέχει τώρα χίλιες γουανίνες.

4.3.4 Συμπεράσματα

Γνωστοί επιλυτές προβλημάτων ικανοποίησης περιορισμών χρησιμοποιούν δομές δεδομένων, οι οποίες για να αποθηκεύσουν ένα πεδίο τιμών χρησιμοποιούν μνήμη ανάλογη με το μέγεθός του. (Με εξαίρεση την περίπτωση που το πεδίο τιμών είναι συνεχές: τότε αυτό μπορεί να αναπαρασταθεί μόνο από τα δύο άκρα του και όχι από ολόκληρο πίνακα τιμών.)

Η δική μας υλοποίηση χρησιμοποιεί μνήμη μεγέθους ανάλογου με τον αριθμό των κενών διαστημάτων που το πεδίο τιμών περιέχει. Ωστόσο, λόγω της δενδρικής μορφής της δομής, οι βασικές λειτουργίες που αφορούν στην αναζήτηση ή διαγραφή μίας τιμής από το πεδίο, κοστίζουν λογαριθμικό χρόνο (ως προς τον αριθμό των κενών που το πεδίο τιμών περιέχει). Οι ίδιες λειτουργίες σε έναν πίνακα που έχει ήδη δημιουργηθεί –με το όποιο τίμημα σε μνήμη και χρόνο– θα έπαιρναν σταθερό χρόνο. Οι λειτουργίες όμως που αφορούν αναζήτηση ή διαγραφή ενός συνεχούς *διαστήματος* ακεραίων τιμών, μεγέθους έστω d, ενώ συνεχίζουν να κοστίζουν τον ίδιο λογαριθμικό χρόνο όσον αφορά στη δενδρική δομή, σε έναν πίνακα κοστίζουν $O(d)$ επαναλήψεις.

Κεφάλαιο 5

Δίκαιη Διχοτόμηση Πεδίων Τιμών

Ας θεωρήσουμε ότι έχουμε ένα δίκτυο περιορισμών σε κατάσταση συνέπειας ακμών. Ως γνωστόν, η συνέπεια ακμών δεν συνεπάγεται ότι οι τιμές των πεδίων τιμών του προβλήματος ικανοποίησης περιορισμών συμμετέχουν απαραίτητα σε μία λύση του προβλήματος αυτού (βλ. κ. Σχήμα 2.1). Έτσι, οδηγούμαστε εκ των πραγμάτων στη χρήση μίας μεθόδου αναζήτησης –σε συνδυασμό συνήθως με έναν αλγόριθμο επιβολής συνέπειας ακμών.

Π.χ. σύμφωνα με την πρακτική της *διατήρησης συνέπειας ακμών* (maintaining arc consistency – MAC) [34], επιλέγουμε μία μη δεσμευμένη περιορισμένη μεταβλητή, της αναθέτουμε μία τιμή από το πεδίο τιμών της και επιβάλλουμε ξανά συνέπεια ακμών. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να βρούμε μία λύση. Αν κάποια στιγμή φτάσουμε σε αδιέξοδο, ήτοι σε ένα ασυνεπές δίκτυο περιορισμών, οπισθοδρομούμε σε μία προηγούμενη κατάσταση και στη συνέχεια επιλέγουμε άλλες ατραπούς προς τη λύση.

Η *διχοτόμηση πεδίων τιμών* (domain splitting) είναι μία παραλλαγή της παραπάνω μεθοδολογίας. Εν ολίγοις, στη διχοτόμηση πεδίων τιμών αντί για μία *ανάθεση* τιμής σε μία μεταβλητή –κατά τη διαδικασία της αναζήτησης–, έχουμε περιορισμό των πεδίων τιμών της: αποκόπτονται οι μισές τιμές της, εξ ου και ο όρος «διχοτόμηση». Με άλλα λόγια, αντί να «σβήνουμε» όλες τις τιμές πλην μίας (από το πεδίο μίας συγκεκριμένης μεταβλητής κατά την ανάθεση), κόβουμε απλά τις μισές τιμές. Έτσι δεν προκύπτει ανάθεση, εκτός αν το πεδίο τιμών αποτελείται από δύο τιμές.

Η μεθοδολογία αυτή έχει ιδιαίτερη εφαρμογή σε *αριθμητικά προβλήματα ικανοποίησης περιορισμών* (numerical constraint satisfaction problems – NCSPs).

Η διάφορα αυτών των προβλημάτων από τα κλασικά ΠΙΠ έγκειται στο ότι τα πεδία τιμών των μεταβλητών αφορούν *διαστήματα πραγματικών αριθμών* [5]. Σε αυτή την εργασία θα πειραματιστούμε μόνο με κλασικά ΠΙΠ, όμως η θεωρία δεν αφορά μόνο αυτά.

5.1 Μία Διαφορετική Οπτική

Έστω λοιπόν ότι κατά τη φάση της αναζήτησης καλούμαστε να διχοτομήσουμε το πεδίο τιμών $D_X = \{1, \ldots, 10\}$ μίας περιορισμένης μεταβλητής X. Το σύνηθες σε αυτές τις περιπτώσεις είναι να περιορίσουμε το D_X, είτε σε $\{1, \ldots, 5\}$, είτε σε $\{6, \ldots, 10\}$: Θα επιλέξουμε ένα από τα δύο υποσύνολα και αν χρειαστεί στο μέλλον να αναιρέσουμε αυτή την επιλογή μας –σε μία πιθανή οπισθοδρόμηση–, θα μπορούμε να επιλέξουμε το έτερο υποσύνολο.

Ωστόσο, είναι εμφανές ότι αυτή η πρακτική υποθέτει ότι όλες οι τιμές ενός πεδίου τιμών είναι «ισοβαρείς». Π.χ. στο $D_X = \{1, \ldots, 10\}$, το ίδιο βάρος έχει το 1 ή το 2 με το 3, 4, 5 και γενικά οποιαδήποτε άλλη τιμή. Είναι προφανές όμως ότι σε πραγματικά προβλήματα αυτό δεν ισχύει. Επίσης είναι ευνόητο ότι δεν μπορούμε να ξέρουμε a priori το «βάρος», δηλαδή την αξία, μιας συγκεκριμένης τιμής ενός πεδίου τιμών· αν το ξέραμε, θα μπορούσαμε άμεσα να βρούμε λύση στο πρόβλημα, καθώς οι τιμές εκείνες που δεν συμμετέχουν σε λύση θα είχαν βάρος ίσο με 0. Οπότε καταφεύγουμε σε ευριστικές εκτιμήσεις.

Ο σκοπός ενός *ευριστικού επιλογής τιμής* (value ordering heuristic) είναι να διαλέξουμε την τιμή εκείνη από το πεδίο μιας μεταβλητής, που θα μας οδηγήσει με μεγαλύτερη πιθανότητα σε λύση (succeed-first heuristic) [4]. Ένα τέτοιο, ευρέως διαδεδομένο ευριστικό είναι η αρχή της επιλογής της τιμής εκείνης για την οποία έχουμε τις περισσότερες τιμές υποστήριξης (supporters). Π.χ. αν είχαμε τον περιορισμό $X \neq Y$, με $D_X = \{1, 2\}$ και $D_Y = \{2, 3\}$, και έπρεπε να επιλέξουμε μία τιμή από το D_X, θα επιλέγαμε το 1, καθώς έχει δύο τιμές υποστήριξης στο D_Y, ενώ το 2 έχει μόνο μία (την τιμή $3 \in D_Y$).

Αν θέλαμε να χρησιμοποιήσουμε το παραπάνω ευριστικό στη διχοτόμηση πεδίων τιμών, θα έπρεπε να το προσαρμόσουμε έτσι ώστε να δίνει πληροφορίες/εκτιμήσεις για *σύνολα* τιμών και όχι για ξεχωριστές τιμές ενός πεδίου. Στη διχοτόμηση πεδίων τιμών, αντί να κοβουμε στη μέση ένα πεδίο, θα ήταν καλύτερο να το χωρίζουμε σε δύο «ισοβαρή» μέρη όσον αφορά τον αριθμό των τιμών υποστήριξης για τα δύο σύνολα τιμών που θα προκύψουν.

Θα μελετήσουμε αυτό το ευριστικό για την περίπτωση που τα πεδία τιμών των εμπλεκόμενων μεταβλητών αφορούν σύνολα από διαδοχικούς ακέραιους

(της μορφής $\{a, a+1, \ldots, b\}$). Θα ασχοληθούμε ιδιαίτερα με τον περιορισμό '$<$'. Πιο τυπικά, θα ασχοληθούμε με τον περιορισμό $X < Y$, για τις περιορισμένες μεταβλητές X και Y, με πεδία τιμών $D_X = \{\underline{X}, \underline{X}+1, \ldots, \overline{X}\}$ και $D_Y = \{\underline{Y}, \underline{Y}+1, \ldots, \overline{Y}\}$.

