ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Αυτόματη Παραγωγή Παράλληλου SPMD Κώδικα για Μετασχηματισμούς Υπερκόμβων σε Φωλιασμένους Βρόχους

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Γεώργιος Ι. Γκούμας

Αθήνα, Δεκέμβριος 2003

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Αυτόματη Παραγωγή Παράλληλου SPMD Κώδικα για Μετασχηματισμούς Υπερκόμβων σε Φωλιασμένους Βρόχους

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

του

Γεωργίου Ι. Γκούμα

Διπλωματούχου Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών Ε.Μ.Π. (1999)

Συμβουλευτική Επιτροπή: Ν. Κοζύρης
Γ. Παπακωνσταντίνου
Π. Τσανάκας

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 23η Δεκεμβρίου 2003.

.....	.....	.....
Ν. Κοζύρης	Γ. Παπακωνσταντίνου	Π. Τσανάκας
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.
.....	.....	.....
Τ. Σελλής	Α. Σταφυλοπάτης	Ε. Ζάχος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

.....
Ε. Παπαδρακάκης
Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2003

.....

Γεώργιος Ι. Γκούμας
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Γεώργιος Ι. Γκούμας, 2003
Με επιφύλαξη παντός δικαιώματος – All rights reserved

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς το συγγραφέα.
Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

στους γονείς μου
Γιάννη και Ντίνα
στον αδελφό μου Μάνο

στο Μίλτο

Περίληψη

Ο μετασχηματισμός υπερκόμβων είναι ένας από τους σημαντικότερους μετασχηματισμούς φωλιασμένων βρόχων, που εφαρμόζεται για να μειωθεί το κόστος επικοινωνίας και συγχρονισμού σε αρχιτεκτονικές παράλληλης επεξεργασίας. Ο μετασχηματισμός αυτός ομαδοποιεί κοντινές επαναλήψεις στο χώρο επαναλήψεων του φωλιασμένου βρόχου σχηματίζοντας υπερκόμβους, οι οποίοι ανατίθενται σε επεξεργαστές, εκτελούνται αδιαίρετα ενώ η επικοινωνία μεταξύ των επεξεργαστών λαμβάνει χώρα πριν και μετά την εκτέλεση των υπολογισμών στο εσωτερικό των υπερκόμβων. Με τον τρόπο αυτό ελαττώνεται η συχνότητα επικοινωνίας μεταξύ των επεξεργαστών και επομένως μειώνεται η επιβάρυνση που οφείλεται στο χρόνο εκκίνησης (startup latency) κατά την αποστολή και λήψη δεδομένων.

Ένας πολύ μεγάλος αριθμός ερευνητικών εργασιών ασχολείται με τις ιδιότητες του μετασχηματισμού υπερκόμβων. Αυτό που ενδιαφέρει κυρίως, είναι η εύρεση ενός κατάλληλου μετασχηματισμού που θα μειώνει περαιτέρω τον όγκο της επικοινωνίας ή το συνολικό χρόνο εκτέλεσης του αλγορίθμου, μέσω της μείωσης του άεργου χρόνου των επεξεργαστών. Για το σκοπό αυτό έχουν προταθεί στη βιβλιογραφία διάφορες μέθοδοι επιλογής του μεγέθους και του σχήματος των υπερκόμβων, ανάλογα με τις εξαρτήσεις του αλγορίθμου και το σχήμα του χώρου επαναλήψεων. Έχει ήδη αποδειχθεί θεωρητικά, ότι σε πολλές περιπτώσεις το βέλτιστο σχήμα του μετασχηματισμού υπερκόμβων δεν είναι τετραγωνικό ή ορθογώνιο, αλλά ένα γενικό παραλληλόγραμμο το οποίο προσδιορίζεται με βάση τις εξαρτήσεις του αρχικού προβλήματος και τα όρια του χώρου επαναλήψεων.

Παρά το γεγονός ότι θεωρητικά έχει αναδειχθεί η επίδραση του σχήματος στο συνολικό χρόνο εκτέλεσης ενός φωλιασμένου βρόχου που έχει μετασχηματιστεί με το μετασχηματισμό υπερκόμβων και πρόκειται να εκτελεστεί σε κάποια παράλληλη αρχιτεκτονική, δεν υπάρχει μέθοδος παραγωγής κώδικα για μη-ορθογώνιους μετασχηματισμούς υπερκόμβων και επομένως δεν υπάρχουν υλοποιήσεις των σχετικών μεθόδων επιλογής του βέλτιστου σχήματος μετασχη-

ματισμού υπερκόμβων. Η παρούσα εργασία παρουσιάζει μια πλήρη μέθοδο αυτόματης παραγωγής παράλληλου SPMD (Single Program Multiple Data) κώδικα για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Η μέθοδος εφαρμόζεται για οποιονδήποτε μετασχηματισμό υπερκόμβων, δεν περιορίζεται δηλαδή μόνο στους ορθογώνιους, όπως συνέβαινε μέχρι στιγμής.

Το πρόβλημα της παραγωγής κώδικα για μετασχηματισμένους φωλιασμένους βρόχους χωρίζεται σε δύο μεγάλα υποπροβλήματα. Το πρώτο είναι η παραγωγή σειριακού μετασχηματισμένου κώδικα. Η προτεινόμενη μέθοδος επιτυγχάνει να γεννήσει αποδοτικότερο κώδικα σε σχέση με τις έως τώρα μεθόδους και μάλιστα με σημαντικά μικρότερο χρόνο μεταγλώττισης. Το δεύτερο υποπρόβλημα είναι η παραλληλοποίηση του σειριακού μετασχηματισμένου κώδικα, για το οποίο δεν έχει δοθεί σχετική μέθοδος στη βιβλιογραφία. Στην παρούσα εργασία παρουσιάζουμε μία μέθοδο που παραλληλοποιεί το σειριακό μετασχηματισμένο κώδικα και παράγει παράλληλο κώδικα για αρχιτεκτονικές κατανεμημένης μνήμης.

Η προτεινόμενη μέθοδος παραγωγής παράλληλου κώδικα εφαρμόστηκε σε ένα εργαλείο αυτόματης παραλληλοποίησης. Με τη βοήθεια του εργαλείου αυτού πραγματοποιήθηκαν μετρήσεις για να ελεγχθεί η αποδοτικότητα του παραγόμενου κώδικα, και συγκρίσεις του συνολικού χρόνου εκτέλεσης μεταξύ διαφόρων σχημάτων μετασχηματισμών ομαδοποίησης. Οι μετρήσεις αυτές επιβεβαίωσαν την αποδοτικότητα του παραγόμενου κώδικα αλλά και την θεωρία περί κατάλληλης επιλογής του βέλτιστου σχήματος του μετασχηματισμού υπερκόμβων.

Λέξεις Κλειδιά: Φωλιασμένοι Βρόχοι, Μετασχηματισμός Υπερκόμβων, Αυτόματη Παραγωγή Κώδικα, SPMD, Μέθοδος Απαλοιφής Fourier-Motzkin, Αρχιτεκτονικές Κατανεμημένης Μνήμης, MPI.

Abstract

Tiling (or supernode) transformation is among the most important loop transformations, used to enforce coarse-grain parallelism in parallel architectures by reducing the frequency of communication. Tiling transformation groups neighbouring iterations of the iteration space to form tiles, which are assigned to processors, executed uninterruptedly, whereas communication occurs before and after the computation within the iterations of a tile. In this way, communication frequency is reduced and so is the total execution time, due to the alleviation of the program from the significant overhead of frequent communication startup latencies.

Tiling transformation properties have been extensively discussed in literature. Finding a proper tiling transformation which will further reduce communication volume or total execution time is the main concern of a large number of papers in the field. To achieve this goal, authors have proposed methods to select the volume and the shape of the tiling transformation to be applied, in order to reduce communication volume per tile and processor idle times. This tile selection is based on the program dependencies and the iteration space shape. It has been proven theoretically, that the optimal supernode shape in several cases is not rectangular or orthogonal, but a general parallelogram which arises from the algorithm's dependencies and the bounds of the iteration space.

Despite the fact that there has been a lot of discussion concerning the selection of a non-rectangular tiling transformation for loops to be executed on a parallel architecture, there are no code generation methods for non-rectangular transformations, and thus the results of the relevant theory have not been evaluated in practice. In this dissertation we present a complete end-to-end method to automatically generate parallel SPMD (Single Program Multiple Data) code for nested loops transformed by a general tiling transformation.

The problem of SPMD code generation is partitioned into the subproblems of sequential

tiled code generation and parallelization. The proposed method succeeds in generating much more efficient sequential tiled code than the already proposed approach and in addition reduces the compilation time. As far as the parallelization of non-rectangularly tiled iteration spaces is concerned, there is no other method proposed in bibliography. In this dissertation we present a new method which parallelizes the sequential tiled code and generates efficient parallel code for distributed memory architectures.

The proposed method was applied in a software tool which automatically generates SPMD parallel code for distributed memory machines. With the aid of this tool we performed a large number of measurements in order to evaluate the efficiency of the generated code and to compare the total execution times of various shapes in tiling transformations. These measurements confirmed previous theoretical work concerning the selection of an optimal shape in tiling transformations.

Keywords: Nested Loops, Tiling/Supernode Transformation, Automatic Code Generation, SPMD, Fourier-Motzkin Elimination Method, Distributed Memory Architectures, MPI.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Αντί Προλόγου

Η διδακτορική αυτή διατριβή εκπονήθηκε στον τομέα Τεχνολογίας Πληροφορικής και Υπολογιστών, της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, του Εθνικού Μετσόβιου Πολυτεχνείου. Περιλαμβάνει την έρευνα και τα αποτελέσματα που προέκυψαν κατά τη διάρκεια των μεταπτυχιακών μου σπουδών στη Σχολή Ηλεκτρολόγων Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου και συγκεκριμένα στο Εργαστήριο Υπολογιστικών Συστημάτων.

Μέσα από αυτό το κείμενο θα ήθελα να εκφράσω τις ευχαριστίες μου σε ένα πλήθος ανθρώπων που συνέβαλαν, ο καθένας με τον τρόπο του, στην ολοκλήρωση της εργασίας αυτής. Πρώτον απ' όλους θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, επίκουρο καθηγητή Νεκτάριο Κοζύρη για την σημαντικότατη επιστημονική καθοδήγηση καθ' όλη τη διάρκεια της έρευνάς μου, αλλά και για την πίστη του στην προσπάθειά μου την οποία εξεδήλωνε κυρίως στις πιο δύσκολες στιγμές. Ιδιαίτερα θερμές είναι οι ευχαριστίες μου και προς τον καθηγητή Γεώργιο Παπακωνσταντίνου, η παρουσία του οποίου υπήρξε ηθική και επιστημονική έμπνευση για μένα, από τον πρώτον καιρό της παρουσίας μου στο Εργαστήριο Υπολογιστικών Συστημάτων ως προπτυχιακός φοιτητής. Θα ήθελα επίσης να ευχαριστήσω το τρίτο μέλος της Συμβουλευτικής Επιτροπής μου, τον καθηγητή Παναγιώτη Τσανάκα για την προθυμία του να συνδράμει με οποιονδήποτε τρόπο στην ολοκλήρωση της διατριβής αυτής.

Ιδιαίτερη αναφορά θα ήθελα επίσης να κάνω και στους συναδέλφους υποψήφιους διδάκτορες Μαρία Αθανασάκη και Νίκο Δροσινό. Η συνεισφορά τους στην παρούσα εργασία ήταν καθοριστική σε όλους τους τομείς και η βοήθειά τους ήταν ανεκτίμητη ακόμα και στα πλέον δύσκολα ερευνητικά σημεία. Είναι δε βέβαιο, ότι χωρίς τη συμβολή τους στην υλοποίηση των προτεινόμενων μεθοδολογιών, δεν θα ήταν δυνατή η ολοκλήρωση της παρούσας εργασίας, παρά μόνο ύστερα από μεγάλο χρονικό διάστημα. Στο πρόσωπο της Μαρίας και του Νίκου δεν βρήκα μόνο δύο πολύτιμους συνεργάτες, αλλά δύο ξεχωριστούς ανθρώπους, δύο πραγματικούς

φίλους. Παιδιά σας ευχαριστώ μέσα απ' την καρδιά μου.

Θα ήταν παράλειψη βέβαια να μην αναφερθώ σε όλα τα μέλη του Εργαστηρίου Υπολογιστικών Συστημάτων για το άψογο κλίμα συνεργασίας που έχουν αναπτύξει. Το εργαστήριο αποτελεί έναν αξιοθαύμαστο χώρο ανταλλαγής επιστημονικής γνώσης υψηλού επιπέδου, και ευρισκόμενος στο χώρο αυτό είχα τη δυνατότητα να συμμετάσχω σε αναρίθμητες συζητήσεις που μου άνοιξαν πολλούς ορίζοντες κατά τη διάρκεια της επιστημονικής μου έρευνας. Γι αυτό το ιδανικό περιβάλλον εργασίας αξίζουν συγχαρητηρίων τα μέλη ΔΕΠ του εργαστηρίου Γ. Παπακωνσταντίνου, Π. Τσανάκας και Ν. Κοζύρης αλλά και όλοι οι μεταπτυχιακοί και προπτυχιακοί φοιτητές που περνούν ένα σημαντικό μέρος της ζωής τους στο χώρο αυτό. Προσωπικά θα ήθελα να ευχαριστήσω ιδιαίτερα το διδάκτορα Ιωάννη Δροσίτη, ο οποίος με βοήθησε στα πρώτα βήματά μου στο εργαστήριο αλλά και στη συνέχεια της έρευνάς μου. Ιδιαίτερα σημαντική ήταν και η συνεργασία μου με το συνάδελφο και φίλο Άρη Σωτηρόπουλο, από τον οποίο απεκόμισα πολύ σημαντικές γνώσεις. Τέλος, θα ήθελα να ευχαριστήσω και τους νεότερους συναδέλφους του εργαστηρίου Βάλια Αθανασάκη, Βαγγέλη Κούκη, Αντώνη Χαζάπη, Αντώνη Ζήσιμο, Γιώργο Τσουκαλά και Κορνήλιο Κούρτη για το άψογο κλίμα συνεργασίας που καλλιεργούν στο εργαστήριο και κυρίως για τη διάθεσή τους να βοηθήσουν χωρίς καμία προσμονή ανταπόδοσης. Εύχομαι σε όλους το αποτέλεσμα των ερευνών τους να είναι τόσο υψηλό όσο η μεγάλη αξία τους.

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπιμότητα

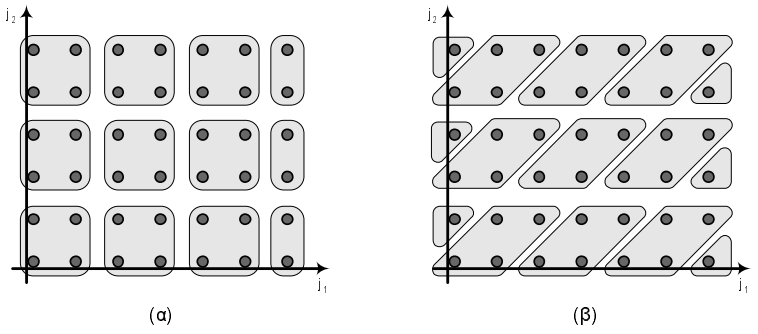
Ο μετασχηματισμός υπερκόμβων (tiling ή supernode transformation) είναι μία τεχνική που χρησιμοποιείται για την εκμετάλλευση της ιεραρχίας μνήμης σε μονοεπεξεργαστικά και πολυεπεξεργαστικά συστήματα και για τη μείωση του κόστους συγχρονισμού και επικοινωνίας σε παράλληλες αρχιτεκτονικές. Η παρούσα διατριβή παρουσιάζει μία νέα μέθοδο αυτόματης παραγωγής παράλληλου κώδικα χρησιμοποιώντας το προγραμματιστικό μοντέλο Single Program Multiple Data (SPMD) για φωλιασμένους βρόχους (nested loops) που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων και πρόκειται να εκτελεστούν σε αρχιτεκτονικές κατανεμημένης μνήμης.

Το μεγαλύτερο πρόβλημα κατά την εκτέλεση ενός φωλιασμένου βρόχου σε μία παράλληλη αρχιτεκτονική, είναι το κόστος συγχρονισμού όταν η εκτέλεση γίνεται σε συστήματα με μοιραζόμενη μνήμη (shared memory) και το κόστος επικοινωνίας (ανταλλαγής δεδομένων) όταν η εκτέλεση γίνεται σε συστήματα με κατανεμημένη μνήμη (distributed memory). Και στις δύο αυτές περιπτώσεις, ο μετασχηματισμός υπερκόμβων μπορεί να επιφέρει δραστική μείωση στο χρόνο εκτέλεσης ενός φωλιασμένου βρόχου, με τη μείωση της συχνότητας συγχρονισμού και επικοινωνίας αντίστοιχα. Ο μετασχηματισμός αυτός ομαδοποιεί κοντινές επαναλήψεις του φωλιασμένου βρόχου δημιουργώντας ομάδες (υπερκόμβους) επαναλήψεων, εκτελεί τις ομαδοποιημένες αυτές επαναλήψεις αδιαίρετα και πραγματοποιεί συγχρονισμό/επικοινωνία για σύνολα

δεδομένων (και όχι για μεμονωμένα δεδομένα), πριν και μετά την εκτέλεση των υπολογισμών στο εσωτερικό ενός υπερκόμβου.