5.2 Ο Περιορισμός $X < Y$

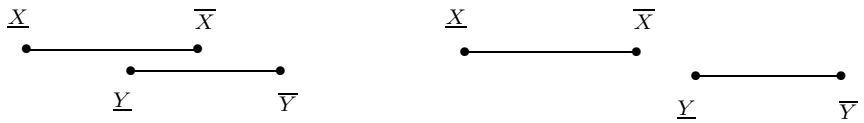
Ας πάρουμε ένα παράδειγμα. Έστω δύο μεταβλητές X και Y με πεδία τιμών $D_X = \{1, \ldots, 6\}$ και $D_Y = \{4, \ldots, 7\}$. Οι μεταβλητές αυτές συνδέονται με τον περιορισμό $X < Y$ και βρίσκονται σε κατάσταση συνέπειας ακμών –το οποίο γίνεται αντιληπτό εξετάζοντας τις τιμές των πεδίων τιμών. Αν επιθυμούμε να διχοτομήσουμε το πεδίο τιμών της X και να πάρουμε δύο υποσύνολα με ισάριθμα στοιχεία, θα πάρουμε το $A_1 = \{1, 2, 3\}$ και το $A_2 = \{4, 5, 6\}$. Για τα στοιχεία του A_1 έχουμε $\text{supp}_Y(1) = \text{supp}_Y(2) = \text{supp}_Y(3) = \{4, 5, 6, 7\}$, όπου $\text{supp}_Y(i)$ το υποσύνολο του D_Y με τις τιμές υποστήριξης για την τιμή $i \in D_X$. Δηλαδή έχουμε $\sum_{i \in A_2} |\text{supp}_Y(i)| = 4 + 4 + 4 = 12$. Για τα στοιχεία του A_2 ισχύει $\text{supp}_Y(4) = \{5, 6, 7\}$, $\text{supp}_Y(5) = \{6, 7\}$ και $\text{supp}_Y(6) = \{7\}$. Δηλαδή $\sum_{i \in A_2} |\text{supp}_Y(i)| = 3 + 2 + 1 = 6$.

Παρατηρούμε λοιπόν μία ανισοκατανομή όσον αφορά τις τιμές υποστήριξης: Για τις τιμές του A_1 υπάρχει αθροιστικά διπλάσιος αριθμός τιμών υποστήριξης σε σχέση με το A_2. Αν όμως αφαιρέσουμε την τιμή 3 από το A_1 και τη βάλουμε στο A_2, τα πράγματα εξισορροπούνται. ($\sum_{i \in A'_1} |\text{supp}_Y(i)| = 8$ και $\sum_{i \in A'_2} |\text{supp}_Y(i)| = 10$, όπου A'_1 και A'_2 τα σύνολα που προκύπτουν μετά από την «μετακίνηση» του 3. Παρατηρούμε ότι η απόλυτη τιμή της διαφοράς των δύο αθροισμάτων μειώθηκε σημαντικά.) Αν στο παράδειγμα είχαμε μεγαλύτερα πεδία τιμών η ανάγκη για εξισορρόπηση των αντίστοιχων συνόλων A_1 και A_2, που θα είχαμε πάρει από διχοτόμηση του D_X, θα ήταν ακόμα μεγαλύτερη.

Συνεπώς, είναι σημαντικό να μπορούμε με έναν συστηματικό τρόπο, αντί να κόβουμε απλά τα πεδία τιμών στη μέση, να έχουμε μία *δίκαιη διχοτόμηση πεδίων τιμών*. Θα πρέπει να βρίσκουμε σε σταθερό χρόνο (σε σχέση με το μέγεθος του πεδίου τιμών) την τιμή εκείνη που θα αποτελεί τη «χρυσή τομή».

Στο Σχήμα 5.1 έχουν αναπαρασταθεί γραφικά τα πεδία τιμών των μεταβλητών X και Y στον άξονα των ακεραίων αριθμών. Ισχύει ο περιορισμός $X < Y$ και υπάρχει συνέπεια ακμών, πράγμα που φαίνεται από τις σχετικές θέσεις των άκρων των πεδίων τιμών (και για τις δύο υποπεριπτώσεις).

Έστω $C = \min\{\overline{X}, \underline{Y} - 1\}$. «Κόβουμε» το D_X σε $D_1 = \{\underline{X}, \ldots, C\}$ και $D_2 = \{C + 1, \ldots, \overline{X}\}$. Σημειώνεται ότι το D_2 ενδέχεται να είναι κενό. Ισχύει



(α′) Περίπτωση $\overline{X} \geq \underline{Y}$ (β′) Περίπτωση $\overline{X} < \underline{Y}$

Σχήμα 5.1: Απεικόνιση των δύο υποπεριπτώσεων του περιορισμού $X < Y$

ότι $\forall i \in D_1$, $\text{supp}_Y(i) = D_Y$ και $\forall i \in D_2$, $\text{supp}_Y(i) = \{i+1, \ldots, \overline{Y}\}$. Από τα παραπάνω προκύπτουν τα εξής:

$$\sum_{i \in D_X} |\text{supp}_Y(i)| = \sum_{i \in D_1} |\text{supp}_Y(i)| + \sum_{i \in D_2} |\text{supp}_Y(i)| \tag{5.1}$$

$$\begin{aligned} \sum_{i \in D_1} |\text{supp}_Y(i)| &= \sum_{i=\underline{X}}^{C} |\text{supp}_Y(i)| \\ &= \sum_{i=\underline{X}}^{C} |D_Y| \\ &= (C - \underline{X} + 1)\,|D_Y| \end{aligned} \tag{5.2}$$

$$\begin{aligned} \sum_{i \in D_2} |\text{supp}_Y(i)| &= \sum_{i=C+1}^{\overline{X}} |\text{supp}_Y(i)| \\ &= \sum_{i=C+1}^{\overline{X}} \left(\overline{Y} - i\right) \\ &\overset{j:=i-C}{=} \sum_{j=1}^{\overline{X}-C} \left(\overline{Y} - j - C\right) \\ &= \frac{1}{2}\left(\overline{X} - C\right)\left(2\overline{Y} - \overline{X} - C - 1\right) \end{aligned} \tag{5.3}$$

Συνεπώς, από τη (5.1) και με τη βοήθεια των (5.2) και (5.3) προκύπτει μία αριθμητική παράσταση για τον υπολογισμό του ζητούμενου αθροίσματος. Έχοντας υπολογίσει αυτό το άθροισμα –σε σταθερό χρόνο–, είμαστε σε θέση να απαιτήσουμε την εύρεση μίας τιμής $a \in D_X$, τέτοιας ώστε το $\sum_{i=\underline{X}}^{a} |\text{supp}_Y(i)|$ να ισούται με $A = \epsilon \sum_{i \in D_X} |\text{supp}_Y(i)|$, $0 < \epsilon < 1$· μία συνηθισμένη τιμή για το ϵ είναι το 0.5. Δηλαδή θα πρέπει να λύσουμε ως προς το a την εξίσωση $\sum_{i=\underline{X}}^{a} |\text{supp}_Y(i)| = A$. Έχουμε τις εξής δύο περιπτώσεις:

(i) Αν $\sum_{i=\underline{X}}^{C} |\text{supp}_Y(i)| \geq A$, δηλαδή αν $(C - \underline{X} + 1)\,|D_Y| \geq A$, τότε αντικαθιστούμε το C στη σχέση (5.2) με a και προκύπτει:

$$\sum_{i=\underline{X}}^{a} |\text{supp}_Y(i)| = A \Longrightarrow \quad (a - \underline{X} + 1)\,|D_Y| = A \Longrightarrow$$

$$a = \underline{X} - 1 + \frac{A}{|D_Y|}.$$

(ii) Αλλιώς, θα χρησιμοποιήσουμε τη σχέση (5.1), αντικαθιστώντας το $\overline{X}$ με a και θα πάρουμε:

$$(5.1) \overset{(5.2),(5.3)}{\Longrightarrow} (C - \underline{X} + 1)\,|D_Y| + \frac{1}{2}(a - C)\left(2\overline{Y} - a - C - 1\right) = A \Longrightarrow$$

$$a^2 + pa + q = 0 \Longrightarrow$$

$$a = \frac{-p - \sqrt{p^2 - 4q}}{2}.$$

Όπου $p = 1 - 2\overline{Y}$ και $q = 2\overline{Y}C - C - C^2 - 2\,(C - \underline{X} + 1)\,|D_Y| + 2A$. Επιλέξαμε τη μικρότερη από τις δύο ρίζες του τριωνύμου. Εξάλλου, για τη μεγαλύτερη ρίζα έχουμε:

$$\frac{-p + \sqrt{p^2 - 4q}}{2} \geq \frac{-p}{2} = \overline{Y} - \frac{1}{2}.$$

Όμως, δεν μπορεί η τιμή του a να είναι μεγαλύτερη ή ίση του $\overline{Y}$, οπότε η συγκεκριμένη ρίζα απορρίπτεται.