Ο μετασχηματισμός υπερκόμβων προτάθηκε από τους Irigoin και Triolet [IT88] και μελετήθηκε σε πληθώρα εργασιών [Wol89], [WL91a], [AI91], [LRW91], [RS92], [TZ94], [BDRR94], [CR95], [CDR97], [Xue97a], [Xue97b], [HCF97], [DDR97], [HS98], [BDRV99], [CM99], [HCF99], [Jim99], [KPCM99], [TX00], [ACN+00], [GSK01], [LLL01], [STK01], [XW02], [STK02], [HS02]. Ένας μεγάλος αριθμός από τις εργασίες αυτές αναζητά το βέλτιστο από πλευράς επικοινωνίας μετασχηματισμό υπερκόμβων [RS92], [BDRR94], [Xue97a], το βέλτιστο μέγεθος υπερκόμβου [OSKO95], [HS98], [CM99], [ACN+00] και το βέλτιστο σχήμα υπερκόμβου για αποδοτικότερη δρομολόγηση των υπερκόμβων σε επεξεργαστές [HCF97], [DDR97], [HCF99], [XW02], [HS02]. Ένα πολύ ενδιαφέρον συμπέρασμα, που προκύπτει από την παραπάνω έρευνα, είναι ότι πολύ συχνά το βέλτιστο σχήμα του μετασχηματισμού υπερκόμβων δεν είναι τετραγωνικό ή ορθογώνιο παραλληλόγραμμο (ή τα ανάλογά τους σε μεγαλύτερες διαστάσεις), αλλά ένα γενικό n-διάστατο παραλληλόγραμμο. Στο Σχήμα 1.1 φαίνεται η απεικόνιση ενός δισδιάστατου φωλιασμένου βρόχου στο επίπεδο και δύο διαφορετικοί μετασχηματισμοί υπερκόμβων: ένας ορθογώνιος και ένας μη-ορθογώνιος. Για την περίπτωση των ορθογώνιων μετασχηματισμών είναι μάλλον εμφανές, ότι μπορεί εύκολα να γραφεί παράλληλος κώδικας που να υλοποιεί τον αντίστοιχο μετασχηματισμό υπερκόμβων. Στους μη-ορθογώνιους μετασχηματισμούς όμως, η παραγωγή κώδικα είναι πολύ δύσκολη ή ακόμα και αδύνατη για τον προγραμματιστή, ακριβώς λόγω του σχήματος των υπερκόμβων. Η υλοποίηση των μη-ορθογώνιων μετασχηματισμών υπερκόμβων μπορεί να γίνει μόνο από κάποιον αυτόματο παραλληλοποιητή-μεταγλωττιστή, που θα ακολουθεί μια αυστηρά καθορισμένη διαδικασία για την παραγωγή του παράλληλου κώδικα. Για την περίπτωση αυτή, δεν υπάρχει στη βιβλιογραφία μέθοδος παραγωγής παράλληλου κώδικα, που να εκτελεί παραλληλόγραμμους υπερκόμβους σε αρχιτεκτονικές κατανεμημένης μνήμης.

Η παρούσα εργασία έρχεται να καλύψει το παραπάνω κενό στη βιβλιογραφία, προτείνοντας μία μέθοδο παραγωγής SPMD κώδικα για γενικούς παραλληλεπίπεδους μετασχηματισμούς υπερκόμβων. Ιδιαίτερη έμφαση δίνεται στη διατήρηση της πολυπλοκότητας της μεθόδου σε χαμηλά επίπεδα, καθώς αν τα σχήματα των υπερκόμβων και του χώρου επαναλήψεων του φωλιασμένου βρόχου είναι μη-ορθογώνια, τότε η διαδικασία μπορεί να είναι ιδιαίτερα χρονοβόρα, έως μη πρακτική (π.χ. ο χρόνος μεταγλώττισης μπορεί να είναι της τάξης των ωρών ή ακόμα και ημερών). Ακόμα όμως μεγαλύτερη προσοχή δίνεται στην αποδοτικότητα του παραγόμενου κώδικα. Είναι δυνατόν ένας μη-ορθογώνιος μετασχηματισμός υπερκόμβων να είναι θεωρητικά καλύτερος από έναν τετραγωνικό, αλλά το κόστος σε χρόνο εκτέλεσης για τη διάσχιση

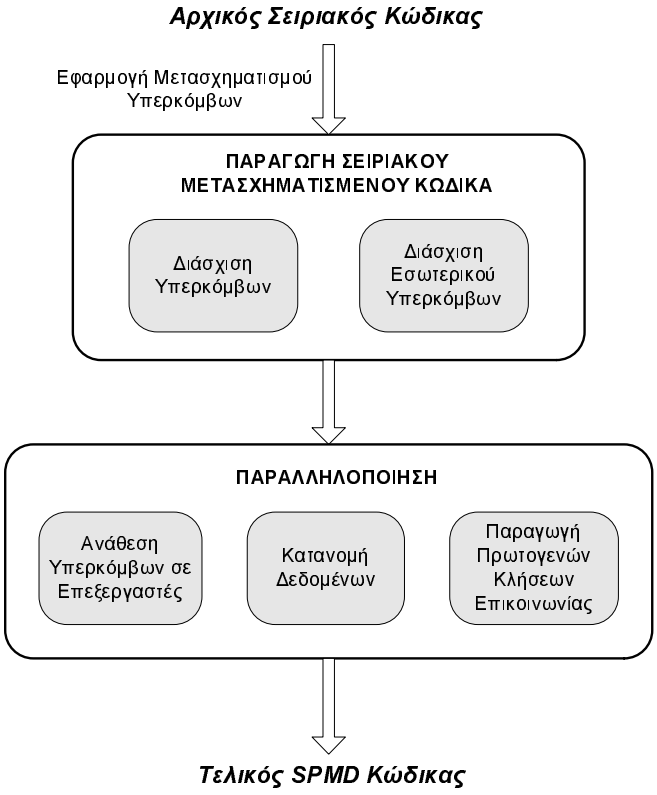


Σχήμα 1.1: Μετασχηματισμοί υπερκόμβων: (α) Ορθογώνιος (β) Μη-Ορθογώνιος

μη-τετραγωνικών υπερκόμβων να υπερκαλύπτει τα θεωρητικά οφέλη. Έτσι, στην πράξη ένας ορθογώνιος μετασχηματισμός που υλοποιείται πιο αποδοτικά, είναι δυνατόν να οδηγήσει σε μικρότερους χρόνους εκτέλεσης, παρ' όλο που αυτό δεν επαληθεύεται θεωρητικά. Η μέθοδος που παρουσιάζεται στην παρούσα διατριβή, λαμβάνει υπ' όψη τις παραμέτρους που αναφέρθηκαν και παράγει αποδοτικό κώδικα που αναδεικνύει τα πλεονεκτήματα των μη-ορθογώνιων μετασχηματισμών κρατώντας ταυτόχρονα το χρόνο μεταγλώττισης σε απολύτως αποδεκτά επίπεδα.

Η διαδικασία παραγωγής SPMD κώδικα για φωλιασμένους βρόχους, που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων, χωρίζεται σε δύο μεγάλα μέρη: στην παραγωγή του *σειριακού μετασχηματισμένου κώδικα* και στην *παραλληλοποίηση* του κώδικα αυτού. Ο σειριακός μετασχηματισμένος κώδικας είναι ένας ενδιάμεσος κώδικας, χωρίς πρακτική αξία, που αναδιατάσσει τη σειρά εκτέλεσης του αρχικού φωλιασμένου βρόχου με βάση το μετασχηματισμό υπερκόμβων. Η διαδικασία παραγωγής του σειριακού μετασχηματισμένου κώδικα χωρίζεται επίσης σε δύο υποπροβλήματα, αυτό της διάσχισης των υπερκόμβων και αυτό της διάσχισης των σημείων στο εσωτερικό κάθε υπερκόμβου. Από την άλλη, η φάση της παραλληλοποίησης που ακολουθεί, αναλαμβάνει την ανάθεση των υπερκόμβων σε επεξεργαστές, την κατανομή των δεδομένων και την παραγωγή των κατάλληλων πρωτογενών κλήσεων ανταλλαγής μηνυμάτων (Σχήμα 1.2).

Η μέθοδος που παρουσιάζεται στην παρούσα εργασία, αντιμετωπίζει ξεχωριστά κάθε ένα από τα πέντε παραπάνω υποπροβλήματα (διάσχιση υπερκόμβων, διάσχιση εσωτερικού υπερκόμβων, ανάθεση υπερκόμβων σε επεξεργαστές, κατανομή δεδομένων, παραγωγή πρωτογενών κλήσεων επικοινωνίας), με κύριο στόχο την παραγωγή αποδοτικού SPMD κώδικα που θα μειώνει το συνολικό χρόνο εκτέλεσης, διατηρώντας παράλληλα μικρό το χρόνο μεταγλώττισης. Πειραματικά αποτελέσματα αποδεικνύουν την αποδοτικότητα της μεθόδου, τόσο σε σχέση με το



Σχήμα 1.2: Διαχωρισμός της παραγωγής SPMD κώδικα σε επιμέρους υποπροβλήματα

χρόνο μεταγλώττισης, όσο και σε σχέση με την ποιότητα του παραγόμενου κώδικα. Έτσι, ενσωματώνοντας τη συγκεκριμένη μέθοδο σε έναν αυτόματο παραλληλοποιητή-μεταγλωττιστή (automatic parallelizing compiler), είμαστε σε θέση να αναδείξουμε τα θεωρητικά πλεονεκτήματα ενός μετασχηματισμού υπερκόμβων, όσο περίπλοκο και αν είναι το σχήμα του ή το σχήμα του χώρου επαναλήψεων.

Επειδή η παρούσα εργασία πραγματεύεται την παραγωγή κώδικα που θα εκτελεστεί σε αρχιτεκτονικές κατανεμημένης μνήμης, η συζήτηση και οι σχετικές αναλύσεις θα εστιάσουν σε αυτή την κατηγορία συστημάτων. Είναι αναγκαίο να τονιστεί βέβαια, ότι ο κώδικας σ' αυτήν την περίπτωση είναι πιο σύνθετος, καθώς η μη ύπαρξη ενός κοινού χώρου διευθύνσεων μεταξύ των επεξεργαστών τους υποχρεώνει να δεσμεύουν τοπικούς χώρους μνήμης και να ανταλλάσσουν δεδομένα με αυστηρά καθορισμένες κλήσεις αποστολής και λήψης δεδομένων. Αντίθετα, στα συστήματα μοιραζόμενης μνήμης τα δεδομένα μπορούν να προσπελαστούν από όλους τους επεξεργαστές μέσω της μοιραζόμενης μνήμης, και το μόνο που χρειάζεται είναι η εφαρμογή

κατάλληλων κλήσεων συγχρονισμού (semaphores, locks, barriers κλπ). Γενικά, ο κώδικας για συστήματα μοιραζόμενης μνήμης μπορεί να προκύψει από τον αντίστοιχο κώδικα για συστήματα κατανεμημένης μνήμης, με αντικατάσταση των κλήσεων αποστολής/λήψης δεδομένων με εγγραφές/αναγνώσεις στη μνήμη και εισαγωγή κατάλληλου συγχρονισμού.

1.2 Μελέτη της Θεματικής Περιοχής

Ήδη από τα τέλη της δεκαετίας του '70 με την εμφάνιση των επεξεργαστών RISC, έγινε σαφές ότι ο ρόλος του μεταγλωττιστή (compiler) στην επίδοση ενός υπολογιστικού συστήματος θα ήταν ζωτικής σημασίας [HP95] (σελ. 89-96). Έτσι, παραδοσιακά ο μεταγλωττιστής ανέλαβε την ανάδειξη του παραλληλισμού στο επίπεδο της εντολής (Instruction Level Parallelism - ILP) για την καλύτερη αξιοποίηση του pipeline των επεξεργαστών RISC. Με την εξέλιξη των χαρακτηριστικών των υπολογιστικών συστημάτων, οι μεταγλωττιστές κλήθηκαν να αξιοποιήσουν τις νέες δυνατότητες του υλικού. Για παράδειγμα, στην περίπτωση των διανυσματικών επεξεργαστών (vector processors), ο μεταγλωττιστής αναλαμβάνει να μεταφέρει τον παραλληλισμό στο εσωτερικότερο σημείο ενός φωλιασμένου βρόχου, έτσι ώστε οι παράλληλες και ανεξάρτητες πράξεις στα δεδομένα ενός πίνακα, να μετασχηματιστούν σε μία διανυσματική εντολή. Ανάλογες τεχνικές χρησιμοποιούνται και για τη βέλτιστη απόδοση multiple-issue superscalar και VLIW (Very Large Instruction Word) επεξεργαστών. Η μελέτη τέτοιων τεχνικών, που βασίζονται σε πλήρη γνώση των αρχιτεκτονικών χαρακτηριστικών του συστήματος, καθιστά προφανές, ότι δεν είναι δυνατόν να εφαρμοστούν από τον προγραμματιστή, όσο έμπειρος και αν είναι αυτός, αλλά αντίθετα πρέπει να ανατεθούν στο μεταγλωττιστή.

Μεγάλο ερευνητικό ενδιαφέρον στο πεδίο των μεταγλωττιστών που πραγματοποιούν βελτιστοποιήσεις (optimizing compilers), έχει προκαλέσει η αξιοποίηση της ιεραρχίας μνήμης σε μονο-επεξεργαστικά ή πολυ-επεξεργαστικά συστήματα. Πληθώρα εργασιών ασχολείται με την εύρεση κατάλληλων μετασχηματισμών φωλιασμένων βρόχων, που αναδιατάσσουν την αρχική σειρά εκτέλεσης, με σκοπό οι τυχόν επαναχρησιμοποιήσεις δεδομένων (data reuse) να βρίσκουν τα δεδομένα αυτά στη λανθάνουσα μνήμη (cache memory) [WL91a], [KM92], [LP92], [CMT94], [MCT96]. Μία άλλη προσέγγιση μελετά μετασχηματισμούς δεδομένων, ή διαφορετικούς τρόπους απεικόνισης πινάκων δεδομένων στη μνήμη [KCS+98], [KCRB99], ενώ μία τρίτη ενοποιεί τους μετασχηματισμούς ελέγχου (φωλιασμένων βρόχων) με τους μετασχηματισμούς δεδομένων [CL95], [KRC97], [KRC99]. Στην περίπτωση των συστημάτων μοιραζόμενης μνήμης, οι ερευνητές έχουν επιπρόσθετα να αντιμετωπίσουν το πρόβλημα του false sharing, το

οποίο εμφανίζεται όταν π.χ. δύο επεξεργαστές προσπελαύνουν διαφορετικές λέξεις (words) που όμως ανήκουν στην ίδια γραμμή λανθάνουσας μνήμης (cache line). Σ' αυτή την περίπτωση, διαδοχικές αναγνώσεις και εγγραφές από τους δύο επεξεργαστές θα προκαλέσουν άχρηστες μετακινήσεις της ίδιας γραμμής λανθάνουσας μνήμης από τον ένα στον άλλο, προκαλώντας επιζήμια κίνηση στο διάδρομο του συστήματος. Επειδή το πρόβλημα του false sharing έχει αποδειχθεί ότι μπορεί να προκαλέσει δραματική μείωση στην απόδοση ενός συστήματος, οι ερευνητές έχουν προτείνει τεχνικές μεταγλώττισης που μειώνουν τις συνέπειές του [TLH94] [KCRB99]. Γενικά, η έρευνα γύρω από την αξιοποίηση της ιεραρχίας μνήμης μέσα από την τοπικότητα των αναφορών, μπορεί να θεωρηθεί ιδιαίτερα ώριμη, κάτι που φαίνεται και από το γεγονός ότι μεγάλο μέρος από τις παραπάνω τεχνικές έχει ενσωματωθεί σε ερευνητικούς και εμπορικούς μεταγλωττιστές.

Στη δεκαετία του '90, πολύ μεγάλη έμφαση δόθηκε στην αυτόματη παραλληλοποίηση σειριακών προγραμμάτων. Το όραμα της εποχής ήταν να μπορεί ένας παράλληλος μεταγλωττιστής να μεταφράσει οποιοδήποτε σειριακό πρόγραμμα για εκτέλεση σε μια παράλληλη αρχιτεκτονική, χωρίς καμία εμπλοκή του προγραμματιστή. Κάτι τέτοιο δεν κατέστη απολύτως εφικτό, ιδίως για τις αρχιτεκτονικές κατανεμημένης μνήμης και έτσι σήμερα πιστεύεται ότι κάποιο ποσοστό αλληλεπίδρασης με τον προγραμματιστή θα είναι πάντα απαραίτητο. Παρ' όλα αυτά, ένας μεγάλος όγκος σχετικών ερευνητικών εργασιών μπορεί να αξιοποιηθεί για τη βέλτιστη απεικόνιση αλγορίθμων σε παράλληλες αρχιτεκτονικές. Από τις βασικές προσπάθειες στην αυτόματη παραλληλοποίηση είναι η εξαγωγή του εγγενούς παραλληλισμού ενός φωλιασμένου βρόχου. Αξίζει να σημειωθεί ότι οι σχετικές εργασίες σχεδόν μονοπωλούνται από έρευνα σχετική με τους φωλιασμένους βρόχους, καθώς αυτοί αποτελούν τη βασικότερη πηγή καθυστέρησης στα ακολουθιακά προγράμματα. Η ανάλυση των εξαρτήσεων (dependence analysis) μπορεί να δώσει απάντηση στο αν ένας φωλιασμένος βρόχος περιλαμβάνει απολύτως ανεξάρτητες και επομένως παράλληλες περιοχές [Ban88], [Pug92]. Ένας μεγάλος αριθμός εργασιών πραγματεύεται την ανάδειξη `DOALL` φωλιασμένων βρόχων [WL91b], [D'H92], [SF92], δηλαδή βρόχων των οποίων τα στιγμιότυπα είναι δυνατόν να εκτελεστούν παράλληλα χωρίς ανάγκη για ενδιάμεσο συγχρονισμό. Ένα άλλο σημαντικό πρόβλημα είναι η βέλτιστη απεικόνιση και δρομολόγηση τέτοιων βρόχων σε αρχιτεκτονικές μοιραζόμενης μνήμης [ML92] [TN93] [HP96].

Ιδιαίτερα μεγάλη ερευνητική προσπάθεια έχει προκαλέσει η υλοποίηση παράλληλων μεταγλωττιστών προγραμμάτων FORTRAN [FHK+91], [SLR+95], [CMZ92], [AMC97] για αρχιτεκτονικές κατανεμημένης μνήμης. Εδώ ο προγραμματιστής καλείται να ορίσει τον τρόπο με τον οποίο διαμοιράζονται τα δεδομένα στις μνήμες των κόμβων του συστήματος (ανά μπλοκ, κυκλικά ή μπλοκ-κυκλικά - block, cyclic, block-cyclic) και ο μεταγλωττιστής αναλαμβάνει να

αναθέσει τις επαναλήψεις του φωλιασμένου βρόχου με βάση τον κανόνα "ο κατέχων υπολογίζει" (owner-computes rule). Εδώ η έρευνα εστιάζεται στην ελαχιστοποίηση του κόστους επικοινωνίας με κατάλληλη κατανομή των δεδομένων [RS91], [GB92], [AKN95], ή με τεχνικές όπως η διανυσματική επικοινωνία (message vectorization), η συνένωση μηνυμάτων (message coalescing) ή η εξάλειψη της πλεονάζουσας επικοινωνίας (redundant communication elimination) [KBC+99], [AL93], [GMS+95], [GSS96]. Όλες οι παραπάνω τεχνικές, εφαρμόζονται κυρίως στην περίπτωση που ο αλγόριθμος περιέχει παραλληλισμό που εκφράζεται με τη βοήθεια DOALL φωλιασμένων βρόχων, όπως είναι για παράδειγμα ο πολλαπλασιασμός πινάκων, η ανάλυση LU (LU decomposition) κ.ά.