5.3 Άλλοι Τύποι Περιορισμών

Ο περιορισμός $X < Y$ –με τα αντίστοιχα πεδία τιμών να περιλαμβάνουν έκαστο διαδοχικούς ακέραιους– παρουσιάζει ενδιαφέρον, καθώς αν εφαρμόσουμε την κλασική διχοτόμηση πεδίου τιμών και προκύψουν δύο υποσύνολα με ισάριθμα στοιχεία, εμφανίζεται στη γενική περίπτωση μία ασυμμετρία όσον αφορά τον αριθμό των τιμών υποστήριξης για το κάθε υποσύνολο. Στα περισσότερα άλλα είδη περιορισμών η κλασική διχοτόμηση πεδίων τιμών δεν αντιμετωπίζει σημαντικό πρόβλημα –μιλώντας πάντα για πεδία τιμών με διαδοχικούς ακέραιους. (Π.χ. ένα τέτοιο είδος περιορισμών είναι ο $X = CY$, με C σταθερό, ή ο $X = Y + Z$.)

Ένας άλλος περιορισμός που παρουσιάζει την προαναφερθείσα ασυμμετρία είναι ο $X = YZ$. (Π.χ. ας σκεφτούμε την περίπτωση που $D_X = \{1, \ldots, 16\}$ και $D_Y = D_Z = \{1, \ldots, 4\}$. Προφανώς εδώ μιλάμε για συνέπεια ορίων, μία πιο χαλαρή έννοια σε σχέση με τη συνέπεια ακμών· εξάλλου η τιμή π.χ. $15 \in D_X$ δεν βρίσκει στήριγμα στις Y και Z. Τότε αν διχοτομούσαμε το D_X δεν θα ήταν «δίκαιο» να το σπάσουμε σε $\{1, \ldots, 16/2\}$ και $\{16/2 + 1, \ldots, 16\}$, καθώς το δεύτερο υποσύνολο έχει σαφώς λιγότερες τιμές υποστήριξης στις μεταβλητές Y και Z, έναντι του πρώτου.) Ωστόσο, από τη μελέτη του συγκεκριμένου περιορισμού, εκτός του ότι καταλήγουμε σε πολύπλοκους μαθηματικούς τύπους, τα αποτελέσματα που προκύπτουν είναι προσεγγιστικά [12, 17, 19, 30].

5.4 Ο Περιορισμός $X = Y^2$

Μία υποπερίπτωση του περιορισμού $X = YZ$, που μπορεί να μελετηθεί ευκολότερα, είναι ο περιορισμός $X = Y^2$. Ας θωρήσουμε ότι υπάρχει συνέπεια ορίων για τις μεταβλητές X και Y που συνδέονται με αυτόν τον περιορισμό. (Η συνέπεια ορίων είναι πιο «χαλαρή» έννοια σε σχέση με τη συνέπεια ακμών. Εδώ αυτό πρακτικά σημαίνει ότι $\underline{X} = \underline{Y}^2$ και $\overline{X} = \overline{Y}^2$, αν $|\underline{Y}| \leq |\overline{Y}|$. Διαφορετικά, αν $|\underline{Y}| > |\overline{Y}|$, τότε θα πρέπει $\underline{X} = \overline{Y}^2$ και $\overline{X} = \underline{Y}^2$.) Τότε μία δίκαιη μοιρασιά του D_X –σύμφωνα πάντα με τη φιλοσοφία της δίκαιης διχοτόμησης πεδίων τιμών– θα ήταν τα δύο υποσύνολα

$$\{\underline{X}, \ldots, a\} \quad \text{και} \quad \{a+1, \ldots, \overline{X}\}, \quad \text{όπου } a = \left\lfloor \left(\frac{\underline{Y} + \overline{Y}}{2} \right)^2 \right\rfloor.$$

Απόδειξη. Έστω ότι $\underline{Y} + \overline{Y} \geq 0$. Για το υποσύνολο $\{\underline{X}, \ldots, a\}$ του D_X τα υποσύνολα υποστήριξης στη μεταβλητή Y είναι το $A_1 = \left\{\max\{0, \underline{Y}\}, \ldots, \left\lfloor \left(\underline{Y} + \overline{Y}\right)/2 \right\rfloor\right\}$ και, εφόσον $\underline{Y} < 0$, το $A_2 = \left\{\max\{-\left\lfloor \left(\underline{Y} + \overline{Y}\right)/2 \right\rfloor, \underline{Y}\}, \ldots, 0\right\}$. Για το έτερο υποσύνολο $\{a+1, \ldots, \overline{X}\}$ του D_X τα υποσύνολα υποστήριξης στην Y είναι το $B_1 = \left\{\left\lfloor \left(\underline{Y} + \overline{Y}\right)/2 \right\rfloor + 1, \ldots, \overline{Y}\right\}$ και, εφόσον $\underline{Y} < 0$, το $B_2 = \left\{\underline{Y}, \ldots, -\left\lfloor \left(\underline{Y} + \overline{Y}\right)/2 \right\rfloor - 1\right\}$. Ισχύει $|A_1 \cup A_2| = |B_1 \cup B_2|$, με μία μικρή απόκλιση της τάξης μίας μονάδας. Παρόμοια είναι η απόδειξη στην περίπτωση που $\underline{Y} + \overline{Y} < 0$. □

5.5 Περισσότεροι Περιορισμοί

Ως τώρα ασχοληθήκαμε με την εύρεση μίας τιμής a η οποία θα διαχωρίζει δίκαια (αναφορικά με τις τιμές υποστήριξης) ένα πεδίο τιμών D_X, όταν υπάρχει *ένας* περιορισμός, π.χ. $X < Y$. Τι γίνεται όμως όταν έχουμε και άλλους περιορισμούς (π.χ. έναν ακόμα περιορισμό $X < Z$); Μία λογική αντιμετώπιση θα ήταν να πάρουμε τη μέση τιμή όλων των τιμών «a» (ή, καλύτερα, «a_i», όπου i ο αύξων αριθμός του περιορισμού) που προκύπτουν για κάθε περιορισμό.

Για να γίνει αυτό, δηλαδή για να βρούμε τη «χρυσή τομή» που δίνει η δίκαιη διχοτόμηση πεδίων τιμών, απαιτούνται $O(e)$ βήματα, αφού τόσες προσθέσεις θα κάνουμε για να βρούμε τον μέσο όρο. Στην απλή διχοτόμηση πεδίων τιμών ο χρόνος ήταν σταθερός με τίμημα μία χειρότερη εκτίμηση.

Η τιμή a_i θα ισούται με $\left[\left(\underline{X} + \overline{X}\right)/2\right]$ για τους περισσότερους περιορισμούς, όχι όμως και για τους ανισοτικούς που μελετήθηκαν προηγουμένως. Σημειώνεται επίσης ότι όταν ένας περιορισμός ικανοποιείται για κάθε συνδυασμό των τιμών των τρεχόντων πεδίων (όπως π.χ. ο περιορισμός $X < Y$, με $D_X = \{0, 1\}$ και $D_Y = \{8, 9\}$), τότε δεν είναι ανάγκη να τον λάβουμε υπόψη (για να βγάλουμε τον μέσο όρο).

Αν λύνουμε ένα πρόβλημα ικανοποίησης περιορισμών *με βάρη*, θα μπορούσαμε αντί της μέσης τιμής να παίρναμε την τιμή $\left[\frac{1}{e}\sum_{i=1}^{e} w_i a_i\right]$, όπου e ο αριθμός των περιορισμών του προβλήματος και w_i το βάρος του i-οστού περιορισμού.

Εμείς στα πειράματα που ακολουθούν, βάζαμε οι ίδιοι βάρη στις τιμές a_i σε ένα ΠΙΠ χωρίς βάρη. Όταν είχαμε π.χ. τους περιορισμούς $X < Y_1$ με $|D_{Y_1}| = 4$ και $X < Y_2$ με $|D_{Y_2}| = 40$, θελήσαμε να ευνοήσουμε περισσότερο την τιμή διχοτόμησης a_1 η οποία προκύπτει για τον πρώτο περιορισμό, παρά την a_2. Ο λόγος που το κάναμε αυτό ήταν επειδή η πράξη έχει δείξει ότι πρέπει να λαμβάνουμε περισσότερο υπόψη τις μεταβλητές εκείνες με μικρά πεδία τιμών, για να έχουμε περισσότερη ελευθερία κινήσεων στην πορεία της αναζήτησης [32]. Π.χ. ένας υποδιπλασιασμός του D_{Y_2} είναι πιθανότερο να έχει λιγότερο αντίκτυπο από μία μείωση του D_{Y_1} που είναι ήδη περιορισμένο. Το βάρος που θα μπορούσαμε να βάλουμε στις αντίστοιχες τιμές a_i είναι αντιστρόφως ανάλογο του μεγέθους του πεδίου τιμών της Y (δηλαδή ίσο με $1/|D_Y|$).