Υπάρχει όμως μια μεγάλη κατηγορία προβλημάτων, τα οποία δεν περιέχουν παραλληλισμό που να εκφράζεται σαν ένας DOALL φωλιασμένος βρόχος, ούτε και μπορούν να μετασχηματιστούν σε αυτή τη μορφή. Πρόκειται για προγράμματα που περιέχουν n γραμμικά ανεξάρτητα διανύσματα εξάρτησης (n είναι το βάθος του φωλιασμένου βρόχου). Τέτοια προβλήματα προκύπτουν κατά την επίλυση συνήθων και μερικών διαφορικών εξισώσεων, στη χωρική παρεμβολή ή κατά την επίλυση προβλημάτων δυναμικού προγραμματισμού. Η απουσία DOALL βρόχων, δεν αποκλείει τον παραλληλισμό των βρόχων αυτών, καθώς υπάρχουν επαναλήψεις που μπορούν να εκτελεστούν παράλληλα, με τη διαμεσολάβηση όμως συγχρονισμού ή επικοινωνίας. Οι φωλιασμένοι βρόχοι με αυτή την ιδιότητα ονομάζονται DOACROSS βρόχοι [TZ94]. Για την ελαχιστοποίηση του κόστους επικοινωνίας κατά την παράλληλη εκτέλεση τέτοιων προγραμμάτων έχει προταθεί ο μετασχηματισμός υπερκόμβων, που μειώνει τη συχνότητα επικοινωνίας και ομαδοποιεί τις αποστολές και λήψεις σε μεγαλύτερα μηνύματα. Ουσιαστικά, ο μετασχηματισμός υπερκόμβων αποτελεί μονόδρομο στην παράλληλη εκτέλεση DOACROSS βρόχων, διότι σε αντίθετη περίπτωση το κόστος επικοινωνίας είναι τόσο μεγάλο που πιθανότατα η παραλληλοποίηση θα επιφέρει ανεπαίσθητη επιτάχυνση ή ακόμα και επιβράδυνση.

Η παρούσα ερευνητική εργασία παρουσιάζει μία νέα μέθοδο αυτόματης παραλληλοποίησης DOACROSS φωλιασμένων βρόχων που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Ο παράλληλος κώδικας που παράγεται είναι της μορφής SPMD (Single Program Multiple Data) και η παράλληλη αρχιτεκτονική στην οποία θα εκτελεστεί ο κώδικας είναι κατανεμημένης μνήμης. Έτσι, οι ανάγκες επικοινωνίας καλύπτονται με την αποστολή και λήψη μηνυμάτων. Για το σκοπό αυτό ο παραγόμενος κώδικας κάνει χρήση της βιβλιοθήκης ανταλλαγής μηνυμάτων MPI (Message Passing Interface) [MPI94], [MPI96].

1.3 Βιβλιογραφική Επισκόπηση

Στην ενότητα αυτή θα αναφερθούμε με περισσότερη λεπτομέρεια σε ερευνητικές εργασίες της βιβλιογραφίας που σχετίζονται άμεσα με την παρούσα διατριβή. Ιστορικά, η πιο σημαντική ερευνητική εργασία παρουσιάστηκε από τους Irigoin και Triolet το 1988 [IT88]. Οι συγγραφείς προτείνουν για πρώτη φορά το μετασχηματισμό υπερκόμβων ως ένα νέο μετασχηματισμό φωλιασμένων βρόχων, που θα επιφέρει καλύτερη αξιοποίηση της λανθάνουσας μνήμης σε ακολουθιακά προγράμματα και μείωση του κόστους επικοινωνίας σε παραλληλοποιημένους αλγορίθμους. Δίνουν επίσης και το μαθηματικό υπόβαθρο του συγκεκριμένου μετασχηματισμού, ο οποίος ορίζεται με τη βοήθεια n παράλληλων υπερεπιπέδων. Έδωσαν αυστηρά κριτήρια για την εγκυρότητα ενός μετασχηματισμού και άλλα κριτήρια που εξασφαλίζουν ότι οι σχηματιζόμενοι υπερκόμβοι θα μπορούν να εκτελεστούν χωρίς διακοπή, θα είναι όλοι όμοιοι, φραγμένοι και η ένωσή τους θα παράγει τον αρχικό χώρο επαναλήψεων.

Ιδιαίτερο ενδιαφέρον, και ουσιαστικά κίνητρο για την παρούσα διατριβή, αποτελούν μια σειρά εργασιών που ασχολούνται με την επιλογή ενός "βέλτιστου" μετασχηματισμού υπερκόμβων. Οι Ramanujam και Sadayappan [RS92] μελέτησαν το μετασχηματισμό υπερκόμβων αποκλειστικά για την περίπτωση της πολυεπεξεργασίας. Απέδειξαν την ισοδυναμία ανάμεσα στο πρόβλημα της εύρεσης ενός κατάλληλου πίνακα υπερεπιπέδων H και στο πρόβλημα της εύρεσης του κώνου που περικλείει ένα σύνολο εξαρτήσεων. Παρουσίασαν επίσης μία προσέγγιση για την εύρεση μετασχηματισμών υπερκόμβων που θα ελαχιστοποιούν το κόστος επικοινωνίας ανά υπερκόμβο. Η προσέγγισή τους βασίζεται στον ακέραιο γραμμικό προγραμματισμό, με συνάρτηση ελαχιστοποίησης τον όγκο της επικοινωνίας και περιορισμό την εγκυρότητα του πίνακα υπερκόμβων. Οι Boulet, Darte, Risset και Robert [BDRR94] ασχολήθηκαν επίσης με το πρόβλημα της επιλογής του βέλτιστου από πλευράς επικοινωνίας μετασχηματισμού υπερκόμβων και έδειξαν ότι αυτός θα πρέπει να επιλεγεί έτσι ώστε να ισχύει $H = D^{-1}$ (όπου H είναι ο πίνακας που περιγράφει το μετασχηματισμό υπερκόμβων και D είναι ο πίνακας που περιέχει τα διανύσματα εξάρτησης). Ο Xue [Xue97a], για το ίδιο πρόβλημα, υπέδειξε ότι οι διευθύνσεις του μετασχηματισμού υπερκόμβων θα πρέπει να επιλέγονται από την επιφάνεια του κώνου υπερκόμβων (tiling cone), ο οποίος είναι κάθετος στον κώνο των διανυσμάτων εξάρτησης.

Οι Hodzic και Shang [HS98], [HS02] αναζήτησαν το βέλτιστο μετασχηματισμό υπερκόμβων, με κριτήριο το συνολικό χρόνο εκτέλεσης του αλγορίθμου. Χώρισαν το μετασχηματισμό σε τρεις ξεχωριστές παραμέτρους, το μέγεθος, το βασικό σχήμα και την αναλογία των πλευρών του μετασχηματισμού, και μελέτησαν κάθε μία από αυτές τις παραμέτρους ξεχωριστά. Για το βέλτιστο μέγεθος δίνουν ένα πολυώνυμο n βαθμού, η λύση του οποίου δίνει το βέλτιστο μέγεθος υπερκόμβου. Επίσης, απέδειξαν ότι και σ' αυτή την περίπτωση, το βέλτιστο βασικό σχήμα

από πλευράς ελάχιστου χρόνου εκτέλεσης προκύπτει με επιλογή διευθύνσεων από τον κώνο υπερκόμβων του αλγορίθμου. Τέλος, η αναλογία των πλευρών του μετασχηματισμού προκύπτει με βάση το χώρο επαναλήψεων, έναν παράγοντα που για πρώτη φορά εισήγαγαν στην επιλογή ενός αποδοτικού μετασχηματισμού. Οι Hogstedt, Carter και Ferrante στην εργασία τους [HCF03] ασχολήθηκαν επίσης με το βέλτιστο σχήμα του μετασχηματισμού υπερκόμβων που θα μειώνει τον άεργο χρόνο (idle time) των επεξεργαστών, δηλαδή το χρόνο που οι επεξεργαστές αναμένουν δεδομένα ή συγχρονίζονται. Κατέληξαν και αυτοί στην επιλογή του σχήματος του μετασχηματισμού υπερκόμβων με βάση τον κώνο υπερκόμβων, επιπρόσθετα όμως έδειξαν ότι μερικοί επιπλέον μετασχηματισμοί είναι βέλτιστοι.

Είναι αξιοσημείωτο το γεγονός, ότι παρά το μεγάλο αριθμό δημοσιεύσεων σχετικά με την επιλογή ενός μη-ορθογώνιου μετασχηματισμού υπερκόμβων, δεν υπάρχει μέθοδος υλοποίησης των παραπάνω μεθόδων. Με άλλα λόγια δεν υπάρχει μέθοδος παραγωγής παράλληλου κώδικα για μη-ορθογώνιους μετασχηματισμούς υπερκόμβων. Χαρακτηριστικά αναφέρουμε ότι όλες οι παραπάνω εργασίες είναι καθαρά θεωρητικές και δεν παρουσιάζουν πειραματικά αποτελέσματα, που να επιβεβαιώνουν τη σχετική θεωρία. Οι Ancourt και Irigoin [AI91] παρουσιάζουν μία μέθοδο παραγωγής σειριακού μετασχηματισμένου κώδικα, που στηρίζεται στην κατασκευή κατάλληλων συστημάτων ανισοτήτων που περιγράφουν τον αρχικό χώρο και το μετασχηματισμό υπερκόμβων (βλ. Ενότητα 2.6.3). Προτείνουν τη μέθοδο απαλοιφής Fourier-Motzkin για το μετασχηματισμό των παραπάνω συστημάτων σε μορφή κατάλληλη, ώστε οι χώροι τους οποίους περιγράφουν να μπορούν να διασχιστούν από ένα φωλιασμένο βρόχο. Η παραπάνω μέθοδος, αν και λύνει ουσιαστικά το πρόβλημα της εκτέλεσης βρόχων που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων, παρουσιάζει αρκετά πρακτικά προβλήματα. Τα συστήματα ανισοτήτων που δημιουργούνται είναι μεγάλα, και σε συνδυασμό με την πολύ μεγάλη χρονική και χωρική πολυπλοκότητα της Fourier-Motzkin καθιστούν τη μέθοδο μη πρακτική σε πολλές περιπτώσεις. Επιπρόσθετα, ο παραγόμενος κώδικας περιέχει μεγάλες επιβαρύνσεις κατά την αποτίμηση των μεταβλητών ελέγχου του νέου φωλιασμένου βρόχου, κάτι που επιφέρει σημαντικές καθυστερήσεις κατά την εκτέλεση του παραλληλοποιημένου αλγορίθμου. Από την άλλη, οι Ancourt και Irigoin παρουσιάζουν μόνο τη μέθοδο παραγωγής του σειριακού μετασχηματισμένου κώδικα και δεν κάνουν λόγο για τη διαδικασία παραλληλοποίησης. Στην παρούσα διατριβή, παρουσιάζουμε μια αποδοτικότερη μέθοδο παραγωγής σειριακού μετασχηματισμένου κώδικα και συνεχίζουμε τη διαδικασία μέχρι την τελική παραγωγή παράλληλου κώδικα για αρχιτεκτονικές κατανεμημένης μνήμης.

Οι παράλληλοι μεταγλωττιστές FORTRAN [FHK+91], [SLR+95], [CMZ92], [AMC97] απαιτούν από το χρήστη να καθορίσει μία κατανομή δεδομένων, με βάση την οποία ο μεταγλωττι-

στής πραγματοποιεί την παραλληλοποίηση του αρχικού σειριακού προγράμματος. Σε όλες τις περιπτώσεις ακολουθείται ο κανόνας του "ο κατέχων-υπολογίζει" για να προσδιοριστεί ποιες επαναλήψεις θα ανατεθούν σε ποιούς επεξεργαστές. Κάτω από αυτούς τους περιορισμούς, οι μη-ορθογώνιοι μετασχηματισμοί υπερκόμβων δεν μπορούν να υλοποιηθούν στους μεταγλωττιστές αυτούς. Αντίθετα, οι Amarasinghe και Lam στην εργασία τους [AL93] παρουσιάζουν μία γενική μέθοδο παραγωγής SPMD κώδικα, που στηρίζεται στην περιγραφή του χώρου δεικτών, του χώρου δεδομένων και των συνόλων επικοινωνίας με συστήματα ανισοτήτων. Το μοντέλο της εργασίας αυτής είναι πολύ γενικό και θα μπορούσε να περιγράψει μη-ορθογώνιους μετασχηματισμούς υπερκόμβων. Στην πράξη όμως κάτι τέτοιο δεν έχει γίνει, και είναι βέβαιο ότι ένα μοντέλο που στηρίζεται σε ένα πολύ μεγάλο αριθμό ανισοτήτων, δεν θα μπορούσε να γεννήσει αποδοτικό κώδικα στην περίπτωση των μη-ορθογώνιων μετασχηματισμών. Η παρούσα εργασία εστιάζει στη βελτιστοποίηση του παραγόμενου παράλληλου κώδικα στην ειδική περίπτωση των μετασχηματισμών υπερκόμβων.

Οι Tang και Xue [TX00] παρουσίασαν μία πλήρη μέθοδο παραγωγής παράλληλου SPMD κώδικα για αρχιτεκτονικές κατανεμημένης μνήμης, που περιορίζεται όμως μόνο σε ορθογώνιους μετασχηματισμούς υπερκόμβων. Η διαδικασία που περιγράφουν χωρίζεται σε δύο φάσεις, την παραγωγή του σειριακού μετασχηματισμένου κώδικα (sequential tiled code) και την παραλληλοποίηση. Η παραλληλοποίηση με τη σειρά της αποτελείται από την απεικόνιση των επαναλήψεων σε επεξεργαστές, την κατανομή των δεδομένων και την παραγωγή των πρωτογενών κλήσεων ανταλλαγής μηνυμάτων. Στην παρούσα εργασία ακολουθείται ακριβώς η ίδια διαδικασία για την παραγωγή SPMD κώδικα, για οποιονδήποτε μετασχηματισμό υπερκόμβων που ορίζεται από έναν n-διάστατο πίνακα H.

1.4 Οργάνωση της Διατριβής

Η παρούσα εργασία οργανώνεται ως εξής:

Στο παρόν κεφάλαιο συνοψίζεται το αντικείμενο της διατριβής, η συμβολή της διατριβής καθώς και οι δημοσιεύσεις που προέκυψαν από την έρευνα που οδήγησε στην εργασία αυτή. Επιπρόσθετα, τοποθετείται η εργασία αυτή στο γενικότερο πεδίο των μεταγλωττιστών υψηλής επίδοσης και των αυτόματων παραλληλοποιητών και γίνεται εκτενέστερη αναφορά στη βιβλιογραφία που επηρέασε άμεσα τη σχετική έρευνα.

Το Κεφάλαιο 2 παρουσιάζει θεμελιώδεις, βασικές γνώσεις από τη θεωρία των φωλιασμένων βρόχων και εισαγάγει τους σχετικούς ορισμούς και συμβολισμούς που είναι απαραίτητοι για

την παρουσίαση των προτεινόμενων μεθόδων των κεφαλαίων που ακολουθούν. Πιο συγκεκριμένα, δίνει το αλγεβρικό μοντέλο των φωλιασμένων βρόχων, τον ορισμό των εξαρτήσεων και παρουσιάζει με μεγαλύτερη λεπτομέρεια τους μετασχηματισμούς των φωλιασμένων βρόχων. Οι μετασχηματισμοί αυτοί διακρίνονται σε γραμμικούς και μη-γραμμικούς. Οι γραμμικοί μετασχηματισμοί με τη σειρά τους διακρίνονται σε ορθομοναδιαίους και μη-ορθομοναδιαίους. Γίνεται λεπτομερής παρουσίαση του σημαντικότερου μη-γραμμικού μετασχηματισμού, του μετασχηματισμού υπερκόμβων, ο οποίος αποτελεί και το αντικείμενο της παρούσας διατριβής. Για όλες τις παραπάνω κατηγορίες μετασχηματισμών δίνονται οι ήδη υπάρχουσες μέθοδοι παραγωγής κώδικα. Το Κεφάλαιο 2 κλείνει με την παρουσίαση της γραμμικής χρονικής δρομολόγησης και του μοντέλου των φωλιασμένων βρόχων που απασχολούν την παρούσα εργασία.

Τα Κεφάλαια 3 και 4 παρουσιάζουν την προτεινόμενη μέθοδο για την αυτόματη παραγωγή παράλληλου SPMD κώδικα για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Συγκεκριμένα στο Κεφάλαιο 3 παρουσιάζεται μια νέα μέθοδος παραγωγής σειριακού μετασχηματισμένου κώδικα, η οποία υπερέχει συγκρινόμενη με την ήδη υπάρχουσα μέθοδο που παρουσιάστηκε στην εργασία [AI91], τόσο στο χρόνο μεταγλώττισης όσο και στην ποιότητα του παραγόμενου κώδικα. Η διαδικασία διαμερίζεται στις επιμέρους διεργασίες της διάσχισης των υπερκόμβων και της διάσχισης των σημείων στο εσωτερικό των υπερκόμβων (Σχήμα 1.2).

Το Κεφάλαιο 4 έρχεται σε συνέχεια του Κεφαλαίου 3 και ολοκληρώνει τη διαδικασία παραγωγής SPMD κώδικα με την παραλληλοποίηση του σειριακού μετασχηματισμένου κώδικα. Στο κεφάλαιο αυτό παρουσιάζονται τεχνικές που υλοποιούν την παραλληλοποίηση και έχουν να κάνουν με την ανάθεση των υπερκόμβων σε επεξεργαστές, την κατανομή των δεδομένων και την αυτόματη παραγωγή των πρωτογενών κλήσεων επικοινωνίας.

Το Κεφάλαιο 5 παρουσιάζει πειραματικά αποτελέσματα για την αποδοτικότητα της προτεινόμενης μεθόδου. Συγκεκριμένα, γίνεται σύγκριση με τη μέθοδο της εργασίας [AI91] σε σχέση με τη διαδικασία μεταγλώττισης και την ποιότητα του σειριακού μετασχηματισμένου κώδικα. Επιπρόσθετα, παρουσιάζονται αποτελέσματα από την εκτέλεση πραγματικών αλγορίθμων για τους οποίους έχει παραχθεί αυτόματα παράλληλος κώδικας με τη βοήθεια της μεθόδου που παρουσιάστηκε στα Κεφάλαια 3 και 4.

Το Κεφάλαιο 6 συνοψίζει τα συμπεράσματα της παρούσας διατριβής και προτείνει θέματα για μελλοντική μελέτη και έρευνα. Τέλος, στο Παράρτημα Α γίνεται μια συνοπτική παρουσίαση της μεθόδου απαλοιφής ανισοτήτων Fourier-Motzkin, στο Παράρτημα Β δίνονται οι κώδικες των προγραμμάτων που χρησιμοποιήθηκαν στα πειράματα, στο Παράρτημα Γ δίνονται πίνακες μετασχηματισμού υπερκόμβων που χρησιμοποιήθηκαν στα πειράματα και στο Παράρτημα Δ

δίνεται ένας πίνακας με τα σύμβολα που χρησιμοποιούνται στην παρούσα εργασία.

1.5 Συμβολή της Διατριβής

Οι βασικές καινοτομίες της διατριβής αυτής είναι:

- Η παρουσίαση μιας ολοκληρωμένης μεθόδου παραγωγής παράλληλου SPMD κώδικα που εκτελείται σε αρχιτεκτονικές κατανεμημένης μνήμης, για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Η μέθοδος λαμβάνει υπ' όψη γενικούς παραλληλόγραμμους μετασχηματισμούς (όχι απαραίτητα τετράγωνους ή ορθογώνιους) και γενικούς κυρτούς χώρους επαναλήψεων. Πιο συγκεκριμένα, συνεισφέρει τα εξής:

  - Μία νέα μέθοδο παραγωγής σειριακού μετασχηματισμένου κώδικα, που σε σχέση με την ήδη υπάρχουσα που παρουσιάστηκε στην εργασία [AI91], παρουσιάζει πολύ σημαντική επιτάχυνση στο χρόνο μεταγλώττισης (compilation time) ενώ ταυτόχρονα παράγει πολύ πιο αποδοτικό κώδικα.

  - Μία μέθοδο παραλληλοποίησης για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων, η οποία επιφέρει πολύ μικρό επιπλέον κόστος στον παραγόμενο κώδικα και αντιμετωπίζει αποδοτικά τα προβλήματα της ανάθεσης των υπερκόμβων σε επεξεργαστές, της κατανομής των δεδομένων και της παραγωγής πρωτογενών κλήσεων ανταλλαγής μηνυμάτων.

- Η παρουσίαση πειραματικών αποτελεσμάτων που επιβεβαιώνουν τα θεωρητικά συμπεράσματα των εργασιών [HS02] και [HCF03]. Με βάση τα συμπεράσματα αυτά, αν ο μετασχηματισμός υπερκόμβων είναι παράλληλος με την επιφάνεια του κώνου υπερκόμβων, τότε ο συνολικός χρόνος εκτέλεσης του μετασχηματισμένου φωλιασμένου βρόχου είναι μικρότερος από το χρόνο εκτέλεσης που αντιστοιχεί σε οποιονδήποτε άλλο μετασχηματισμό.

1.6 Δημοσιεύσεις

Εργασίες σε Επιστημονικά Περιοδικά Η έρευνα που διεξήχθη στα πλαίσια της παρούσας διατριβής, οδήγησε στις εξής δημοσιεύσεις σε επιστημονικά περιοδικά και διεθνή συνέδρια.

- N. Koziris, A. Sotiropoulos and G. Goumas, **"A Pipelined Schedule to Minimize Completion Time for Loop Tiling with Computation and Communication Overlapping"**, *Journal of Parallel and Distributed Computing,* Vol. 63, No. 11, November 2003.

- G. Goumas, M. Athanasaki and N. Koziris, **"An Efficient Code Generation Technique for Tiled Iteration Spaces"**, *IEEE Transactions on Parallel and Distributed Systems,* Vol. 14, No 10, October 2003.

- G. Goumas, M. Athanasaki and N. Koziris, "**Code Generation Methods for Tiling Transformations"**, *Journal of Information Science and Engineering, Special Section on Parallel and Distributed Systems,* Vol. 18, No 5, September 2002.

Εργασίες σε Διεθνή Συνέδρια

- G. Goumas, N. Drosinos, M. Athanasaki and N. Koziris, **"Compiling Tiled Iteration Spaces for Clusters"**, Proceedings of the *IEEE International Conference on Cluster Computing (CLUSTER 2002),* Chicago, USA, September 2002.

- G. Goumas, N. Drosinos, M. Athanasaki and N. Koziris, **"Data Parallel Code Generation for Arbitrarily Tiled Loop Nests"**, Proceedings of the *2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, Las Vegas, USA, June 2002.

- G. Goumas, M. Athanasaki and N. Koziris, **"Automatic Code Generation for Executing Nested Loops onto Parallel Architectures"**, Proceedings of the *ACM Symposium on Applied Computing (SAC 2002)- Parallel and Distributed Systems and Networking Track,* Madrid, Spain, March 2002.

- G. Goumas, A.Sotiropoulos and N. Koziris, **"Minimizing Completion Time for Loop Tiling with Computation and Communication Overlapping"**, Proceedings of the *2001 International Parallel and Distributed Processing Symposium (IPDPS 2001),* IEEE Press, San Francisco, California, April 2001 (best paper award).

- N. Drosinos, G. Goumas, M. Athanasaki and N. Koziris, **"Delivering High Performance to Parallel Applications Using Advanced Scheduling"**, Proceedings of the *Parallel Computing 2003 (ParCo 2003),* Dresden, Germany, September 2003.
- G. Goumas, N. Drosinos, M. Athanasaki and N. Koziris, **"Automatic Parallel Code Generation for Tiled Nested Loops"**, to appear in *19th ACM Symposium on Applied Computing (SAC 2004),* Nicosia, Cyprus, 14-17 March, 2004.

Κεφάλαιο 2

Βασική Θεωρία Φωλιασμένων Βρόχων

Στο κεφάλαιο αυτό θα αναφερθούμε στη βασική θεωρία των φωλιασμένων βρόχων, καλύπτοντας θέματα όπως η μαθηματική τους περιγραφή, η μελέτη των εξαρτήσεών τους, οι μετασχηματισμοί που εφαρμόζονται, η χρονική δρομολόγηση κ.ά. Όπως αναφέρθηκε και στο Κεφάλαιο 1, οι φωλιασμένοι βρόχοι αποτελούν αντικείμενο εκτενούς μελέτης, καθώς είναι η βασικότερη πηγή καθυστέρησης στα ακολουθιακά προγράμματα. Βασικός στόχος της έρευνας γύρω από τους φωλιασμένους βρόχους είναι η εύρεση κατάλληλων μετασχηματισμών που, αλλάζοντας με έγκυρο τρόπο τη σειρά εκτέλεσης των στιγμιοτύπων ενός φωλιασμένου βρόχου, θα οδηγήσουν σε καλύτερη αξιοποίηση των αρχιτεκτονικών χαρακτηριστικών ενός υπολογιστικού συστήματος υψηλής επίδοσης, και συνακόλουθα σε σημαντική μείωση του συνολικού χρόνου εκτέλεσης.

2.1 Συμβολισμοί

Πριν προχωρήσουμε στην ανάλυση των ιδιοτήτων των φωλιασμένων βρόχων, παραθέτουμε μερικούς συμβολισμούς. Έτσι, θα συμβολίζουμε με Z το σύνολο των ακεραίων αριθμών και με R

το σύνολο των ρητών. Ένα διάνυσμα θα συμβολίζεται ως $\vec{a}$. Αν A είναι ένας πίνακας, τότε το στοιχείο της i γραμμής και j στήλης συμβολίζεται με a_{ij}. Αν $\vec{a_1}, \vec{a_2}, \ldots, \vec{a_k}$ είναι k διανύσματα διάστασης n, τότε ο $k \times n$ πίνακας A που σχηματίζεται αν τα παραπάνω διανύσματα τοποθετηθούν ως διανύσματα-γραμμές, συμβολίζεται $A = [\vec{a_1}, \vec{a_2}, \ldots, \vec{a_k}]^T$. Αν $c = \sigma\tau\alpha\theta$. τότε $\vec{c}$ είναι το διάνυσμα που όλα τα στοιχεία του ισούνται με c. Αν ένα διάνυσμα $\vec{a_1}$ είναι λεξικογραφικά μεγαλύτερο από ένα άλλο $\vec{a_2}$, τότε συμβολίζουμε $\vec{a_1} > \vec{a_2}$. Ανάλογα ορίζονται και τα σύμβολα $<, \leq, \geq$.

2.2 Φωλιασμένοι Βρόχοι - Χώροι Επαναλήψεων

Οι αλγόριθμοι που μας απασχολούν στην παρούσα εργασία περιλαμβάνουν *τέλεια φωλιασμένους βρόχους* (perfectly nested loops) της μορφής:

```
FOR (j1=l1; j1 ≤ u1; j1+=c1)
   FOR (j2=l2; j2 ≤ u2; j2+=c2)
      ...
      FOR (jn=ln; jn ≤ un; jn+=cn)
         S1(j1, j2, ..., jn);
         ...
         Sμ(j1, j2, ..., jn);
      ENDFOR
      ...
   ENDFOR
ENDFOR
```

Ο παραπάνω βρόχος είναι ένας n-διάστατος φωλιασμένος βρόχος ή ισοδύναμα ένας φωλιασμένος βρόχος με *βάθος* n. Οι $j_1, j_2, \ldots, j_n$ είναι οι *μεταβλητές ελέγχου* (control variables) του βρόχου. Στη γενική περίπτωση, τα άνω και κάτω όρια (u_k και l_k αντίστοιχα) μιας μεταβλητής ελέγχου j_k είναι συνάρτηση των $k-1$ εξωτερικότερων μεταβλητών ελέγχου. Στην παρούσα εργασία, τα όρια αυτά είναι της μορφής: $l_k = max(\lceil f_{k1}(j_1, \ldots, j_{k-1}) \rceil, \ldots, \lceil f_{kr}(j_1, \ldots, j_{k-1}) \rceil)$ και $u_k = min(\lfloor g_{k1}(j_1, \ldots, j_{k-1}) \rfloor, \ldots, \lfloor g_{kr}(j_1, \ldots, j_{k-1}) \rfloor)$, όπου οι f_{ki} και g_{ki} είναι γραμμικές συναρτήσεις. Οι εντολές $\mathtt{S}_k$, $1 \leq k \leq \mu$ είναι συνήθως εντολές ανάθεσης που προσπελαύνουν πολυδιάστατους πίνακες και δεν περιέχουν εντολές εισόδου/εξόδου (I/O), ούτε εντολές μεταφοράς του ελέγχου έξω από το βρόχο ή κλήσεις σε υπορουτίνες που τροποποιούν τα δεδομένα.

Ορισμός 2.1 *Ο χώρος δεικτών ή επαναλήψεων (index/iteration space) J ενός φωλιασμένου βρόχου περιλαμβάνει το σύνολο των επαναλήψεων ή στιγμιοτύπων του βρόχου και ορίζεται ως εξής:*

$$J = \{\vec{j} = (j_1, ..., j_n) \mid j_i \in Z \wedge l_i \leq j_i \leq u_i,\ 1 \leq i \leq n\} \tag{2.1}$$

Έτσι, κάθε στοιχείο $\vec{j}$ του παραπάνω συνόλου είναι ένα n-διάστατο *διάνυσμα επανάληψης* και αντιστοιχεί σε ένα και μοναδικό στιγμιότυπο των μεταβλητών ελέγχου $j_1, j_2, \ldots, j_n$ του βρόχου. Ο χώρος επαναλήψεων J μπορεί να περιγραφεί και με ένα σύστημα ανισοτήτων ως εξής:

$$J = \{\vec{j} \in Z^n \mid B\vec{j} \leq \vec{b}\} \tag{2.2}$$

Ο πίνακας B και το διάνυσμα $\vec{b}$ μπορούν εύκολα να προκύψουν από τις γραμμικές εκφράσεις l_k και u_k και αντιστρόφως.

Παράδειγμα 2.1 Έστω οι εξής φωλιασμένοι βρόχοι:

Βρόχος1:

```
FOR (j1=0; j1 ≤7; j1++)
   FOR (j2=0; j2 ≤5; j2++)
      A[j1,j2]=A[j1-1,j2]+A[j1,j2-1];
   ENDFOR
ENDFOR
```

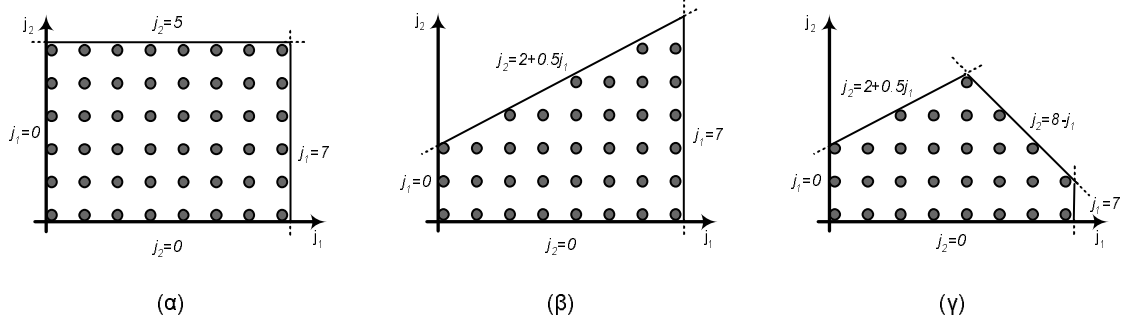
Βρόχος2:

```
FOR (j1=0; j1 ≤7; j1++)
   FOR (j2=0; j2 ≤2+0.5j1; j2++)
      A[j1,j2]=A[j1-1,j2-1]+A[j1-1,j2];
   ENDFOR
ENDFOR
```

Βρόχος3:

```
FOR (j1=0; j1 ≤7; j1++)
   FOR (j2=0; j2 ≤min(2+0.5j1,8-j1); j2++)
      A[j1,j2]=A[j1-1,j2]+A[j1-1,j2-1]+A[j1-1,j2+1];
   ENDFOR
ENDFOR
```

Ο Χώρος Επαναλήψεων για το Βρόχο1 είναι $J_1 = \{\vec{j} = (j_1, j_2) \mid j_1, j_2 \in Z \wedge 0 \leq j_1 \leq 7, 0 \leq j_2 \leq 5\}$ και το αντίστοιχο σύστημα ανισοτήτων που τον περιγράφει είναι $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 7 \\ 5 \\ 0 \\ 0 \end{pmatrix}$. Ανάλογα, για το Βρόχο2 θα ισχύει $J_2 = \{\vec{j} = (j_1, j_2) \mid j_1, j_2 \in Z \wedge 0 \leq j_1 \leq 7, 0 \leq j_2 \leq 2+0.5j_1\}$ και το αντίστοιχο σύστημα θα είναι $\begin{pmatrix} 1 & 0 \\ -0.5 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 7 \\ 2 \\ 0 \\ 0 \end{pmatrix}$, ενώ για το Βρόχο3 θα ισχύει $J_3 = \{\vec{j} = (j_1, j_2) \mid j_1, j_2 \in Z \wedge 0 \leq j_1 \leq 7, 0 \leq j_2 \leq min(2+0.5j_1, 8-j_1)\}$ και το αντίστοιχο σύστημα θα είναι $\begin{pmatrix} 1 & 0 \\ -0.5 & 1 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 7 \\ 2 \\ 8 \\ 0 \\ 0 \end{pmatrix}$. Στο Σχήμα 2.1 φαίνεται η απεικόνιση των παραπάνω χώρων στο επίπεδο, καθώς και τα ημιεπίπεδα που τους ορίζουν. ⊣



Σχήμα 2.1: Χώροι Επαναλήψεων για το Παράδειγμα 2.1: (α) Βρόχος1 (β) Βρόχος2 (γ) Βρόχος3

2.3 Εξαρτήσεις

Αυτό που μας ενδιαφέρει κυρίως κατά τη μελέτη ενός φωλιασμένου βρόχου, είναι η δυνατότητα αναδιάταξης της σειριακής εκτέλεσής του, ή η ανίχνευση επαναλήψεων που μπορούν να εκτελεστούν ταυτόχρονα. Και στις δύο παραπάνω περιπτώσεις, πρέπει να εξασφαλίζεται ότι το αποτέλεσμα του αναδιατεταγμένου ή του παραλληλοποιημένου φωλιασμένου βρόχου θα είναι πάντα το ίδιο με το αποτέλεσμα του αρχικού. Το 1966, ο Bernstein [Ber66] υπέδειξε ότι δύο διαφορετικές επαναλήψεις $\vec{j_1}$ και $\vec{j_2}$ μπορούν να εκτελεστούν παράλληλα αν:

1. Η επανάληψη $\vec{j_1}$ δεν γράφει σε θέση μνήμης την οποία διαβάζει η επανάληψη $\vec{j_2}$.
2. Η επανάληψη $\vec{j_2}$ δεν γράφει σε θέση μνήμης την οποία διαβάζει η επανάληψη $\vec{j_1}$.
3. Η επανάληψη $\vec{j_1}$ δεν γράφει σε θέση μνήμης στην οποία επίσης γράφει η επανάληψη $\vec{j_2}$.

Ουσιαστικά, οι τρεις παραπάνω προτάσεις μας εισάγουν στην έννοια των εξαρτήσεων. Έτσι, διατυπωμένο διαφορετικά, εξάρτηση μεταξύ δύο επαναλήψεων υπάρχει όταν οι επαναλήψεις αυτές προσπελαύνουν την ίδια θέση μνήμης, και τουλάχιστον η μία πραγματοποιεί εγγραφή. Παρακάτω δίδεται τυπικά ο ορισμός της εξάρτησης.