Τέλος, όταν τελικά κόψουμε το D_X στα δύο υποσύνολα $\{\underline{X}, \ldots, a\}$ και $\{a+1, \ldots, \overline{X}\}$, δεν είναι απαραίτητο να επιλέγουμε πρώτα το αριστερό υποσύνολο για να συνεχίσουμε την αναζήτηση· θεωρήσαμε πρακτικό να επιλέγουμε πρώτα το υποσύνολο με το μικρότερο μέγεθος, καθώς –χωρίς να μιλούμε με αυστηρή γλώσσα– οι τιμές που αυτό περιέχει φαίνεται ότι έχουν περισσότερη αξία, λόγω των περισσότερων τιμών υποστήριξης που έχει η καθεμία τους.

5.6 Πειραματικά Αποτελέσματα

Στο Κεφάλαιο 3 χρησιμοποιήσαμε το πρόβλημα κατάρτισης ωρολογίων προγραμμάτων για να εντοπίσουμε την όποια χρησιμότητα των προτεινόμενων αλγορίθμων και το ίδιο θα κάνουμε και εδώ. Υιοθετήσαμε την περιγραφή του προβλήματος όπως δημοσιεύθηκε στον δεύτερο αντίστοιχο διεθνή διαγωνισμό (International Timetabling Competition 2007). Στο Παράρτημα Α′ παρατίθενται αναλυτικές πληροφορίες για το πρόβλημα αυτό.

Χρησιμοποιήσαμε ξανά τον επιλυτή προβλημάτων ικανοποίησης περιορισμών Naxos και τον μηχανισμό διατύπωσης μεθόδων αναζήτησης που ενσωματώνεται σε αυτόν [42]. Οι μέθοδοι που υλοποιήσαμε ήταν η κλασική διχοτόμηση πεδίων τιμών και η δίκαιη διχοτόμηση πεδίων τιμών.

Για κάθε δυνατό συνδυασμό των δύο μεθοδολογιών με τα 14 στιγμιότυπα του προβλήματος (που δημοσιεύθηκαν στον διαγωνισμό) αφήσαμε τον Naxos να τα επιλύει βρίσκοντας διαδοχικά λύσεις, με τη μία λύση να είναι υποχρεωτικά καλύτερη από την προηγούμενή της, σύμφωνα με τη μεθοδολογία *διακλάδωσε-και-φράξε* (branch-and-bound). Ο χρόνος που έτρεχε ο Naxos για κάθε ΠΙΠ ήταν προκαθορισμένος από τον διαγωνισμό μέσω ενός προγράμματος (benchmark) που ανέλυε τις δυνατότητες κάθε υπολογιστή. (Π.χ. σε ένα μηχάνημα με διπύρηνο επεξεργαστή Intel Core στα 2.8 GHz, το χρονικό όριο που έπρεπε να τεθεί ήταν 7 λεπτά και 20 δευτερόλεπτα.)

5.6.1 Υβριδικό Ευριστικό Επιλογής Μεταβλητής

Για να λύσουμε ένα ΠΙΠ επιλέγουμε επαναληπτικά μία μεταβλητή του και είτε της αναθέτουμε μία τιμή, είτε περιορίζουμε με κάποιον τρόπο (π.χ. διχοτόμηση) το πεδίο τιμών της. Ποια μεταβλητή είναι όμως καλό να επιλέγουμε κάθε φορά; Το ευριστικό επιλογής μεταβλητής (variable ordering heuristic) μας καθοδηγεί όταν έχουμε να πάρουμε τέτοιου είδους αποφάσεις.

Η γενικότερη εμπειρία έχει δείξει ότι όταν έχουμε να κάνουμε με *ανάθεση* τιμής, η χρήση των παρακάτω ευριστικών κανόνων έχει σαν αποτέλεσμα τη γρηγορότερη επίλυση των περισσότερων ΠΙΠ:

- Επιλέγουμε από το σύνολο των μεταβλητών τη μεταβλητή εκείνη που έχει το *μικρότερο* πεδίο τιμών. (Στη βιβλιογραφία συναντάμε διάφορες ονομασίες για αυτό το ευριστικό, όπως *ελάχιστες απομένουσες τιμές*, minimum remaining values και first-fail.)

- Αν υπάρχει ισοβαθμία για το παραπάνω κριτήριο, τότε συμβουλευόμαστε το δεύτερο κριτήριο. Το δεύτερο κριτήριο ευνοεί τη μεταβλητή η οποία εμπλέκεται στους *περισσότερους* περιορισμούς· δηλαδή τη μεταβλητή V με το *μεγαλύτερο* σύνολο $\{c \mid V \in \text{vars}(c)\}$. (Το κριτήριο αυτό το συναντάμε με τις ονομασίες *ευριστικό βαθμού*, degree heuristic και Brélaz προς τιμήν του ερευνητή που το πρωτοχρησιμοποίησε το 1979 [11].)

Η διχοτόμηση πεδίων τιμών και η παραλλαγή της που εξετάζουμε, δεν είναι σύνηθες να χρησιμοποιούν τα καθιερωμένα ευριστικά επιλογής μεταβλητής, όπως τα προαναφερθέντα. Ο λόγος που δεν το κάνουν αυτό είναι επειδή αν χρησιμοποιήσουν π.χ. το ευριστικό των ελαχίστων απομενουσών τιμών, η μέθοδος της διχοτόμησης πεδίων τιμών θα ταυτιζόταν με την πρώτα-κατά-βάθος αναζήτηση: Θα επέλεγε μία μεταβλητή και θα διχοτομούσε το πεδίο τιμών της, αλλά στη συνέχεια θα επέλεγε πιθανότατα την ίδια πάλι μεταβλητή (λόγω του ότι πάλι θα έχει το μικρότερο πεδίο τιμών) και θα ξαναδιχοτομούσε το πεδίο τιμών της κ.ο.κ. μέχρι αυτό να γίνει μονομελές. Πρακτικά δηλαδή θα είχαμε αναθέσεις.

Για να μην φτάσουμε σε αυτήν την «εκφυλισμένη» περίπτωση, ναι μεν συμβουλευόμαστε τα παραπάνω ευριστικά, αλλά προσθέτουμε έναν βαθμό τυχαιότητας σε αυτά. Αν έχουμε να επιλέξουμε μία από τις μεταβλητές $V_1, \ldots, V_n$, αξιολογούμε καταρχήν καθεμία από αυτές μέσω του παρακάτω τύπου:

$$h_i = \Big(|D_{V_i}| \cdot (e+1) \;+\; \big|\{c \mid V_i \in \text{vars}(c)\}\big|\Big)^{rand}.$$

Ο τύπος εντός της παρένθεσης δεν είναι τίποτα άλλο από ένας συνδυασμός των δύο ευριστικών που αναφέρθηκαν. Ο παράγοντας $(e+1)$ υπάρχει για να δώσει προτεραιότητα στο ευριστικό των ελαχίστων απομενουσών τιμών $|D_{V_i}|$, αφού για το ευριστικό του βαθμού ισχύει ότι $\big|\{c \mid V_i \in \text{vars}(c)\}\big| \leq e$, όπου e ο αριθμός των περιορισμών που υπάρχουν στο ΠΙΠ.

Η παρένθεση υψώνεται στη $rand$. Όσο μεγαλύτερη είναι η τιμή $rand$, τόσο περισσότερο θα μεγαλώσει η διαφορά μεταξύ των h_i. Δηλαδή όσο μεγαλώνουμε το $rand$, τόσο «υποχρεωνόμαστε» να ακολουθήσουμε την εκτίμηση των δύο προαναφερθέντων ευριστικών.

Σε αυτό το σημείο θα επιχειρήσουμε να εισάγουμε μία «δόση» τυχαιότητας στα δύο παραπάνω –μη τυχαία– ευριστικά. Ορίζουμε την εξής ακολουθία:

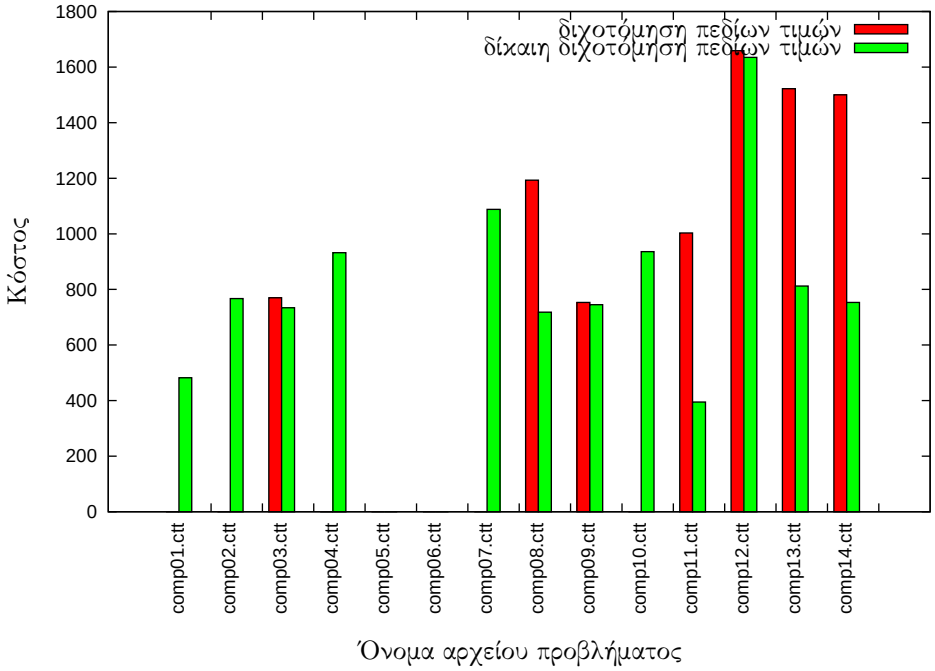
$$H_0 = 0,$$
$$H_i = \frac{h_i}{\sum_{j=1}^{n} h_j} + H_{i-1}, \quad 1 \leq i \leq n.$$