Ορισμός 2.2 *Υπάρχει εξάρτηση από μία εντολή S_1 σε μία εντολή S_2, σε έναν κοινό φωλιασμένο βρόχο, αν και μόνο αν υπάρχουν δύο διανύσματα επανάληψης $\vec{j_1}$ και $\vec{j_2}$ τέτοια ώστε:*

1. *$\vec{j_1} < \vec{j_2}$ ή $\vec{j_1} = \vec{j_2}$ και υπάρχει μονοπάτι από την εντολή S_1 στην S_2 στο σώμα του βρόχου.*
2. *Η εντολή S_1 προσπελαύνει τη διεύθυνση μνήμης M στην επανάληψη $\vec{j_1}$ και η εντολή S_2 προσπελαύνει τη διεύθυνση μνήμης M στην επανάληψη $\vec{j_2}$.*
3. *Μία τουλάχιστον από αυτές τις προσπελάσεις είναι εγγραφή.*

Με βάση τον παραπάνω ορισμό, αν $\vec{j_1} < \vec{j_2}$ τότε η εξάρτηση δημιουργείται εξ αιτίας των επαναλήψεων του βρόχου και ονομάζεται *εξαρτώμενη* από το βρόχο (loop-carried). Αν ισχύει $\vec{j_1} = \vec{j_2}$ τότε η εξάρτηση είναι *ανεξάρτητη* των επαναλήψεων (loop-independent). Επίσης,

αν η εγγραφή προηγείται της ανάγνωσης τότε η εξάρτηση ονομάζεται *εξάρτηση ροής* (flow ή true dependence), ενώ αν συμβαίνει το αντίθετο η εξάρτηση ονομάζεται *αντιεξάρτηση* (anti-dependence). Στην περίπτωση που έχουμε δύο εγγραφές τότε η εξάρτηση ονομάζεται *εξάρτηση εξόδου* (output dependence). Στο μοντέλο των αλγορίθμων που μας απασχολούν στη συγκεκριμένη εργασία, θεωρούμε ότι υπάρχουν μόνο εξαρτώμενες από το βρόχο εξαρτήσεις και εξαρτήσεις ροής. Οι αντιεξαρτήσεις και οι εξαρτήσεις εξόδου μπορούν να εξαλειφθούν με τη χρήση επιπλέον μεταβλητών [CDRV98]. Για να είναι έγκυρη η σειριακή εκτέλεση ενός φωλιασμένου βρόχου, θα πρέπει τα διανύσματα εξάρτησής του να είναι λεξικογραφικά θετικά.

Γενικά είναι βολικό να χαρακτηρίζουμε μία εξάρτηση από την απόσταση ανάμεσα στην πηγή και τον προορισμό της στο χώρο επαναλήψεων του φωλιασμένου βρόχου. Για το σκοπό αυτό χρησιμοποιούνται τα *διανύσματα απόστασης* (distance vectors) και τα *διανύσματα κατεύθυνσης* (direction vectors). Ακολουθούν οι σχετικοί ορισμοί:

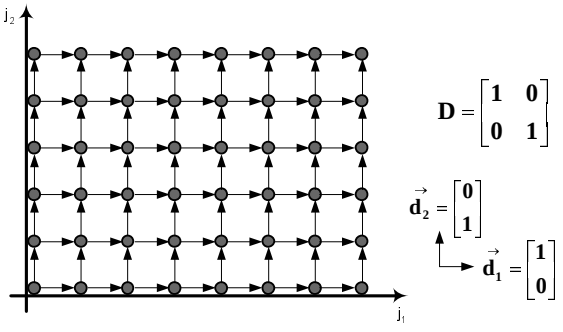
Ορισμός 2.3 *Αν υποθέσουμε ότι υπάρχει εξάρτηση από μία εντολή S_1 στην επανάληψη $\vec{j_1}$, σε μία εντολή S_2 στην επανάληψη $\vec{j_2}$ ενός φωλιασμένου βρόχου, τότε το διάνυσμα απόστασης $\vec{d}$ της εξάρτησης ορίζεται ως: $\vec{d} = \vec{j_2} - \vec{j_1}$.*

Ορισμός 2.4 *Αν υποθέσουμε ότι υπάρχει εξάρτηση από μία εντολή S_1 στην επανάληψη $\vec{j_1}$, σε μία εντολή S_2 στην επανάληψη $\vec{j_2}$ ενός φωλιασμένου βρόχου, τότε το διάνυσμα κατεύθυνσης $\vec{D} = (D_1, D_2, \ldots, D_n)$ της εξάρτησης ορίζεται ως: (1) $D_k =$ "$<$" αν $d_k < 0$, (2) $D_k =$ "$=$" αν $d_k = 0$ και (3) $D_k =$ "$>$" αν $d_k > 0$, όπου $\vec{d} = (d_1, d_2, \ldots, d_n)$ είναι το αντίστοιχο διάνυσμα απόστασης της εξάρτησης.*

Το διάνυσμα απόστασης όπως προκύπτει από τον ορισμό, ενώνει τα σημεία $\vec{j_1}$ και $\vec{j_2}$ (με κατεύθυνση από το $\vec{j_1}$ στο $\vec{j_2}$) στο χώρο επαναλήψεων J. Το διάνυσμα κατεύθυνσης παρέχει πιο γενικές πληροφορίες σχετικά με την διεύθυνση της εξάρτησης σε μία διάσταση. Τα διανύσματα απόστασης περιγράφουν ακριβώς τις εξαρτήσεις για τους αλγορίθμους που μας απασχολούν και είναι αυτά που θα χρησιμοποιηθούν στη συνέχεια. Για συντομία και απλότητα θα αναφέρονται ως διανύσματα εξάρτησης. Ένας αλγόριθμος μπορεί να έχει γενικά q διανύσματα εξάρτησης. Ο $n \times q$ πίνακας D περιέχει σαν στήλες τα διανύσματα εξάρτησης και ονομάζεται *πίνακας*

εξαρτήσεων.

Παράδειγμα 2.2 Για τους τρεις χώρους επαναλήψεων του Παραδείγματος 2.1 έχουμε τις παρακάτω εξαρτήσεις: Με βάση τον Ορισμό 2.3, στο Βρόχο1 υπάρχουν δύο εξαρτήσεις που περιγράφονται από τα διανύσματα απόστασης $\vec{d_1} = (1,0)$ και $\vec{d_2} = (0,1)$ και από τον πίνακα εξαρτήσεων $D_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Στο Βρόχο2 υπάρχουν δύο εξαρτήσεις που περιγράφονται από τα διανύσματα $\vec{d_1} = (1,1)$ και $\vec{d_2} = (1,0)$ και από τον πίνακα εξαρτήσεων $D_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, και στο Βρόχο3 υπάρχουν τρεις εξαρτήσεις που περιγράφονται από τα διανύσματα $\vec{d_1} = (1,0)$, $\vec{d_2} = (1,1)$ και $\vec{d_3} = (1,-1)$ και από τον πίνακα εξαρτήσεων $D_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix}$. Το Σχήμα 2.2 δείχνει τα διανύσματα εξάρτησης στο χώρο επαναλήψεων του Βρόχου1. ⊣



Σχήμα 2.2: Χώρος επαναλήψεων και διανύσματα εξάρτησης

2.4 Γραμμικοί Μετασχηματισμοί

Προκειμένου να αξιοποιήσουμε καλύτερα τα ιδιαίτερα χαρακτηριστικά των υπολογιστών υψηλής επίδοσης, είναι συχνά αναγκαίο να μετασχηματίσουμε τους προς εκτέλεση φωλιασμένους βρόχους. Για παράδειγμα, είναι δυνατόν να γίνει καλύτερη χρήση της λανθάνουσας μνήμης και αποδοτικότερη απεικόνιση σε επεξεργαστές με διανυσματικές μονάδες υπολογισμών (vector processors) ή σε αρχιτεκτονικές που περιλαμβάνουν πολλούς επεξεργαστές. Η σημαντικότερη ίσως κατηγορία μετασχηματισμών βρόχων, που έχει προκαλέσει μεγάλο ερευνητικό ενδιαφέρον και πληθώρα εργασιών, είναι οι γραμμικοί μετασχηματισμοί.

Ένας γραμμικός μετασχηματισμός ενός n-διάστατου φωλιασμένου βρόχου ορίζεται με τη βοήθεια ενός $n \times n$ πίνακα ακεραίων T. Έτσι, αν ένας γραμμικός μετασχηματισμός T εφαρμοστεί σε ένα φωλιασμένο βρόχο με χώρο επαναλήψεων J, τότε ο χώρος μετασχηματίζεται σε ένα νέο J', για τον οποίο ισχύει:

$$J' = \{\vec{j'} = (j'_1, ..., j'_n) \mid \vec{j'} = T\vec{j} \mid \vec{j} \in J\} \tag{2.3}$$

Αν D είναι ο πίνακας εξαρτήσεων του αρχικού φωλιασμένου βρόχου, τότε $D' = TD$ είναι ο πίνακας εξαρτήσεων του μετασχηματισμένου φωλιασμένου βρόχου. Για να είναι *έγκυρος* ένας μετασχηματισμός T, θα πρέπει όλα τα μετασχηματισμένα διανύσματα εξάρτησης (δηλ. οι στήλες του πίνακα D') να είναι λεξικογραφικά θετικά.

Αν ισχύει $det(T) = \pm 1$, τότε ο μετασχηματισμός ονομάζεται *ορθομοναδιαίος* (unimodular), διαφορετικά ονομάζεται *μη-ορθομοναδιαίος* (non-unimodular). Οι ορθομοναδιαίοι μετασχηματισμοί έχουν τα εξής σημαντικά χαρακτηριστικά:

1. Διαδοχικοί μετασχηματισμοί, που παρίστανται σαν γινόμενο πινάκων επιμέρους ορθομοναδιαίων μετασχηματισμών, οδηγούν σε επίσης ορθομοναδιαίους μετασχηματισμούς.

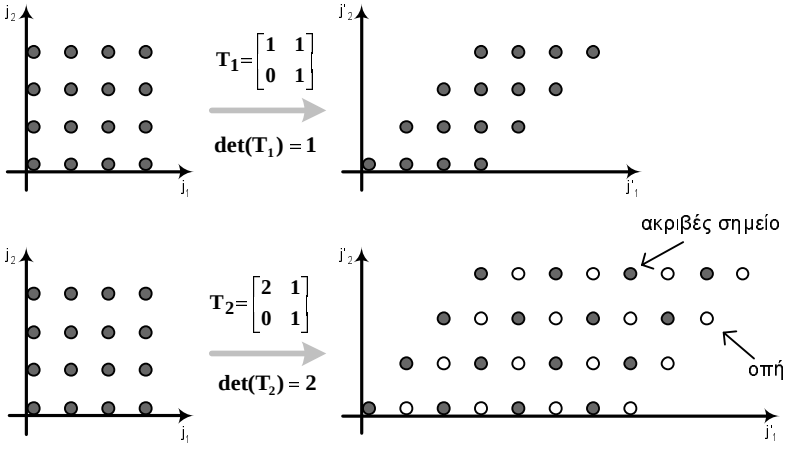
2. Ο αντίστροφος ενός ορθομοναδιαίου μετασχηματισμού είναι επίσης ορθομοναδιαίος.

3. Διατηρούν τον όγκο του χώρου επαναλήψεων. Έτσι, κάθε ακέραιο σημείο του μετασχηματισμένου χώρου αντιστοιχεί σε ένα επίσης ακέραιο σημείο του αρχικού χώρου.

Στο Σχήμα 2.3 ο μετασχηματισμός T_1 είναι ορθομοναδιαίος ενώ ο T_2 μη-ορθομοναδιαίος. Τα λευκά σημεία στη δεύτερη περίπτωση αντιστοιχούν σε ακέραια σημεία του μετασχηματισμένου χώρου, που όμως δεν έχουν ακέραια εικόνα στον αρχικό χώρο. Ονομάζουμε αυτά τα σημεία

οπές του μετασχηματισμένου χώρου. Ονομάζουμε *ακριβή* σημεία του μετασχηματισμένου χώρου τα σημεία που έχουν ακέραια εικόνα στον αρχικό χώρο.

Στη συνέχεια δίνονται τυπικά όσα αναφέρθηκαν παραπάνω, καθώς και μερικές επιπλέον χρήσιμες ιδιότητες των μετασχηματισμών και των χώρων που προκύπτουν.



Σχήμα 2.3: Ορθομοναδιαίος (T_1) και Μη-Ορθομοναδιαίος (T_2) μετασχηματισμός

Ορισμός 2.5 *Ένας τετραγωνικός πίνακας ονομάζεται* ορθομοναδιαίος *αν είναι ακέραιος και η ορίζουσά του ισούται με* ± 1

Ορισμός 2.6 *Αν* A *είναι ένας* $m \times n$ *ακέραιος πίνακας, ονομάζουμε το σύνολο* $\mathcal{L}(A) = \{\vec{y} \mid \vec{y} = A\vec{x} \wedge \vec{x} \in Z^n\}$ *το* πλέγμα *που σχηματίζεται από τις στήλες του* A.

Ορισμός 2.7 *Αν* T *είναι ένας μη-ορθομοναδιαίος μετασχηματισμός, ονομάζουμε* οπές *τα σημεία* $\vec{j'} \in Z^n$ *για τα οποία ισχύει* $T^{-1}\vec{j'} \notin Z^n$.

Θεώρημα 2.1 Αν ο T είναι ένας $m \times n$ ακέραιος πίνακας και ο C είναι ένας $n \times n$ ορθομοναδιαίος πίνακας, τότε: $\mathcal{L}(T) = \mathcal{L}(TC)$.

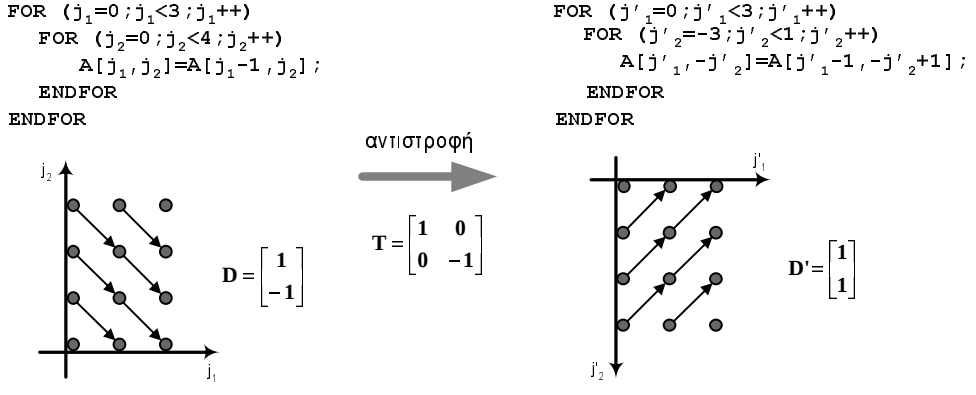
Απόδειξη 2.1 *Δίνεται στο [Ram92].*

Ορισμός 2.8 *Ένας τετραγωνικός αντιστρέψιμος πίνακας* $H = [\vec{h_1}, \ldots, \vec{h_n}]^T \in R^{n \times n}$ *είναι σε* Ερμητιανή Κανονική Μορφή κατά στήλες *(EKM), αν και μόνο αν είναι κάτω τριγωνικός (*$h_{ij} = 0$ *για* $i < j$*), και για κάθε* $i > j$ *ισχύει* $0 \leq h_{ij} < h_{ii}$ *(τα στοιχεία της διαγωνίου είναι τα μεγαλύτερα κάθε γραμμής και κάθε στοιχείο είναι θετικό).*

Θεώρημα 2.2 Αν T είναι ένας $m \times n$ ακέραιος πίνακας με m γραμμικά ανεξάρτητες γραμμές, τότε υπάρχει ένας $n \times n$ ορθομοναδιαίος πίνακας C τέτοιος ώστε $TC = [\widetilde{T}0]$ και ο $\widetilde{T}$ να είναι σε Ερμητιανή Κανονική Μορφή.

Απόδειξη 2.2 *Δίνεται στο [Ram92].*

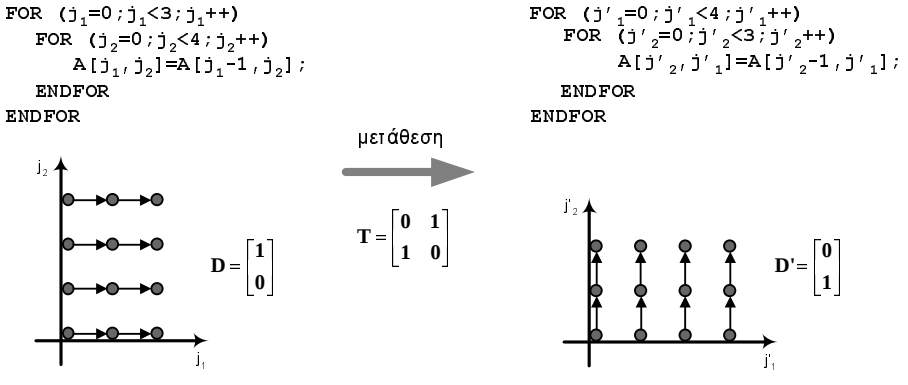
Κάθε ακέραιος $m \times n$ πίνακας με m γραμμικά ανεξάρτητες γραμμές έχει μία και μοναδική Ερμητιανή Κανονική Μορφή. Από τα Θεωρήματα 2.1 και 2.2 προκύπτει ότι $\mathcal{L}(T) = \mathcal{L}(\widetilde{T})$, δηλαδή ότι ένας ακέραιος πίνακας και η ΕΚΜ του έχουν το ίδιο πλέγμα.



Σχήμα 2.4: Αντιστροφή φωλιασμένου βρόχου

Οι πιο συχνά χρησιμοποιούμενοι ορθομοναδιαίοι μετασχηματισμοί είναι η *αντιστροφή* (reversal), η *μετάθεση* (permutation) και η *αλλαγή κλίσης* (skewing). Μία ειδική περίπτωση

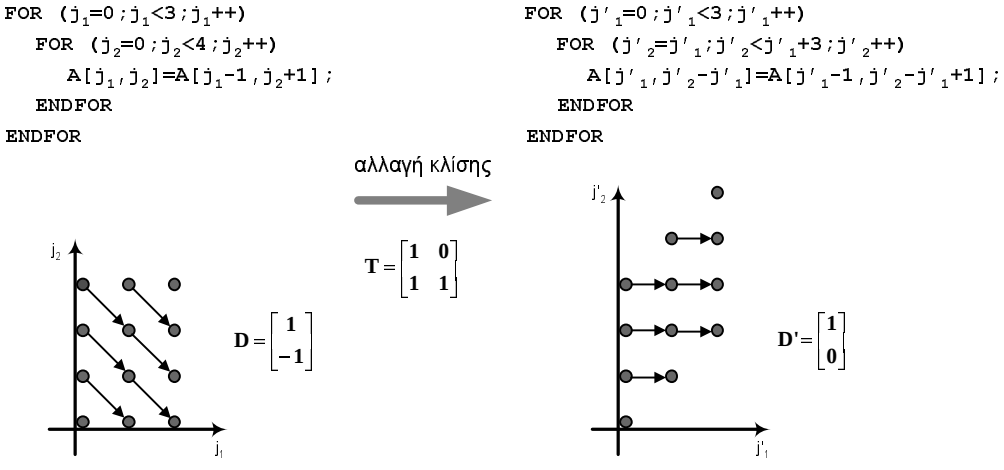
αντιστροφής είναι η *εναλλαγή* (interchange). Ένας πολύ μεγάλος αριθμός εργασιών στη βιβλιογραφία ασχολείται με την εφαρμογή τέτοιων μετασχηματισμών σε φωλιασμένους βρόχους, προκειμένου να βελτιωθεί η τοπικότητα των αναφορών, να αναδειχθεί ο παραλληλισμός στο κατάλληλο επίπεδο κλπ. Για παράδειγμα, οι Kennedy και McKinley [KM92] πρότειναν αλγόριθμο που βελτιώνει την τοπικότητα των αναφορών και αναδεικνύει τον παραλληλισμό στο εξωτερικότερο δυνατό επίπεδο του φωλιασμένου βρόχου ("χονδροειδής" ή coarse-grained παραλληλισμός) με τη βοήθεια κατάλληλων μεταθέσεων. Οι Wolf και Lam [WL91a], [WL91b] προτείνουν μέθοδο προσδιορισμού κατάλληλου συνδυασμού αλλαγής κλίσης, αντιστροφής και μετάθεσης (Skewing - Reversal - Permutation, SRP) για τη δημιουργία *πλήρως μεταθέσιμων* (fully permutable) υποφωλιασμάτων σε ένα φωλιασμένο βρόχο. Πλήρως μεταθέσιμος είναι ένας φωλιασμένος βρόχος που έχει όλα τα στοιχεία των διανυσμάτων εξάρτησής του θετικά ή μηδέν, και γι αυτό το λόγο οποιαδήποτε μετάθεση του παραπάνω βρόχου είναι έγκυρη. Επιπρόσθετα, αν η τάξη του πίνακα εξαρτήσεων του προβλήματος είναι $n_D < n$, τότε ο βρόχος μπορεί να μετασχηματιστεί ώστε να έχει $n - n_D$ `DOALL` βρόχους. Οι συγγραφείς προτείνουν τη μετάθεση των `DOALL` βρόχων στο εξωτερικότερο σημείο του πλήρως μεταθέσιμου βρόχου και τη μετάθεση των βρόχων που προκαλούν επαναχρησιμοποίηση δεδομένων (data reuse) στο εσωτερικότερο. Για την ακόμα καλύτερη εκμετάλλευση της τοπικότητας των αναφορών, εφαρμόζεται και τετραγωνικός μετασχηματισμός υπερκόμβων (βλ. Ενότητα 2.5) στους βρόχους που επαναχρησιμοποιούν δεδομένα.