Από αυτήν προκύπτουν τα εξής n διαστήματα τιμών, τα οποία είναι ξένα μεταξύ τους:

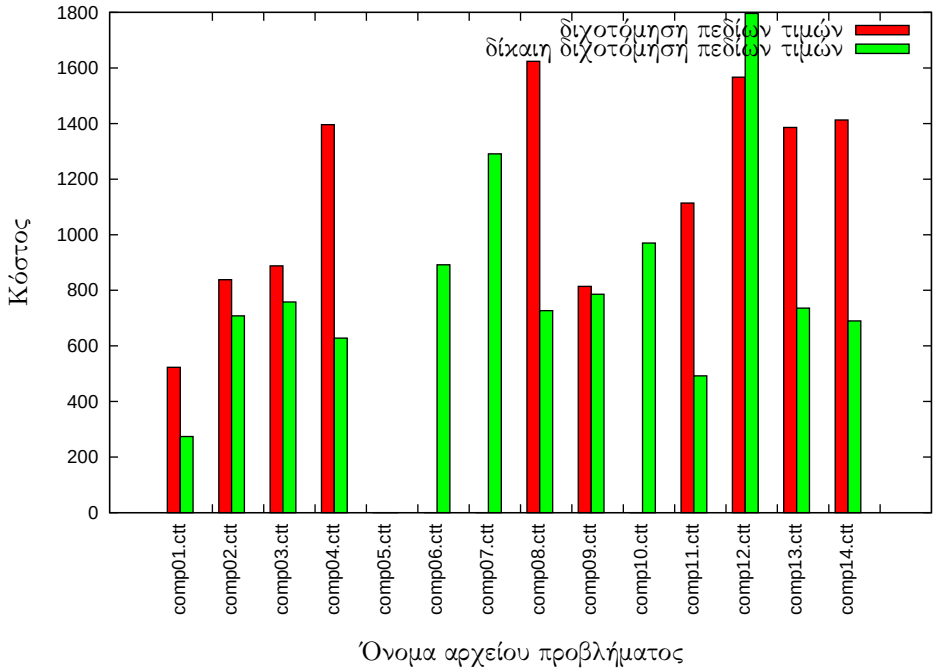
$$I_i = [H_{i-1}, H_i), \quad 1 \leq i \leq n.$$

Η ένωση όλων αυτών των διαστημάτων δίνει το σύνολο $[0, 1)$. Ανάλογα με το πόσο μεγάλη είναι η τιμή h_i, το διάστημα I_i καταλαμβάνει αντίστοιχη μερίδα από το $[0, 1)$. Επομένως επιλέγοντας μία τυχαία τιμή στο $[0, 1)$, μπορούμε να πάρουμε το αντίστοιχο I_i, το οποίο αφορά μία συγκεκριμένη μεταβλητή V_i.

Όταν $rand \to \infty$, τότε η μεταβλητή που θα πάρουμε από την παραπάνω διαδικασία θα είναι αυτή που επιλέγουν τα μη τυχαία ευριστικά που χρησιμοποιήσαμε. Αν όμως $rand \to 0$, τότε η παραπάνω διαδικασία θα επιλέξει ένα τυχαίο διάστημα I_i και συνακόλουθα μία εντελώς τυχαία μεταβλητή V_i.



Σχήμα 5.2: Τα κόστη των λύσεων που βρέθηκαν από τις δύο διαφορετικές μεθοδολογίες που εφαρμόστηκαν (σε καθένα από τα 14 προβλήματα κατάρτισης ωρολογίου προγράμματος) για $rand = 8$

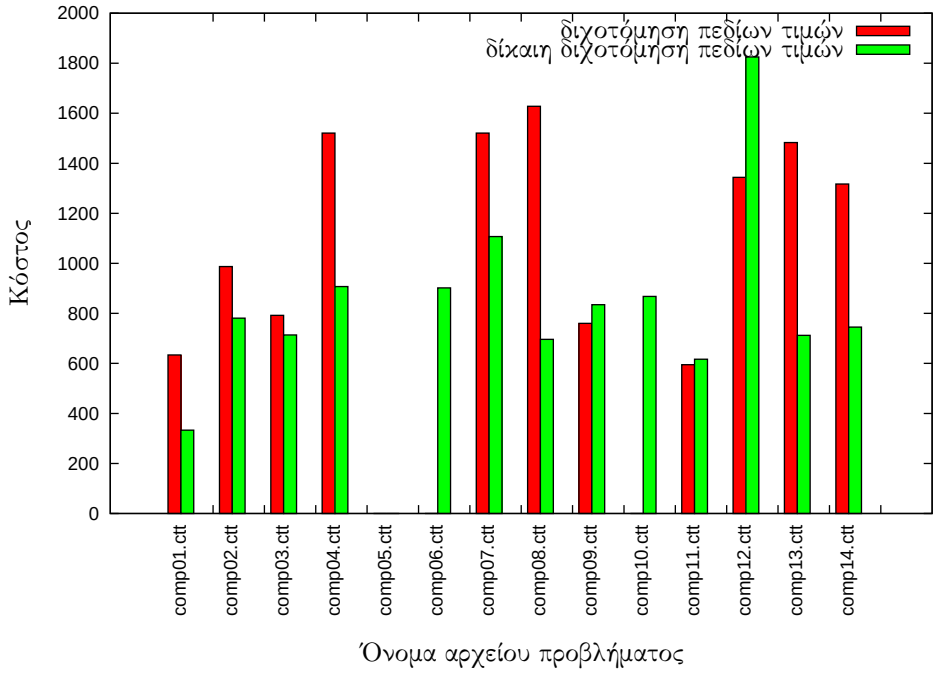


Σχήμα 5.3: Τα κόστη των λύσεων για $rand = 16$

5.6.2 Αποτελέσματα Βελτιστοποίησης

Στα σχήματα που παρατίθενται συγκρίνονται τα κόστη των καλύτερων λύσεων που βρίσκουν οι μεθοδολογίες της κλασικής και της δίκαιης διχοτόμησης πεδίων τιμών. Κάθε διάγραμμα αντιστοιχεί και σε έναν διαφορετικό βαθμό τυχαιότητας $rand$: το πρώτο σχήμα έχει την περισσότερη τυχαιότητα, η οποία ελαττώνεται σταδιακά μέχρι να φτάσουμε στο τελευταίο. Στα σημεία που δεν φαίνεται το κόστος, η μέθοδος δεν κατάφερε να βρει καμία λύση.

Βλέπουμε ότι με τη δίκαιη διχοτόμηση πεδίων τιμών φτάνουμε στο 90% των περιπτώσεων σε καλύτερες λύσεις σε σχέση με την απλή διχοτόμηση πεδίων τιμών. Κάποιες φορές η κλασική μέθοδος δεν βρίσκει καμία λύση, ενώ η δίκαιη διχοτόμηση πεδίων τιμών καταφέρνει να επιλύσει το πρόβλημα. Ενώ η δίκαιη διχοτόμηση δεν έχει σχεδιαστεί για προβλήματα βελτιστοποίησης (αλλά για ΠΙΠ στα οποία η εύρεση μίας οποιασδήποτε λύσης μάς αρκεί), η ταχύτητα με την οποία φτάνει σε μία λύση αφήνει τα χρονικά περιθώρια για να αναζητήσουμε τη μείωση του κόστους της. Δεν είναι ωστόσο ξεκάθαρο το ποιο $rand$ είναι καλύτερο να επιλέξουμε.



Σχήμα 5.4: Τα κόστη των λύσεων για $rand = 32$

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Η εμπειρία έχει δείξει ότι τα ΠΙΠ και ιδιαίτερα τα δύσκολα (NP-πλήρη) δεν είναι σοφό να τα βάζουμε όλα στα ίδιο «καζάνι» και να τα επιλύουμε κατά τον ίδιο τρόπο. Όμως η αρχική προσδοκία για τον Προγραμματισμό με Περιορισμούς –όπως εκφράστηκε από έναν από τους πρωτοπόρους του χώρου, τον Eugene Freuder– ήταν αυτό ακριβώς αυτό το πράγμα: Ο χρήστης να διατυπώνει με απλό τρόπο το ΠΙΠ του και ο επιλυτής να αναλαμβάνει την επίλυσή του, χωρίς την ουσιαστική παρέμβαση του χρήστη [15]. Οι μεθοδολογίες αυτής της εργασίας μπορούν να ενταχθούν σε έναν επιλυτή *γενικών* ΠΙΠ, αφού τα πειραματικά αποτελέσματα τα οποία παραθέσαμε προέκυψαν μέσα από την επέκταση ενός τέτοιου γενικού επιλυτή.

Η ιδέα της απεικόνισης ενός πεδίου τιμών μέσα από ένα δυαδικό δένδρο αναζήτησης το οποίο περιέχει τα *κενά* του, είδαμε ότι είναι εξαιρετικά χρήσιμη όταν έχουμε να κάνουμε με μεγάλα πεδία τιμών. Από εδώ και πέρα είναι ενδιαφέρον να εξετάσουμε τη δημιουργία ενός υβριδίου για την απεικόνιση ενός πεδίου τιμών, το οποίο θα ενσωματώνει περισσότερες από μία δομές δεδομένων.

Η μηχανή διάδοσης περιορισμών AC-5+ προέκυψε από το «πάντρεμα» της θεωρίας στην οποία στηρίζονται οι μηχανές διάδοσης περιορισμών που χρησιμοποιούνται στους σύγχρονους επιλυτές και της πιο «κλασικής» θεωρίας των πρώτων αλγορίθμων επιβολής συνέπειας ακμών –συγκεκριμένα του AC-5. Θα ήταν ενδιαφέρον να δούμε μελλοντικά το πώς μπορεί να γίνει αυτή η μεθοδολογία κατανεμημένη, έτσι ώστε να αυξήσουμε την αποδοτικότητά της, αξιοποιώντας όλους τους υπολογιστικούς πόρους που έχουμε στη διάθεσή μας.

Τέλος, η επέκταση που κάναμε στη μεθοδολογία της διχοτόμησης πεδίων τι-

μών θα μπορούσε να ιδωθεί από τη σκοπιά και άλλων περιορισμών, πέραν αυτών που εξετάσαμε, όπως για παράδειγμα τους καθολικούς. Τα ενθαρρυντικά αποτελέσματα που πήραμε για τα στιγμιότυπα ενός κλασικού ΠΙΠ πιστεύουμε ότι μπορούν να επιδειχθούν καλύτερα σε ένα *αριθμητικό* ΠΙΠ –δηλαδή σε ένα πρόβλημα που αφορά πραγματικούς αριθμούς και όχι μόνο ακέραιες μεταβλητές–, αφού σε αυτά εφαρμόζεται πιο φυσικά η διχοτόμηση πεδίων τιμών.

Παράρτημα Α΄

Πρόβλημα Κατάρτισης Ωρολογίου Προγράμματος

Η αυτόματη κατασκευή ενός ωρολογίου προγράμματος είναι ένα δύσκολο πρόβλημα που μπορεί να αντιμετωπισθεί μέσω του Προγραμματισμού με Περιορισμούς. Για την παρούσα εργασία αποτέλεσε τον συνδετικό κρίκο ανάμεσα στη θεωρία και στην πράξη, αφού πρόκειται για ένα από τα πιο πρακτικά προβλήματα μεγάλης κλίμακας.[1]

Υπάρχουν πολλές διαφορετικές διατυπώσεις τέτοιου είδους προβλημάτων χρονοπρογραμματισμού. Εμείς μοντελοποιήσαμε το πρόβλημα σύμφωνα με τις προδιαγραφές του τελευταίου Διεθνούς Διαγωνισμού Κατάρτισης Ωρολογίων Προγραμμάτων (International Timetabling Competition – ITC-2007). Πιο συγκεκριμένα, επικεντρωθήκαμε στον τομέα Κατάρτισης Ωρολογίου Προγράμματος Μαθημάτων βάσει Προγράμματος Σπουδών (Curriculum-based Course Timetabling) [14].

Πέρα από τη μοντελοποίηση των προβλημάτων, δανειστήκαμε από τον εν λόγω διαγωνισμό και δεκατέσσερα στιγμιότυπα προβλημάτων. (Τα προβλήματα αυτά περιγράφονται στα αρχεία `comp01.ctt`, ..., `comp14.ctt` που είναι διαθέσιμα στην ιστοσελίδα του διαγωνισμού και χρησιμοποιήθηκαν για την αξιολόγηση των αλγορίθμων που υποβλήθηκαν σε αυτόν.) Τα στιγμιότυπα αφορούν πραγματικά προβλήματα ωρολογίων προγραμμάτων του Πανεπιστήμιου

[1] Η ποικιλία των περιορισμών τους οποίους εμπεριέχει, κάνει αυτό το πρόβλημα κατάλληλη δοκιμαστική πλατφόρμα για τη μηχανή διάδοσης περιορισμών (AC-5+) που παρουσιάστηκε στο Κεφάλαιο 3 και για το μηχανισμό της Δίκαιης Διχοτόμησης Πεδίων Τιμών του Κεφαλαίου 5.

του Ούντινε.[2]

Α΄.1 Εισαγωγή

Αναλυτικότερα, το πρόβλημα έχει σαν ζητούμενο τον χρονοπρογραμματισμό συγκεκριμένου αριθμού μονόωρων διαλέξεων –καθεμία από τις οποίες αφορά ένα μάθημα–, έτσι ώστε να κατασκευαστεί ένα εβδομαδιαίο ωρολόγιο πρόγραμμα. Πολλά διεθνή πανεπιστήμια βασίζονται στο μοντέλο που θα περιγράψουμε, αν και σε μερικά σημεία έχει απλοποιηθεί έτσι ώστε να είναι όσο το δυνατό γενικότερο.

Α΄.2 Οντότητες

Ημέρες, Ώρες και Περίοδοι. Στο πρόβλημα ορίζεται ο αριθμός των *ημερών* κατά τις οποίες διδάσκονται μαθήματα (συνήθως 5 ή 6). Κάθε μέρα υποδιαιρείται σε έναν σταθερό αριθμό *ωρών*, ο οποίος είναι ο ίδιος για όλες τις ημέρες. Μία *περίοδος* είναι ένα ζεύγος μίας συγκεκριμένης ημέρας και ώρας. Ο συνολικός αριθμός των διδακτικών περιόδων ισούται με το γινόμενο του αριθμού των ημερών επί τον αριθμό των ωρών (ανά ημέρα).

Μαθήματα και Καθηγητές. Κάθε *μάθημα* περιλαμβάνει έναν σταθερό αριθμό *διαλέξεων*, οι οποίες παραδίδονται σε διαφορετικές μεταξύ τους περιόδους· σημειώνεται ότι γίνεται να οριστούν περίοδοι κατά τις οποίες δεν επιτρέπεται η διδασκαλία του μαθήματος. Ένα μάθημα παρακολουθείται από συγκεκριμένο αριθμό *μαθητών* και διδάσκεται από έναν *καθηγητή*. Οι διαλέξεις καλό είναι να κατανέμονται μέσα στην εβδομάδα, έτσι ώστε να καλύπτουν έναν *ελάχιστο αριθμό ημερών διδασκαλίας* του μαθήματος.

Αίθουσες. Κάθε *αίθουσα* χαρακτηρίζεται από μία συγκεκριμένη *χωρητικότητα*, που είναι ο αριθμός των διαθέσιμων θέσεών της. Όλες οι αίθουσες είναι κατάλληλες για όλα τα μαθήματα (εφόσον επαρκούν οι θέσεις τους).

Ομάδες Μαθημάτων. Δύο μαθήματα ανήκουν στην ίδια *ομάδα μαθημάτων* αν και μόνον αν υπάρχουν μαθητές που παρακολουθούν και τα δύο.

[2]Για την ιστορία, οι νικητές του διαγωνισμού χρησιμοποίησαν τοπική αναζήτηση [31], ενώ στην παρούσα εργασία εστιάσαμε στις συστηματικές μεθόδους αναζήτησης με οπισθοδρόμηση.

Συνεπώς, τα μαθήματα που ανήκουν στην ίδια ομάδα θα πρέπει να παραδίδονται σε διαφορετικές διδακτικές περιόδους, αλλιώς έχουμε τις λεγόμενες *συγκρούσεις* διαλέξεων. Τέλος, καλό είναι οι διδακτικές περίοδοι κατά τις οποίες παραδίδονται οι διαλέξεις μίας ομάδας μαθημάτων να μην έχουν πολλά *κενά* μεταξύ τους· δηλαδή καλό είναι το ωρολόγιο πρόγραμμα για μία συγκεκριμένη ομάδα μαθημάτων να είναι όσο το δυνατόν πιο συμπαγές.