Σχήμα 2.5: Μετάθεση φωλιασμένου βρόχου

Στο Σχήμα 2.4 φαίνεται η αντιστροφή ενός φωλιασμένου βρόχου. Στο συγκεκριμένο παράδειγμα, η αντιστροφή μετέτρεψε το βρόχο σε πλήρως μεταθέσιμο βρόχο (όλα τα στοιχεία του διανύσματος εξάρτησης είναι πλέον θετικά), επομένως οι δύο βρόχοι μπορούν πλέον να

εναλλαχθούν, κάτι που δεν συνέβαινε στον αρχικό βρόχο. Στο Σχήμα 2.5 φαίνεται η μετάθεση (εναλλαγή) ενός φωλιασμένου βρόχου. Παρατηρούμε ότι στον αρχικό χώρο οι επαναλήψεις του εσωτερικότερου βρόχου (j_2) μπορούν να εκτελεστούν παράλληλα (DOALL βρόχος). Επειδή όμως ο j_2 είναι ο εσωτερικότερος βρόχος, σε κάθε επανάληψη του j_1 είναι αναγκαίος συγχρονισμός που επιβαρύνει την εκτέλεση του παραλληλοποιημένου βρόχου [CDK+01]. Εναλλάσσοντας τους δύο βρόχους, παρατηρούμε ότι πλέον ο εξωτερικότερος βρόχος (j'_1) μπορεί να εκτελεστεί παράλληλα, κάτι που σημαίνει ότι κατά την εκτέλεσή του δεν θα υπάρχει ανάγκη συγχρονισμού. Τέλος, στο Σχήμα 2.6 φαίνεται η αλλαγή κλίσης ενός αλγορίθμου. Και εδώ, όπως στην περίπτωση της αντιστροφής, ο νέος βρόχος είναι πλήρως μεταθέσιμος. Μία μετάθεση δηλαδή των δεικτών του νέου βρόχου -που πλέον είναι έγκυρη- θα μετέφερε και σ' αυτή την περίπτωση τον παραλληλισμό στον εξωτερικότερο βρόχο. Άρα με ένα συνδυασμό αλλαγής κλίσης-μετάθεσης μπορεί να προκύψει ένας φωλιασμένος βρόχος που θα εκτελεστεί πολύ αποδοτικά σε μία παράλληλη αρχιτεκτονική.

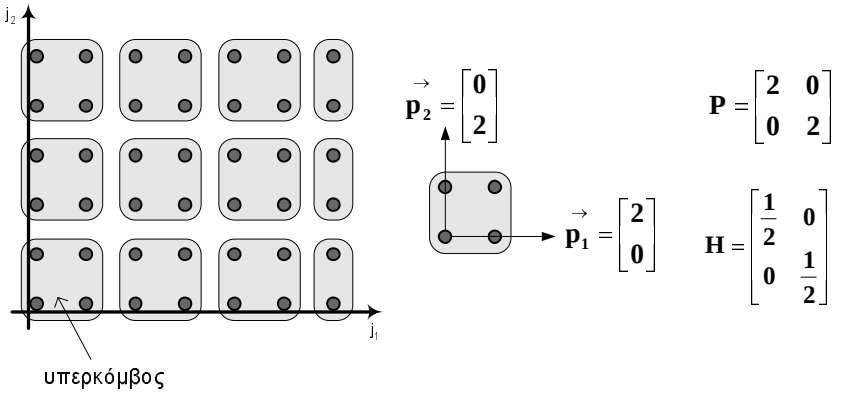


Σχήμα 2.6: Αλλαγή κλίσης φωλιασμένου βρόχου

2.5 Μετασχηματισμός Υπερκόμβων

Οι γραμμικοί μετασχηματισμοί, παρά την ευκολία εφαρμογής τους και την ποικιλία των δυνατοτήτων που παρέχουν, δεν αρκούν πάντα για τον αποδοτικό μετασχηματισμό ενός φωλιασμένου βρόχου. Ένας μεγάλος αριθμός από μη-γραμμικούς μετασχηματισμούς έχει προταθεί στη βιβλιογραφία (*διάσπαση* (distribution), *συνένωση* (fusion), *διαχωρισμός* (fission) κ.ά.)

[KM92] [KM93], [CMT94], [Wol95], [MCT96], [SM97], [KPCM99]. Ο πιο σημαντικός από τους μη-γραμμικούς μετασχηματισμούς είναι ο *μετασχηματισμός υπερκόμβων* (tiling ή supernode transformation) [IT88], [Wol89], [RS92], [Xue97b]. Ο μετασχηματισμός αυτός, ομαδοποιεί κοντινές επαναλήψεις στο χώρο επαναλήψεων ενός φωλιασμένου βρόχου, δημιουργώντας *υπερκόμβους* (tiles ή supernodes), οι οποίοι εκτελούνται αδιαίρετα.



Σχήμα 2.7: Μετασχηματισμός υπερκόμβων σε δισδιάστατο φωλιασμένο βρόχο

Ο μετασχηματισμός υπερκόμβων προτάθηκε από τους Irigoin και Triolet [IT88]. Στην εργασία αυτή, οι υπερκόμβοι προτείνεται να σχηματίζονται από την τομή n παράλληλων υπερεπιπέδων. Ο πίνακας H περιέχει σαν γραμμές κάθετα στα υπερεπίπεδα που σχηματίζουν τους υπερκόμβους διανύσματα. Έτσι, ο πίνακας H ορίζει μονοσήμαντα το σχήμα και το μέγεθος ενός υπερκόμβου. Ο αντίστροφος του πίνακα H περιλαμβάνει σαν στήλες διανύσματα που είναι παράλληλα στις πλευρές του υπερκόμβου και συμβολίζεται με P. Ισχύει $P{=}H^{-1}$. Στο Σχήμα 2.7 φαίνεται ένας μετασχηματισμός που ομαδοποιεί τέσσερα γειτονικά σημεία, καθώς και οι πίνακες H και P που τον ορίζουν.

Τυπικά, ο μετασχηματισμός υπερκόμβων ορίζεται ως εξής:

$$r : Z^n \longrightarrow Z^{2n}, r(\vec{j}) = \begin{bmatrix} \lfloor H\vec{j} \rfloor \\ \vec{j} - H^{-1}\lfloor H\vec{j} \rfloor \end{bmatrix} \tag{2.4}$$

όπου $\lfloor H\vec{j} \rfloor$ είναι οι συντεταγμένες του υπερκόμβου στον οποίο ανήκει το σημείο $\vec{j} \in Z^n$ και $\vec{j} - H^{-1}\lfloor H\vec{j} \rfloor$ είναι οι συντεταγμένες του $\vec{j}$ παίρνοντας σαν αρχή των αξόνων την αρχή του υπερκόμβου. Έτσι, ο αρχικός n-διάστατος χώρος επαναλήψεων J μετασχηματίζεται σε έναν

$2n$-διάστατο, που αποτελείται από το n-διάστατο χώρο των υπερκόμβων και τον επίσης n-διάστατο χώρο των σημείων στο εσωτερικό των υπερκόμβων. Οι παρακάτω χώροι προκύπτουν με την εφαρμογή ενός μετασχηματισμού υπερκόμβων H σε ένα χώρο επαναλήψεων J:

1. Ο *Χώρος Επαναλήψεων Υπερκόμβου* (Tile Iteration Space) $TIS(H) = \{\vec{j} \in Z^n \mid \lfloor H\vec{j} \rfloor = 0\}$, που περιλαμβάνει όλα τα σημεία που ανήκουν στον υπερκόμβο που ξεκινά στην αρχή των αξόνων.

2. Ο *Χώρος Υπερκόμβων* (Tile Space) $J^S(J,H) = \{\vec{j^S} \mid \vec{j^S} = \lfloor H\vec{j} \rfloor,\ \vec{j} \in J\}$, που περιλαμβάνει τις εικόνες όλων των σημείων $\vec{j} \in J$ σύμφωνα με το μετασχηματισμό υπερκόμβων.

3. Ο *Χώρος Αρχών των Υπερκόμβων* (Tile Origin Space) $TOS(J^S, H^{-1}) = \{\vec{j} \in Z^n \mid \vec{j} = H^{-1}\vec{j^S},\ \vec{j^S} \in J^S\}$, που περιέχει τις αρχές των υπερκόμβων στον αρχικό χώρο.

4. Ο *Πίνακας Εξαρτήσεων Υπερκόμβων* $D^S = \{\vec{d^S} \mid \vec{d^S} = \lfloor H(\vec{j} + \vec{d}) \rfloor,\ \vec{d} \in D,\ \vec{j} \in TIS\}$, που περιέχει τις εξαρτήσεις μεταξύ των υπερκόμβων.

Άρα, σύμφωνα με τα παραπάνω, θα ισχύει: $J \xrightarrow{H} J^S$ και $J^S \xrightarrow{P} TOS$. Για λόγους απλότητας, θα αναφερόμαστε στο σύνολο $TIS(H)$ ως TIS, στο σύνολο $J^S(J,H)$ ως J^S και στο $TOS(J^S, H^{-1})$ ως TOS. Αξίζει προσοχής το γεγονός, ότι όλα τα σημεία του J που ανήκουν στον ίδιο υπερκόμβο, απεικονίζονται στο ίδιο σημείο του J^S. Επίσης, ο χώρος TOS δεν είναι απαραίτητα υποσύνολο του J, καθώς είναι δυνατόν να υπάρχουν αρχές υπερκόμβων που δεν ανήκουν στον αρχικό χώρο επαναλήψεων J, αλλά κάποια σημεία των υπερκόμβων αυτών να ανήκουν.

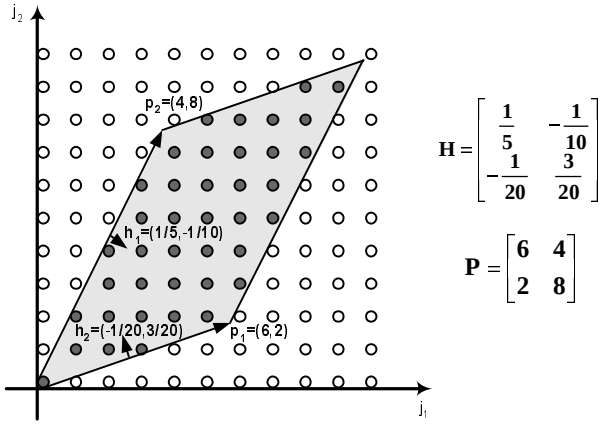
Για να είναι ένας μετασχηματισμός υπερκόμβων έγκυρος, θα πρέπει να ισχύει $HD \geq 0$ [IT88]. Ο αριθμός των σημείων που περιέχονται στο εσωτερικό ενός υπερκόμβου δίνεται από την ορίζουσα του πίνακα P και παρέχει ένα μέτρο για τον όγκο υπολογισμού V_{comp} που θα διεξαχθεί στον υπερκόμβο. Έτσι, ισχύει $V_{comp} = |det(P)| = |1/det(H)|$. Ο όγκος επικοινωνίας ανά υπερκόμβο είναι ανάλογος του αριθμού των σημείων που περιέχονται στον υπερκόμβο και χρειάζονται να αποσταλούν στους γειτονικούς επεξεργαστές. Με άλλα λόγια ο όγκος επικοινωνίας ισούται με το άθροισμα των διανυσμάτων εξάρτησης που τέμνουν τα όρια του υπερκόμβου. Αυτό υπολογίζεται από την έκφραση:

$$V_{comm}(H) = \frac{1}{|\det(H)|} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{m} h_{ik} d_{kj} \tag{2.5}$$

Πρακτικά η παραπάνω σχέση υπολογίζει και αθροίζει όλα τα $\vec{h_i}\vec{d_j}$, που εκφράζουν τη συνεισφορά κάθε διανύσματος εξάρτησης σε κάθε οριακή επιφάνεια του υπερκόμβου. Αν οι υπερκόμβοι κατά μήκος μίας διάστασης ανήκουν στον ίδιο επεξεργαστή, τα διανύσματα εξάρτησης που τέμνουν τη διάσταση αυτή δεν προκαλούν επικοινωνία. Σ' αυτή την περίπτωση ο όγκος επικοινωνίας ανά υπερκόμβο δίνεται από τη σχέση:

$$V_{comm}(H) = \frac{1}{|\det(H)|} \sum_{i \in \{1,\ldots,x-1,x+1,\ldots,n\}, j \in \{1,\ldots,m\}} (H_{-x}D)_{ij'} \tag{2.6}$$

όπου H_{-x} είναι ο πίνακας H χωρίς το διάνυσμα-στήλη που είναι παράλληλο προς τη διάσταση απεικόνισης.



Σχήμα 2.8: Χώρος Επαναλήψεων Υπερκόμβου (TIS)

Παράδειγμα 2.3 Έστω ο παρακάτω φωλιασμένος βρόχος:

```
FOR (j1=0; j1 ≤39; j1++)
   FOR (j2=0; j2 ≤29; j2++)
      A[j1, j2]=A[j1 − 1, j2 − 2]+A[j1 − 3, j2 − 1];
   ENDFOR
ENDFOR
```

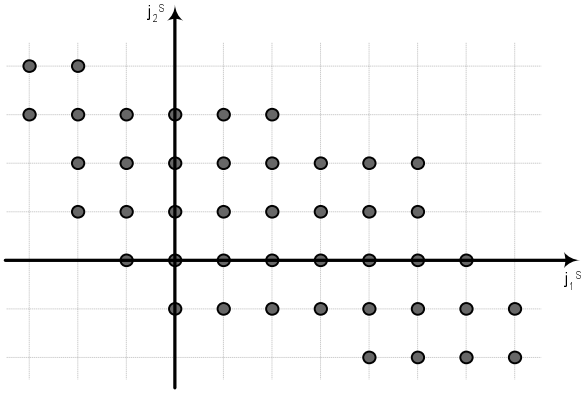
Ο αντίστοιχος χώρος δεικτών J είναι: $J = \{(j_1, j_2) \mid 0 \leq j_1 \leq 39, 0 \leq j_2 \leq 29\}$ και ο πίνακας

εξαρτήσεων του συγκεκριμένου αλγορίθμου είναι $D = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$. Εφαρμόζουμε στον παραπάνω χώρο το μετασχηματισμό υπερκόμβων που ορίζεται από τον πίνακα $H = \begin{bmatrix} \frac{1}{5} & -\frac{1}{10} \\ -\frac{1}{20} & \frac{3}{20} \end{bmatrix}$ ή, ισοδύναμα, από τον πίνακα $P = \begin{bmatrix} 6 & 4 \\ 2 & 8 \end{bmatrix}$. Ο μετασχηματισμός αυτός είναι έγκυρος, αφού ισχύει $HD \geq 0$. Τότε, όπως φαίνεται στο Σχήμα 2.8, ο Χώρος Επαναλήψεων Υπερκόμβου TIS περιέχει τα σημεία $\{(0,0), (1,1), (1,2), (2,1), (2,2), (2,3), (2,4), \ldots, (7,5), (7,6), (7,7), (7,8), (8,7), (8,8), (8,9), (9,9)\}$. Επιπρόσθετα, όπως φαίνεται στο Σχήμα 2.9, ο J μετασχηματίζεται από τον πίνακα H στο Χώρο Υπερκόμβων $J^S = \{(-3,3), (-3,4), (-2,1), (-2,2), (-2,3), (-2,4), \ldots, (6,-2), (6,-1), (6,0), (7,-2), (7,-1)\}$. Στη συνέχεια, όπως φαίνεται από τα γκρίζα σημεία του Σχήματος 2.10, ο Χώρος Υπερκόμβων J^S μετασχηματίζεται από τον πίνακα P στο Χώρο Αρχών Υπερκόμβων $TOS = \{(-6,18), (-2,26), (-8,4), (-4,12), (0,20), (4,28), \ldots, (28,-4), (32,-4), (36,12), (34,-2), (38,6)\}$. ⊣

Τα σημεία που ανήκουν στον ίδιο υπερκόμβο με αρχή το σημείο $\vec{j_0} \in TOS$, επαληθεύουν το σύστημα ανισοτήτων $\vec{0} \leq H(\vec{j} - \vec{j_0}) < \vec{1}$. Προκειμένου να χειριζόμαστε ανισότητες ακεραίων, ορίζουμε το g ως το μικρότερο ακέραιο τέτοιο ώστε ο gH να είναι ακέραιος πίνακας. Έτσι, μπορούμε να γράψουμε το παραπάνω σύστημα ανισοτήτων ως: $\vec{0} \leq gH(\vec{j} - \vec{j_0}) < \vec{g} \Leftrightarrow \vec{0} \leq gH(\vec{j} - \vec{j_0}) \leq (\vec{g} - \vec{1})$. Συμβολίζουμε $S = \begin{pmatrix} gH \\ -gH \end{pmatrix}$ και $\vec{s} = \begin{pmatrix} (g-1)\vec{1} \\ \vec{0} \end{pmatrix}$. Ισοδύναμα, το παραπάνω σύστημα γίνεται:

$$S(\vec{j} - \vec{j_0}) \leq \vec{s} \tag{2.7}$$

Παράδειγμα 2.4 Το σύστημα ανισοτήτων που περιγράφει το χώρο δεικτών J στο φωλιασμένο