Η *λύση* στο πρόβλημα είναι μία ανάθεση μίας περιόδου (δηλαδή ημέρας και ώρας) και μίας αίθουσας σε κάθε διάλεξη κάθε μαθήματος.

Α΄.3 (Αυστηροί) Περιορισμοί

Διαλέξεις. Όλες οι διαλέξεις κάθε μαθήματος πρέπει να μπουν στο ωρολόγιο πρόγραμμα, σε διαφορετικές διδακτικές περιόδους.

Χρήση Αίθουσας. Δύο διαλέξεις δεν μπορούν να λαμβάνουν χώρα στην ίδια αίθουσα την ίδια διδακτική περίοδο.

Συγκρούσεις. Οι διαλέξεις μίας ομάδας μαθημάτων καθώς και οι διαλέξεις που διδάσκει ο ίδιος ο καθηγητής πρέπει να παραδίδονται σε διαφορετικές διδακτικές περιόδους.

Διαθεσιμότητες. Κάθε διάλεξη πρέπει να παραδίδεται σε διδακτική περίοδο διαθέσιμη για το μάθημα. (Όπως προαναφέρθηκε, γίνεται να οριστούν περίοδοι μη διαθέσιμες για κάποιο μάθημα.)

Α΄.4 Ποιότητα Ωρολογίου Προγράμματος

Η ποιότητα μίας λύσης, ονομάζεται και *κόστος* της: όσο μικρότερο είναι το κόστος, τόσο πιο ποιοτική είναι η λύση. Το κόστος είναι μία *αντικειμενική συνάρτηση* (objective function) κάποιων ποσοτικών παραγόντων που θα απαριθμήσουμε. (Μία άλλη ονομασία για το κόστος είναι οι λεγόμενοι *χαλαροί περιορισμοί* (soft constraints).)

Χωρητικότητα Αίθουσας. Για κάθε διάλεξη, ο αριθμός των μαθητών που την παρακολουθούν δεν πρέπει να υπερβαίνει τη χωρητικότητα της αίθουσας. **Κάθε «υπεράριθμος» μαθητής προσθέτει 1 πόντο ποινής στο κόστος της λύσης**.

Ελάχιστος Αριθμός Ημερών Διδασκαλίας. Οι διαλέξεις κάθε μαθήματος πρέπει να «απλώνονται» τουλάχιστον στον αντίστοιχο (δεδομένο) *ελάχιστο αριθμό ημερών διδασκαλίας* μέσα στην εβδομάδα. **Κάθε μέρα κάτω του ελαχίστου μετράει σαν 5 πόντοι ποινής.**

Κενές Περίοδοι Ομάδας Μαθημάτων. Δύο διαλέξεις που ανήκουν στην ίδια ομάδα μαθημάτων και λαμβάνουν χώρα την ίδια ημέρα, πρέπει να γίνονται σε γειτονικές ώρες (δηλαδή να μην υπάρχουν κενές ώρες μεταξύ τους, για να μην περιμένουν άσκοπα οι μαθητές που τις παρακολουθούν). **Κάθε «απομονωμένη» διάλεξη, δηλαδή κάθε διάλεξη της οποίας δεν προηγείται και δεν έπεται άλλη διάλεξη που ανήκει στην ίδια ομάδα μαθημάτων, προσμετράται σαν 2 πόντοι ποινής.**

Σταθερότητα Επιλογής Αίθουσας. Οι διαλέξεις ενός μαθήματος πρέπει να λαμβάνουν χώρα στην ίδια αίθουσα. **Κάθε επιπλέον αίθουσα που θα χρησιμοποιηθεί για το μάθημα –πέραν της πρώτης– μετράει σαν 1 πόντος ποινής.**

Παράρτημα Β′

Πηγαίος Κώδικας για το Πρόβλημα Ακολουθίας DNA της § 4.3.2

Στην § 4.3.2 ορίσαμε ένα πρόβλημα με μεταβλητές που περιέχουν ένα ευρύ πεδίο τιμών (247,200,000 τιμές). Παρακάτω παρουσιάζεται ο κώδικας που το επιλύει, για τα τρία διαφορετικά συστήματα προγραμματισμού με περιορισμούς που χρησιμοποιήθηκαν.

Β′.1 Κώδικας ECL^{i}PS^e

Αρχίζουμε με την προσέγγιση του λογικού προγραμματισμού, η οποία είναι αν μη τι άλλο εύγλωττη. Εκμεταλλευτήκαμε τη βιβλιοθήκη IC (Interval Constraints – Περιορισμοί Διαστημάτων) της ECL^{i}PS^e μέσω του κατηγορήματος `lib(ic)`.

```
  Nucleotides = [C1, C2, C3, C4, G1, G2, G3, G4],
  Nucleotides :: 1..247200000,

  %  Cytosines geometric progression.  %
  C1 #= C2 / 99,
  C2 #= C3 / 99,
  C3 #= C4 / 99,

  %  Guanines arithmetic progression.  %
```

```
  G2 #= G1 + 99,
  G3 #= G2 + 99,
  G4 #= G3 + 99,

  alldifferent(Nucleotides),
  labeling(Nucleotides).
```

Το παραπάνω δίνει τα εξής αποτελέσματα:

```
  Nucleotides = [1, 99, 9801, 970299, 2, 101, 200, 299]
  C1 = 1
  C2 = 99
  C3 = 9801
  C4 = 970299
  G1 = 2
  G2 = 101
  G3 = 200
  G4 = 299
```

Β′.2 Κώδικας Επιλυτή ILOG

Παρακάτω αντιμετωπίζουμε το πρόβλημα μέσω διαδικαστικού –και όχι λογικού– προγραμματισμού με περιορισμούς, χρησιμοποιώντας τον επιλυτή ILOG. Σημειώνεται ότι έχει ενεργοποιηθεί το σύστημα ελέγχου λαθών μέσω εξαιρέσεων (exceptions).

```
#include <ilsolver/ilcint.h>

int  main (void)
{
  try  {

    IlcManager pm(IlcEdit);
    pm.useExceptions(IlcTrue);

    IlcIntVarArray  Nucleotides(pm, 8, 1, 247200000);

    //  Cytosines geometric progression.  //
```

```
    pm.add( Nucleotides[0] == Nucleotides[1] / 99 );
    pm.add( Nucleotides[1] == Nucleotides[2] / 99 );
    pm.add( Nucleotides[2] == Nucleotides[3] / 99 );

    //  Guanines arithmetic progression.  //
    pm.add( Nucleotides[5] == Nucleotides[4] + 99 );
    pm.add( Nucleotides[6] == Nucleotides[5] + 99 );
    pm.add( Nucleotides[7] == Nucleotides[6] + 99 );

    pm.add( IlcAllDiff(Nucleotides) );

    pm.add( IlcGenerate(Nucleotides) );

    if ( pm.nextSolution()  ==  IlcTrue )
        pm.out() << Nucleotides << "\n";

    pm.end();

  }  catch (IlcFailException& exc)  {
      cerr << "IlcFailException" << "\n";

  }  catch (exception& exc)  {
      cerr << "exception" << "\n";

  }  catch (...)  {
      cerr << "Unknown exception" << "\n";
  }
}
```

Β′.3 Κώδικας Επιλυτή Naxos

Ο Naxos βασίζεται στην ίδια γλώσσα προγραμματισμού με τον Ilog, τη C++ και το συντακτικό των δύο επιλυτών μοιράζεται πολλά κοινά σημεία. Έτσι ο παρακάτω πηγαίος κώδικας μοιάζει αρκετά με αυτόν του Ilog.

```
#include <naxos.h>
#include <iostream>
```

```
using namespace std;
using namespace naxos;

int  main (void)
{
  try  {

    NsProblemManager  pm;

    NsIntVarArray  Nucleotides;
    for (NsIndex i=0;  i < 8;  ++i)
        Nucleotides.push_back( NsIntVar(pm, 1, 247200000) );

    //  Cytosines geometric progression.  //
    pm.add( Nucleotides[0] == Nucleotides[1] / 99 );
    pm.add( Nucleotides[1] == Nucleotides[2] / 99 );
    pm.add( Nucleotides[2] == Nucleotides[3] / 99 );

    //  Guanines arithmetic progression.  //
    pm.add( Nucleotides[5] == Nucleotides[4] + 99 );
    pm.add( Nucleotides[6] == Nucleotides[5] + 99 );
    pm.add( Nucleotides[7] == Nucleotides[6] + 99 );

    pm.add( NsAllDiff(Nucleotides) );

    pm.addGoal( new NsgLabeling(Nucleotides) );

    if ( pm.nextSolution()  ==  true )
        cout << Nucleotides << "\n";

  }  catch (exception& exc)  {
      cerr << exc.what() << "\n";

  }  catch (...)  {
      cerr << "Unknown exception" << "\n";
  }
}
```

Βιβλιογραφία

[1] "ECL^{i}PS^e constraint programming system," διαθέσιμο στη διεύθυνση http://www.eclipse-clp.org/, 2008.

[2] K. Apt and M. Wallace, *Constraint Logic Programming using ECL^{i}PS^e*. Cambridge University Press, 2007.

[3] R. Backofen and D. Gilbert, "Bioinformatics and constraints," *Constraints*, vol. 6, no. 2-3, pp. 141–156, 2001.

[4] R. Barták, "Constraint-based scheduling: An introduction for newcomers," *Preprints of the 7th IFAC Workshop on Intelligent Manufacturing Systems*, pp. 6–8, 2003.

[5] H. Batnini, C. Michel, and M. Rueher, "Mind the gaps: A new splitting strategy for consistency techniques," in *Proc. 11th Int. Conf. on Principles and Practices of Constraint Programming (CP'05)*. Springer, 2005, pp. 77–91.

[6] C. Bessière, "Arc-consistency and arc-consistency again," *Artificial Intelligence*, vol. 65, no. 1, pp. 179–190, 1994.

[7] C. Bessière, "Constraint propagation," in *Handbook of Constraint Programming*. Elsevier Science, 2006, pp. 29–83.

[8] C. Bessière, E. Freuder, and J. Régin, "Using constraint metaknowledge to reduce arc consistency computation," *Artificial Intelligence*, vol. 107, no. 1, pp. 125–148, 1999.

[9] C. Bessière, J. C. Régin, R. Yap, and Y. Zhang, "An optimal coarse-grained arc consistency algorithm," *Artificial Intelligence*, vol. 165, no. 2, pp. 165–185, 2005.

[10] C. Bessière and J. Régin, "Refining the basic constraint propagation algorithm," in *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, vol. 17, no. 1, 2001, pp. 309–315.

[11] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[12] K. Chandrasekharan, *Arithmetical Functions*. Springer, 1970, ch. VIII, pp. 194–228.

[13] A. Chmeiss and P. Jégou, "Efficient path-consistency propagation," *Int. Journal on Artificial Intelligence Tools*, vol. 7, no. 2, pp. 121–142, 1998.

[14] L. Di Gaspero, B. McCollum, and A. Schaerf, "The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (Track 3)," Tech. Rep., 2007.

[15] E. Freuder, "In pursuit of the holy grail," *ACM Comput. Surv.*, p. 63, 1997.

[16] C. Gu, P. Purdom, J. Franco, and B. Wah, "Algorithms for satisfiability (SAT) problem: A survey," *Discrete Mathematics and Theoretical Computer Science: Satisfiability*, pp. 378–383, 1997.

[17] L. Hoehn and J. Ridenhour, "Summations involving computer-related functions," *Mathematics Magazine*, vol. 62, no. 3, pp. 191–196, 1989.

[18] H. Hoos and E. Tsang, "Local search methods," in *Handbook of Constraint Programming*. Elsevier Science, 2006, pp. 135–167.

[19] M. N. Huxley, *The Distribution of Prime Numbers*. Oxford University Press, 1972, p. 7.

[20] *ILOG Solver 4.4: Reference Manual*, ILOG S.A., 1999.

[21] *ILOG Solver 4.4: User's Manual*, ILOG S.A., 1999.

[22] A. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[23] C. Lecoutre, F. Boussemart, and F. Hemery, "Exploiting multidirectionality in coarse-grained arc consistency algorithms," in *Proc. 9th Int. Conf. on Principles and Practices of Constraint Programming (CP'03)*, 2003, pp. 480–494.

[24] A. Lesk, *Introduction to Bioinformatics*. Oxford University Press, 2002.

[25] A. Mackworth, "Consistency in networks of relations," Dept. of Computer Science, Univ. of B.C. Vancouver, Tech. Rep. 75-3, 1975.

[26] A. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.

[27] A. Mackworth, "On reading sketch maps," in *Proc. 5th Int. Joint Conf. on Artificial Intelligence (IJCAI'77)*, 1977, pp. 598–606.

[28] J. J. McGregor, "Relational consistency algorithms and their application in finding subgraph and graph isomorphism," *Information Science*, vol. 19, pp. 229–250, 1979.

[29] R. Mohr and T. Henderson, "Arc and path consistency revisited," *Artificial Intelligence*, vol. 28, pp. 225–233, 1986.

[30] H. L. Montgomery, *Ten Lectures on the Interface Between Analytic Number Theory and Harmonic Analysis*. American Mathematical Society, 1996, p. 56.

[31] T. Müller, "ITC2007 solver description: A hybrid approach," in *Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT'08)*, 2008, to appear.

[32] P. Refalo, "Impact-based search strategies for constraint programming," in *Proc. 10th Int. Conf. on Principles and Practices of Constraint Programming (CP'04)*. Springer, 2004, pp. 557–571.

[33] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995, ch. 5, pp. 179–204.

[34] D. Sabin and E. C. Freuder, "Contradicting conventional wisdom in constraint satisfaction," in *Proc. 2nd Int. Workshop on Principles and Practice of Constraint Programming (PPCP'94)*, 1994, pp. 125–129.

[35] C. Schulte and M. Carlsson, "Finite domain constraint programming systems," in *Handbook of Constraint Programming.* Elsevier Science, 2006, pp. 495–526.

[36] D. Subramanian, "Computational genefinding," Connexions website: http://cnx.org/content/m15083/1.1/, 2007.

[37] P. van Hentenryck, Y. Deville, and C. Teng, "A generic arc-consistency algorithm and its specializations," *Artificial Intelligence*, vol. 57, no. 2-3, pp. 291–321, 1992.

[38] T. Walsh, "Depth-bounded discrepancy search," in *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, vol. 15, 1997, pp. 1388–1395.

[39] Y. Zhang and R. Yap, "Making AC-3 an optimal algorithm," in *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, vol. 17, no. 1, 2001, pp. 316–321.

[40] N. Zhou, "Programming finite-domain constraint propagators in action rules," *Theory and Practice of Logic Programming*, vol. 6, no. 5, pp. 483–507, 2006.

[41] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Ι. Ρεφανίδης, Φ. Κόκκορας και Η. Σακελλαρίου, *Τεχνητή Νοημοσύνη.* Εκδόσεις Γαρταγάνη, 2002, pp. 68–82.

[42] Ν. Ποθητός, Naxos Solver*: Εγχειρίδιο Χρήσης*, διαθέσιμο στη διεύθυνση http://www.di.uoa.gr/~grad0780/naxos/naxos.pdf, 2008.

Ευρετήριο

AC, *βλέπε* arc consistency
AC-1, 31
AC-3, 31
AC-5, 37
AC-5+, 55
CP, *βλέπε* constraint programming
CSP, *βλέπε* constraint satisfaction problem
GAC, *βλέπε* generalized arc consistency
GAC-5, 44
ITC-2007, *βλέπε* International Timetabling Competition 2007
International Timetabling Competition 2007, 83
MAC, *βλέπε* maintaining arc consistency
NCSP, *βλέπε* numerical constraint satisfaction problem
arc consistency, 30
bounds consistency, 33
branch-and-bound, 58
coarse-grained, 36
constraint programming, 21
constraint satisfaction problem, 21
domain splitting, 69
event, 48
fine-grained, 36
generalized arc consistency, 33
maintaining arc consistency, 69
numerical constraint satisfaction problem, 69
propagator, 48
succeed-first heuristic, 70
value ordering heuristic, 70
variable ordering heuristic, 76

αδρής υφής αλγόριθμοι, 36
αναζήτηση, 24
ανάθεση, 22
ανώτερης τάξης περιορισμός, 22
αριθμητικό πρόβλημα ικανοποίησης περιορισμών, 69

γενικευμένη συνέπεια ακμών, 33
γέννα-και-δοκίμαζε, 25

δεσμευμένη, 22
διάδοση περιορισμών, 29
διακλάδωσε-και-φράξε, 58
διατήρηση συνέπειας ακμών, 69
δίκαιη διχοτόμηση πεδίων τιμών, 71
δίκτυο περιορισμών, 29
διχοτόμηση πεδίων τιμών, 69
δυαδικός περιορισμός, 22

ευριστικό επιλογής μεταβλητής, 76
ευριστικό επιλογής τιμής, 70

καθολικός περιορισμός, 22
κρυπτάριθμος, 23

λεπτής υφής αλγόριθμοι, 36
λύση, 22

μεταβλητή, *βλέπε* περιορισμένη μεταβλητή
μοναδιαίος περιορισμός, 22

οπισθοδρόμηση, 27

πεδίο τιμών, 22
περιορισμένη μεταβλητή, 22
περιορισμός, 22
ΠΙΠ, *βλέπε* πρόβλημα ικανοποίησης περιορισμών
πρόβλημα ικανοποίησης περιορισμών, 21
προγραμματισμός με περιορισμούς, 21

συμβάν, 48
συνάρτηση διάδοσης, 48
συνέπεια ακμών, 30
συνέπεια ορίων, 33

τοπική αναζήτηση, 26

χρωμόσωμα, 65

ωρολόγιο πρόγραμμα, 83