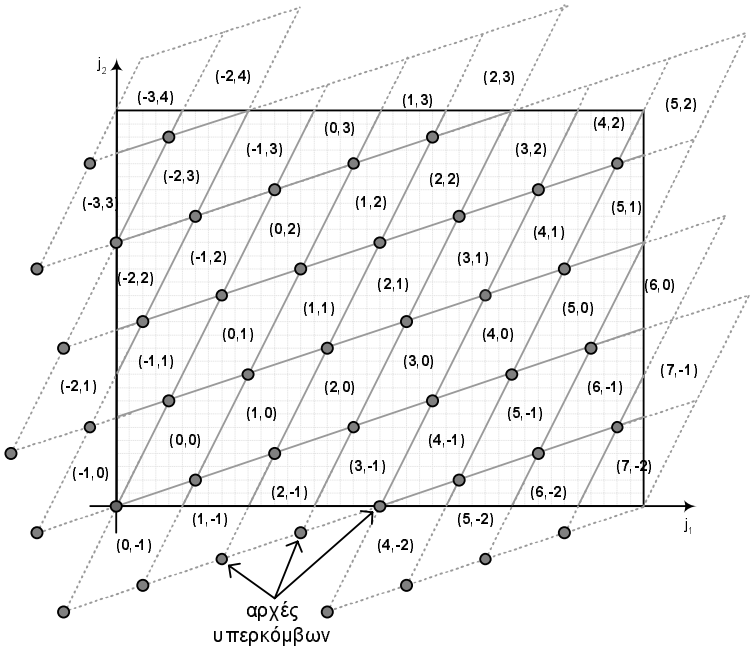
Σχήμα 2.9: Χώρος Υπερκόμβων (J^S)

βρόχο του Παραδείγματος 2.3, είναι το παρακάτω:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 39 \\ 29 \\ 0 \\ 0 \end{pmatrix}$$

Χρησιμοποιώντας τον ίδιο πίνακα μετασχηματισμού υπερκόμβων $H = \begin{bmatrix} \frac{1}{5} & -\frac{1}{10} \\ -\frac{1}{20} & \frac{3}{20} \end{bmatrix}$, το σύστημα ανισοτήτων $S(\vec{j} - \vec{j_0}) \leq \vec{s}$, που επαληθεύεται από τα εσωτερικά σημεία του υπερκόμβου με αρχή το $\vec{j_0} = (j_{0_1}, j_{0_2})$ είναι ($g = 20$): $\begin{pmatrix} 4 & -2 \\ -1 & 3 \\ -4 & 2 \\ 1 & -3 \end{pmatrix} \begin{pmatrix} j_1 - j_{0_1} \\ j_2 - j_{0_2} \end{pmatrix} \leq \begin{pmatrix} 19 \\ 19 \\ 0 \\ 0 \end{pmatrix}$. ⊣

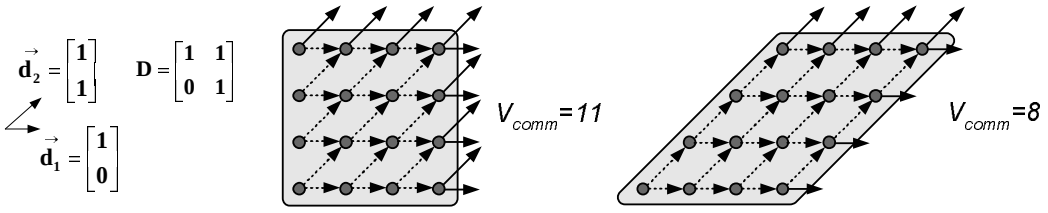
Ο μετασχηματισμός υπερκόμβων έχει χρησιμοποιηθεί κατά κόρον σε ερευνητικούς και εμπορικούς μεταγλωττιστές, για τη βελτιστοποίηση της χρήσης της λανθάνουσας μνήμης μέσω της βελτίωσης της τοπικότητας των αναφορών (reference locality) ενός προγράμματος [WL91a], [LRW91], [Li94], [CM99]. Σ' αυτή την περίπτωση, χρησιμοποιούνται αποκλειστικά τετραγωνικοί μετασχηματισμοί υπερκόμβων, ώστε οι υπολογισμοί να ολοκληρώνονται εσωτερικά σε έναν υπερκόμβο και εν συνεχεία να μεταφέρονται στον επόμενο υπερκόμβο κ.ο.κ. Ιδανικά, τα δεδομένα που προσπελαύνει ένας υπερκόμβος χωρούν στη λανθάνουσα μνήμη του συστήματος και έτσι τυχόν επαναχρησιμοποιήσεις των ίδιων δεδομένων θα οδηγήσουν σε προσπέλασή τους από τη λανθάνουσα μνήμη και όχι από την κύρια μνήμη. Το γεγονός αυτό εξοικονομεί πολύ μεγάλο αριθμό ωφέλιμων κύκλων επεξεργαστή [HP95] (σελ. 373-375).



Σχήμα 2.10: Μετασχηματισμός υπερκόμβων στο Χώρο Επαναλήψεων του Παραδείγματος 2.3

Ιδιαίτερο ενδιαφέρον όμως παρουσιάζει η εφαρμογή του μετασχηματισμού υπερκόμβων σε φωλιασμένους βρόχους, που πρόκειται να εκτελεστούν σε παράλληλες αρχιτεκτονικές, είτε μοιραζόμενης είτε κατανεμημένης μνήμης. Στην ειδική κατηγορία προβλημάτων που έχουν n γραμμικά ανεξάρτητα διανύσματα εξάρτησης (`DOACROSS` βρόχοι), δεν είναι δυνατόν να μετασχηματίσουμε το βρόχο ώστε να έχει εξωτερικούς `DOALL` βρόχους και να δημιουργήσουμε έτσι τελείως παράλληλες και χωρίς συγχρονισμό εκτελέσεις [WL91b] (βλ. Ενότητα 2.4). Σ' αυτές τις περιπτώσεις χρησιμοποιείται ο μετασχηματισμός υπερκόμβων για να μειώσει τη συχνότητα συγχρονισμού στις αρχιτεκτονικές μοιραζόμενης μνήμης και τη συχνότητα επικοινωνίας στις αρχιτεκτονικές κατανεμημένης μνήμης. Η εκτέλεση βρόχων που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων σε παράλληλες αρχιτεκτονικές πραγματοποιείται ως εξής: κάθε επεξεργαστής λαμβάνει τα δεδομένα που χρειάζεται για τη διεξαγωγή υπολογισμών σε έναν υπερκόμβο, εκτελεί τους σχετικούς υπολογισμούς και στη συνέχεια στέλνει τα δεδομένα που μόλις υπολόγισε στους γειτονικούς επεξεργαστές για τους περαιτέρω υπολογισμούς τους.

Όπως αναφέρθηκε και στην Ενότητα 1.3, στην περίπτωση που χρησιμοποιείται μετασχηματισμός υπερκόμβων για την παράλληλη εκτέλεση φωλιασμένων βρόχων, είναι δυνατόν να επιλεγούν γενικοί παραλληλεπίπεδοι μετασχηματισμοί H και όχι απαραίτητα ορθογώνιοι, όπως



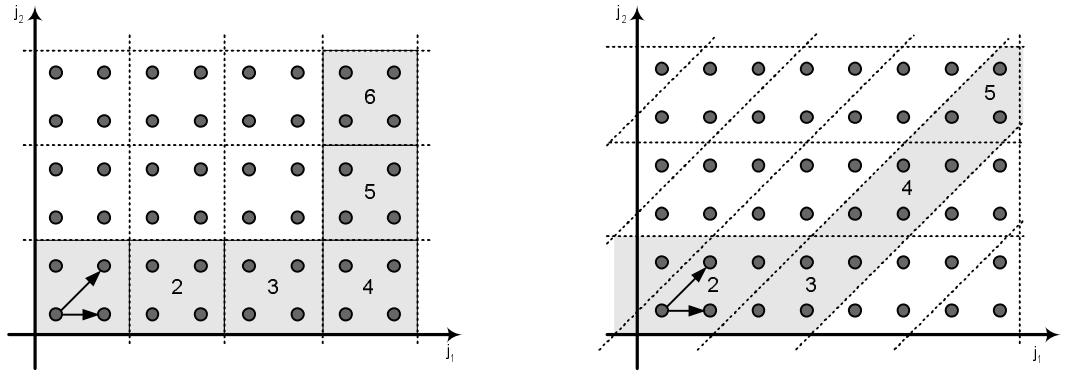
Σχήμα 2.11: Διαφορά στον όγκο επικοινωνίας ανάμεσα σε έναν ορθογώνιο και σε ένα μη-ορθογώνιο μετασχηματισμό υπερκόμβων

συμβαίνει για την αξιοποίηση της τοπικότητας των αναφορών. Στο Σχήμα 2.11 φαίνεται η επίδραση του σχήματος του υπερκόμβου στον όγκο επικοινωνίας. Αν θεωρήσουμε ότι απαιτείται επικοινωνία για κάθε διάνυσμα εξάρτησης που τέμνει τις οριακές επιφάνειες ενός υπερκόμβου, τότε στο Σχήμα 2.11 φαίνεται ότι για το συγκεκριμένο παράδειγμα οκτώ διανύσματα εξάρτησης τέμνουν της οριακή επιφάνεια του υπερκόμβου στην περίπτωση του μη-ορθογώνιου μετασχηματισμού υπερκόμβων, ενώ ένδεκα διανύσματα εξάρτησης τέμνουν τις οριακές επιφάνειες στην περίπτωση του ορθογώνιου μετασχηματισμού [RS92], [BDRR94], [Xue97a].

Ακόμα πιο σημαντική όμως για την επίδοση ενός αλγορίθμου, είναι η επίδραση του σχήματος του υπερκόμβου στο συνολικό χρόνο εκτέλεσής του. Στο Σχήμα 2.12 φαίνεται απλοποιημένα το γεγονός αυτό. Αν αγνοήσουμε το κόστος επικοινωνίας και θεωρήσουμε πως οι υπερκόμβοι κατά μήκος της διάστασης j_1 εκτελούνται στον ίδιο επεξεργαστή, τότε παρατηρούμε πως στην περίπτωση του ορθογώνιου μετασχηματισμού απαιτούνται έξι βήματα για την περάτωση του αλγορίθμου, ενώ αντίθετα στην περίπτωση του μη-ορθογώνιου μετασχηματισμού απαιτούνται πέντε. Η επίδραση του σχήματος του μετασχηματισμού υπερκόμβων στο συνολικό χρόνο εκτέλεσης ενός φωλιασμένου βρόχου μελετάται στις εργασίες [HS02] και [HCF03].

2.6 Παραγωγή Κώδικα για Μετασχηματισμένους Φωλιασμένους Βρόχους

Μετά την επιλογή του κατάλληλου μετασχηματισμού για τη βέλτιστη απεικόνιση ενός φωλιασμένου βρόχου σε κάποια αρχιτεκτονική υψηλής επίδοσης, ακολουθεί η διαδικασία μετασχηματισμού του αρχικού κώδικα στον ισοδύναμο μετασχηματισμένο. Η διαδικασία αυτή πρέπει να είναι απλή, ώστε να μην καθιστά το χρόνο μεταγλώττισης απαγορευτικό, αλλά κυρίως πρέπει να



Σχήμα 2.12: Διαφορά στο συνολικό χρόνο εκτέλεσης ανάμεσα σε έναν ορθογώνιο και σε ένα μη-ορθογώνιο μετασχηματισμό υπερκόμβων

οδηγεί σε όσο το δυνατόν αποδοτικότερο μετασχηματισμένο κώδικα. Κάθε μετασχηματισμός, όπως θα φανεί και στη συνέχεια, εισφέρει κάποιους επιπλέον υπολογισμούς που επιβαρύνουν την εκτέλεση του αλγορίθμου. Προφανώς, αναμένεται ότι με την εφαρμογή του μετασχηματισμού, τα οφέλη θα είναι πολύ μεγαλύτερα, ώστε να ξεπερνούν κατά πολύ την επιβάρυνση που επιφέρει ο ίδιος ο μετασχηματισμός. Στην ενότητα αυτή θα παρουσιάσουμε μεθόδους παραγωγής κώδικα στην περίπτωση των ορθομοναδιαίων και μη-ορθομοναδιαίων μετασχηματισμών και των μετασχηματισμών υπερκόμβων.

2.6.1 Ορθομοναδιαίοι Μετασχηματισμοί

Προκειμένου να παράγουμε κώδικα για μετασχηματισμένους χώρους στην περίπτωση των ορθομοναδιαίων μετασχηματισμών, πρέπει να δημιουργήσουμε `FOR LOOPS` που θα διατρέχουν το νέο χώρο με λεξικογραφική σειρά, και να αντικαταστήσουμε τις μεταβλητές ελέγχου του βρόχου με τις νέες μετασχηματισμένες, όπου αυτές εμφανίζονται. Έτσι, σ' αυτή την περίπτωση, η διαδικασία είναι σχετικά απλή και ολοκληρώνεται σε δύο βήματα:

- **Βήμα 1:** *Υπολογισμός των ορίων του μετασχηματισμένου φωλιασμένου βρόχου.*
  Έστω T ο ορθομοναδιαίος μετασχηματισμός που εφαρμόζεται σε ένα n-διάστατα φωλιασμένο βρόχο, του οποίου τα όρια περιγράφονται από το σύστημα ανισοτήτων $B\vec{j} \leq \vec{b},\ \vec{j} \in J$. Τότε θα ισχύει, $B\vec{j} \leq \vec{b} \Rightarrow BT^{-1}T\vec{j} \leq \vec{b} \Rightarrow BT^{-1}\vec{j'} \leq \vec{b},\ \vec{j'} \in J'$. Το νέο σύστημα $BT^{-1}\vec{j'} \leq \vec{b}$ περιγράφει τα όρια του μετασχηματισμένου χώρου. Για να μετατραπεί όμως το σύστημα αυτό σε κατάλληλη μορφή, ώστε ο νέος χώρος να μπορεί να διασχιστεί με

τη βοήθεια ενός φωλιασμένου βρόχου, θα πρέπει να εφαρμοστεί η μέθοδος απαλοιφής Fourier-Motzkin Elimination (FME) (βλ. Παράρτημα Α). Το νέο σύστημα μετά την απαλοιφή συμβολίζεται $B'T^{-1}\vec{j'} \leq \vec{b'}$ και έχει την πολύ χρήσιμη ιδιότητα, ότι κάθε ανισότητα που περιλαμβάνει τη μεταβλητή j'_k, δεν περιλαμβάνει μεταβλητές $j'_{k+1} \ldots j'_n$, κάτι που σημαίνει ότι πλέον ο χώρος μπορεί να διασχιστεί από ενα n-διάστατο φωλιασμένο βρόχο.

- **Βήμα 2:** *Αντικατάσταση των μεταβλητών ελέγχου στο σώμα του βρόχου.*
  Το βήμα αυτό είναι ιδιαίτερα απλό καθώς κάθε εμφάνιση της μεταβλητής $\vec{j}$ αντικαθίσταται με βάση το μετασχηματισμό T, δηλ. $\vec{j} = T^{-1}\vec{j'}$.

Παράδειγμα 2.5 Στο παράδειγμα αυτό θα δείξουμε πως προκύπτει ο μετασχηματισμένος κώδικας για τους φωλιασμένους βρόχους των Σχημάτων 2.4-2.6.

- Για το Σχήμα 2.4 έχουμε τα εξής:

  Βήμα 1: Το σύστημα που περιγράφει το χώρο επαναλήψεων είναι $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$ ενώ ο μετασχηματισμός ορίζεται από τον πίνακα $T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = T^{-1}$. Έτσι, ο μετασχηματισμένος χώρος θα περιγράφεται από το σύστημα $\begin{pmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} j'_1 \\ j'_2 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$. Άρα $0 \leq j'_1 \leq 2$ και $-3 \leq j'_2 \leq 0$.

  Βήμα 2: Οι δείκτες j_1, j_2 αντικαθίστανται από τους j'_1, j'_2 με βάση τη σχέση $\vec{j} = T^{-1}\vec{j'}$ που δίνει $j_1 = j'_1$ και $j_2 = -j'_2$.

- Για το Σχήμα 2.5 έχουμε:

Βήμα 1: Το σύστημα που περιγράφει το χώρο επαναλήψεων είναι το ίδιο με την προηγούμενη περίπτωση, ενώ ο μετασχηματισμός ορίζεται από τον πίνακα

$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = T^{-1}$. Έτσι, ο μετασχηματισμένος χώρος θα περιγράφεται από το σύστημα $\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} j'_1 \\ j'_2 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$. Άρα $0 \leq j'_1 \leq 3$ και $0 \leq j'_2 \leq 2$.

Βήμα 2: Οι δείκτες j_1, j_2 αντικαθίστανται από τους j'_1, j'_2 με βάση τη σχέση $\vec{j} = T^{-1}\vec{j'}$ που δίνει $j_1 = j'_2$ και $j_2 = j'_1$.

- Για το Σχήμα 2.6 έχουμε:

  Βήμα 1: Ο χώρος επαναλήψεων παραμένει και εδώ ο ίδιος, ενώ ο μετασχηματισμός ορίζεται από τον πίνακα

  $T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \Rightarrow T^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$. Έτσι, ο μετασχηματισμένος χώρος θα περιγράφεται από το σύστημα $\begin{pmatrix} 1 & 0 \\ -1 & 1 \\ -1 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$. Άρα $0 \leq j'_1 \leq 2$ και $j'_1 \leq j'_2 \leq 3 + j'_1$.

  Βήμα 2: Οι δείκτες j_1, j_2 αντικαθίστανται από τους j'_1, j'_2 με βάση τη σχέση $\vec{j} = T^{-1}\vec{j'}$ που δίνει $j_1 = j'_1$ και $j_2 = j'_2 - j'_1$.

Αξίζει να σημειωθεί ότι σε καμία από τις παραπάνω περιπτώσεις δεν ήταν απαραίτητη η εφαρμογή της μεθόδου FME, καθώς τα συστήματα που προέκυψαν ήταν όλα σε μορφή κατάλληλη για χρήση σε φωλιασμένους βρόχους. ⊣

2.6.2 Μη-Ορθομοναδιαίοι Μετασχηματισμοί

Σ' αυτή την περίπτωση η διαδικασία παραγωγής κώδικα είναι πιο περίπλοκη. Όπως αναφέρθηκε και στην Ενότητα 2.4, όταν εφαρμόζεται ένας μη-ορθομοναδιαίος μετασχηματισμός, ο μετασχηματισμένος χώρος είναι αραιός, ή αλλιώς περιέχει οπές, δηλαδή σημεία τα οποία δεν πρέπει να προσπελαστούν για να διατηρηθεί η ορθότητα και η αποδοτικότητα του προγράμματος. Πολλοί ερευνητές έχουν ασχοληθεί με την παραγωγή κώδικα για μη-ορθομοναδιαία μετασχηματισμένους χώρους επαναλήψεων [Ram92], [Ram95], [Xue94], [Li93], [FLV95]. Όλες οι παραπάνω μέθοδοι στηρίζονται στη χρήση της Ερμητιανής Κανονικής Μορφής $\widetilde{T}$ του πίνακα T. Η μέθοδος που παρουσιάζεται εδώ προτάθηκε από τους Li και Pingali [LP94] και στηρίζεται στη χρήση ενός ενδιάμεσου *βοηθητικού* χώρου. Με βάση το Θεώρημα 2.2, ο πίνακας T μπορεί να γραφεί ως $T = \widetilde{T}U$, όπου U είναι ένας ορθομοναδιαίος πίνακας. Ο βοηθητικός χώρος J^u, είναι αυτός που προκύπτει από την εφαρμογή του μετασχηματισμού U στον αρχικό χώρο. Άρα ισχύει $J \xrightarrow{U} J^u$ και $J^u \xrightarrow{\widetilde{T}} J'$. Η γενική ιδέα της μεθόδου στηρίζεται στην εύρεση των ορίων του βοηθητικού χώρου και στην κατάλληλη τροποποίησή τους για τον υπολογισμό των ορίων του τελικού χώρου.

- **Βήμα 1:** *Υπολογισμός των ορίων του βοηθητικού χώρου.*
  Η διαδικασία είναι προφανώς η ίδια που περιγράφηκε στην Ενότητα 2.6.1 (Βήμα 1), αφού, όπως αναφέρθηκε, ο U είναι ορθομοναδιαίος μετασχηματισμός.

- **Βήμα 2:** *Υπολογισμός των ορίων του τελικού χώρου.*
  Αν l_k^u και u_k^u είναι το άνω και κάτω όριο μίας μεταβλητής ελέγχου στο βοηθητικό χώρο όπως υπολογίστηκαν στο προηγούμενο βήμα, το αντίστοιχο άνω και κάτω όριο στον τελικό χώρο θα είναι: $l'_k = v_k + l_k^u \widetilde{t}_{kk}$ και $u'_k = v_k + u_k^u \widetilde{t}_{kk}$. Οι διορθωτικοί παράγοντες v_k προσδιορίζονται από τη σχέση $v_k = \widetilde{t}_{k1} j_1^u + \widetilde{t}_{k2} j_2^u + \ldots + \widetilde{t}_{k(k-1)} j_{k-1}^u$, ενώ οι μεταβλητές $\vec{j^u} \in J^u$ αντικαθίστανται από τη σχέση $\vec{j^u} = \widetilde{T}^{-1} \vec{j'}$.

- **Βήμα 3:** *Υπολογισμός των βημάτων.*
  Προκειμένου να αποφύγουμε τη διάσχιση των οπών στο μετασχηματισμένο χώρο, ο φωλιασμένος βρόχος που θα τον διατρέξει θα πρέπει να έχει μη μοναδιαίο βήμα. Τα βήματα αυτά $c_1, c_2, \ldots, c_n$ προσδιορίζονται άμεσα από την ΕΚΜ του πίνακα T. Ισχύει, $c_k = \widetilde{t}_{kk}$.

- **Βήμα 4:** *Αντικατάσταση των μεταβλητών ελέγχου.*
  Όμοια με πριν, κάθε εμφάνιση της μεταβλητής $\vec{j}$ αντικαθίσταται με βάση το μετασχηματισμό T, δηλ. $\vec{j} = T^{-1} \vec{j'}$.

Παράδειγμα 2.6 Έστω ο φωλιασμένος βρόχος του Σχήματος 2.4, που επαναλαμβάνεται εδώ για λόγους ευκολίας:

```
FOR (j1=0; j1 ≤3; j1++)
   FOR (j2=0; j2 ≤4; j2++)
      A[j1,j2]=A[j1-1,j2];
   ENDFOR
ENDFOR
```

Εφαρμόζουμε στο βρόχο αυτό τον μη-ορθομοναδιαίο μετασχηματισμό $T = \begin{bmatrix} 1 & 0 \\ 3 & 3 \end{bmatrix}$ ($det(T) = 3$). Για την παραγωγή του κώδικα του μετασχηματισμένου φωλιασμένου βρόχου, αναλύουμε τον T στο γινόμενο ενός ορθομοναδιαίου πίνακα και ενός πίνακα σε ΕΚΜ, δηλ. $T = \widetilde{T}U$, οπότε έχουμε $\begin{bmatrix} 1 & 0 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{3} & 1 \end{bmatrix}$. Άρα $\widetilde{T} = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$ και $U = \begin{bmatrix} 1 & 0 \\ \frac{1}{3} & 1 \end{bmatrix}$.

Βήμα 1: Αφού $U^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{3} & 1 \end{bmatrix}$, ο βοηθητικός χώρος θα περιγράφεται από το σύστημα $\begin{pmatrix} 1 & 0 \\ -\frac{1}{3} & 1 \\ -1 & 0 \\ \frac{1}{3} & -1 \end{pmatrix} \begin{pmatrix} j_1^u \\ j_2^u \end{pmatrix} \leq \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix}$.

Άρα τα όρια του χώρου αυτού θα είναι $l_1^u = 0$, $l_2^u = \frac{1}{3}j_1^u$, $u_1^u = 3$ και $u_2^u = 4 + \frac{1}{3}j_1^u$.

Βήμα 2: Ισχύει $v_2 = \widetilde{t}_{21}j_1^u = 2j_1^u$. Έτσι, τα όρια του τελικού χώρου θα είναι $l_1' = l_1^u\widetilde{t}_{11} = 0 \times 1 = 0$, $l_2' = v_2 + l_2^u \times 3 = 2j_1^u + 1j_1^u = 3j_1^u$, $u_1' = 3$ και $u_2' = v_2 + \widetilde{t}_{22}u_2^u = 2j_1^u + 3(4 + \frac{1}{3}j_1^u) = 3j_1^u + 12$. Ισχύει επίσης $\widetilde{T}^{-1} = \frac{1}{3}\begin{bmatrix} 3 & 0 \\ -2 & 1 \end{bmatrix}$, επομένως $j_1^u = j_1'$ και τελικά παραπάνω εκφράσεις γίνονται $l_1' = 0$, $l_2' = 5j_1'$, $u_1' = 3$ και $u_2' = 3j_1' + 12$.

Βήμα 3: Τα βήματα των βρόχων δίνονται από τη διαγώνιο του πίνακα $\widetilde{T}$, δηλ. $c_1 = \widetilde{t}_{11} = 1$ και $c_2 = \widetilde{t}_{22} = 3$.

Βήμα 4: Ισχύει $T^{-1} = \frac{1}{3}\begin{bmatrix} 3 & 0 \\ -3 & 1 \end{bmatrix}$, επομένως $j_1 = j'_1$ και $j_2 = -j'_1 + \frac{1}{3}j'_2$.

Ο τελικός μετασχηματισμένος κώδικας είναι ως εξής:

```
FOR (j'_1=0; j'_1 ≤3; j'_1++)
   FOR (j'_2=3*j'_1; j'_2 ≤12+3*j'_1; j'_2+=3)
      A[j'_1,-j'_1+1/3 j'_2]=A[j'_1-1,-j'_1+1/3 j'_2+1];
   ENDFOR
ENDFOR                                                    ⊣
```

2.6.3 Μετασχηματισμοί Υπερκόμβων

Σ' αυτή την ενότητα θα αναφερθούμε στην παραγωγή κώδικα για χώρους επαναλήψεων που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Ονομάζουμε αυτό τον κώδικα *σειριακό μετασχηματισμένο κώδικα*. Με την εφαρμογή ενός μετασχηματισμού υπερκόμβων H σε ένα χώρο J, προκύπτουν ο Χώρος Υπερκόμβων J^S, ο Χώρος Επαναλήψεων Υπερκόμβου (TIS) και ο Χώρος Αρχών των Υπερκόμβων (TOS). Στην Ενότητα 2.5 δείξαμε ότι ο μετασχηματισμός υπερκόμβων είναι ένας μετασχηματισμός $Z^n \longrightarrow Z^{2n}$, κάτι που σημαίνει ότι ένα σημείο $\vec{j} \in J$ μετασχηματίζεται σε μια δυάδα n-διάστατων παραγόντων $(\vec{j_a}, \vec{j_b})$, όπου το $\vec{j_a}$ προσδιορίζει τον υπερκόμβο στον οποίο ανήκει το σημείο ($\vec{j_a} \in J^S$), και το $\vec{j_b}$ καθορίζει τις συντεταγμένες του σημείου ως προς την αρχή του υπερκόμβου ($\vec{j_b} \in TIS$). Ο σειριακός μετασχηματισμένος κώδικας αναδιατάσσει τη σειρά διάσχισης των βρόχων όπως αυτή επιβάλλεται από το αρχικό πρόγραμμα, η οποία καταλήγει να έχει τη μορφή: `ΓΙΑ (ΚΑΘΕ υπερκόμβο στο Χώρο Υπερκόμβων` J^S`) ΔΙΕΣΧΙΣΕ ΤΑ ΣΗΜΕΙΑ ΣΤΟ ΕΣΩΤΕΡΙΚΟ ΤΟΥ ΥΠΕΡΚΟΜΒΟΥ`. Σύμφωνα με τα παραπάνω, ο σειριακός μετασχηματισμένος κώδικας θα αποτελείται από $2n$ φωλιασμένους βρόχους. Οι n εξωτερικότεροι διασχίζουν το Χώρο Υπερκόμβων J^S, χρησιμοποιώντας τις μεταβλητές ελέγχου $t_1^S, t_2^S, \ldots, t_n^S$ και οι n εσωτερικότεροι προσπελαύνουν τα σημεία στο εσωτερικό του τρέχοντος υπερκόμβου που υπαγορεύουν τα $t_1^S, t_2^S, \ldots, t_n^S$, χρησιμοποιώντας τις μεταβλητές ελέγχου $j'_1, j'_2, \ldots, j'_n$. Συμβολίζουμε με l_k^S, u_k^S τα κάτω και άνω όρια αντίστοιχα της μεταβλητής ελέγχου t_k^S. Όμοια, συμβολίζουμε με l'_k, u'_k τα κάτω και άνω όρια αντίστοιχα της μεταβλητής ελέγχου j'_k. Σε όλες τις περιπτώσεις τα κάτω όρια L_k είναι της μορφής $L_k = max(l_{k,0}, l_{k,1}, \ldots)$ και τα άνω όρια της μορφής $U_k = min(u_{k,0}, u_{k,1}, \ldots)$, όπου $l_{k,j}$, $u_{k,j}$ είναι γραμμικές συναρτήσεις των εξωτερικότερων μεταβλητών ελέγχου. Αν υπολογίσουμε τους

παράγοντες $l_1^S, \ldots, l_n^S$, $u_1^S, \ldots, u_n^S$, $l'_1, \ldots, l'_n$ και $u'_1, \ldots, u'_n$, τότε θα διασχίσουμε το μετασχηματισμένο χώρο όπως υπαγορεύει ο μετασχηματισμός υπερκόμβων.

Το παραπάνω πρόβλημα μπορεί να διαχωριστεί σε δύο υποπροβλήματα: τη διάσχιση του Χώρου Υπερκόμβων J^S και την προσπέλαση των σημείων στο εσωτερικό των υπερκόμβων, ή, σύμφωνα με όσα αναφέρθηκαν παραπάνω, στον υπολογισμό των κάτω και άνω ορίων των n εξωτερικότερων μεταβλητών ελέγχου $t_1^S, t_2^S, \ldots, t_n^S$ και των κάτω και άνω ορίων των n εσωτερικότερων μεταβλητών ελέγχου $j'_1, j'_2, \ldots, j'_n$. Οι Ancourt και Irigoin [AI91] έδωσαν μία λύση στο συγκεκριμένο πρόβλημα, κατασκευάζοντας ένα σύστημα ανισοτήτων για κάθε υποπρόβλημα. Προκειμένου να διασχίσουν το Χώρο Υπερκόμβων J^S, δημιουργείται το πρώτο σύστημα, με τη συνένωση των ανισοτήτων που περιγράφουν τον αρχικό χώρο δεικτών και των ανισοτήτων που περιγράφουν το μετασχηματισμό υπερκόμβων (Ανισότητα (2.7)). Από την Ενότητα 2.5 γνωρίζουμε ότι ένα σημείο $\vec{j} \in J$ που ανήκει σε έναν υπερκόμβο με αρχή το $\vec{j_0} \in TOS$, επαληθεύει το σύστημα ανισοτήτων $S(\vec{j} - \vec{j_0}) \leq \vec{s}$. Ισχύει επίσης $\vec{j_0} = P\vec{j^S}$, επομένως το προηγούμενο σύστημα γίνεται: $\begin{pmatrix} -gI & gH \\ gI & -gH \end{pmatrix} \begin{pmatrix} \vec{j^S} \\ \vec{j} \end{pmatrix} \leq \vec{s}$. Επιπρόσθετα, το σημείο $\vec{j} \in J$ επαληθεύει το σύστημα ανισοτήτων που περιγράφει τον αρχικό χώρο, $B\vec{j} \leq \vec{b}$. Ο συνδυασμός των δύο αυτών συστημάτων δίνει:

$$\begin{pmatrix} 0 & B \\ -gI & gH \\ gI & -gH \end{pmatrix} \begin{pmatrix} \vec{j^S} \\ \vec{j} \end{pmatrix} \leq \begin{pmatrix} \vec{b} \\ \vec{s} \end{pmatrix} \tag{2.8}$$

Προκειμένου να διασχίσουμε τα εσωτερικά σημεία των υπερκόμβων, το παραπάνω σύστημα μπορεί να γραφεί με τη μορφή:

$$\begin{pmatrix} B \\ gH \\ -gH \end{pmatrix} \vec{j} \leq \begin{pmatrix} \vec{b} \\ (g-1)\vec{1} + g\vec{j^S} \\ \vec{0} - g\vec{j^S} \end{pmatrix} \tag{2.9}$$

όπου το διάνυσμα $\vec{j^S}$ δίνει τις συντεταγμένες του υπερκόμβου του οποίου τα σημεία θα προσπελαστούν. Οι Ancourt και Irigoin προτείνουν τη χρήση της μεθόδου FME στα παραπάνω συστήματα για την εξαγωγή εκφράσεων που θα επιτρέπουν τη διάσχιση του μετασχηματισμένου φωλιασμένου βρόχου με ένα $2n$-διάστατο φωλιασμένο βρόχο.

Παράδειγμα 2.7 Θα δείξουμε πώς με βάση τη μέθοδο που παρουσιάστηκε παραπάνω, μπορεί να παραχθεί κώδικας για το φωλιασμένο βρόχο και το μετασχηματισμό υπερκόμβων του Παραδείγματος 2.3. Το σύστημα ανισοτήτων (2.8) σ' αυτή την περίπτωση γίνεται:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -20 & 0 & 4 & -2 \\ 0 & -20 & -1 & 3 \\ 20 & 0 & -4 & 2 \\ 0 & 20 & 1 & -3 \end{pmatrix} \begin{pmatrix} j_1^S \\ j_2^S \\ j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 39 \\ 29 \\ 0 \\ 0 \\ 19 \\ 19 \\ 0 \\ 0 \end{pmatrix}$$

Αν εφαρμόσουμε τη μέθοδο FME, το σύστημα που προκύπτει είναι το εξής:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ -1 & -4 & 0 & 0 \\ -3 & -2 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ -5 & 0 & 1 & 0 \\ 0 & 20 & 1 & 0 \\ -6 & -4 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -20 & -1 & 0 \\ 5 & 0 & -1 & 0 \\ 6 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -20 & -1 & 3 \\ 10 & 0 & -2 & 1 \\ 0 & 0 & 0 & -1 \\ -10 & 0 & 2 & -1 \\ 0 & 20 & 1 & -3 \end{pmatrix} \begin{pmatrix} j_1^S \\ j_2^S \\ j_1 \\ j_2 \end{pmatrix} \leq \begin{pmatrix} 7 \\ 3 \\ 14 \\ 4 \\ 19 \\ 4 \\ 4 \\ 2 \\ 19 \\ 87 \\ 9 \\ 39 \\ 19 \\ 0 \\ 0 \\ 0 \\ 29 \\ 19 \\ 0 \\ 0 \\ 9 \\ 0 \end{pmatrix}$$

Με βάση τα παραπάνω προκύπτει ο σειριακός μετασχηματισμένος κώδικας που δίνεται στη συνέχεια:

```
FOR (j1^S=3; j1^S ≤7; j1^S++)
   FOR (j2^S=max((-4-j1^S)/4,(-4-3j1^S)/2,-2); j2^S ≤ min((14-j1^S)/4,(19-3j1^S)/2,4); j2^S++)
      FOR (j1=max(-19-20j2^S,5j1^S,6j1^S+4j2^S,0); j1 ≤ min(19+5j1^S,87-20j2^S,9+6j1^S+4j2^S,39); j1++)
         FOR (j2=max(0,-10j1^S+2j1-9,(20j2^S+j1)/3); j2 ≤ min(29,(19+20j2^S+j1)/3,-10j1^S+2j1); j2++)
            A[j1,j2]=A[j1-1,j2-2]+A[j1-3,j2-1];
         ENDFOR
      ENDFOR
   ENDFOR
ENDFOR                                                                              ⊣
```

2.7 Χρονική Δρομολόγηση Φωλιασμένων Βρόχων

Χρονική Δρομολόγηση ενός φωλιασμένου βρόχου είναι η διαδικασία απεικόνισης των επαναλήψεων του βρόχου σε διαφορετικές χρονικές στιγμές. Σκοπός της χρονικής δρομολόγησης είναι η εύρεση του μέγιστου παραλληλισμού, έτσι ώστε να μικρύνει όσο το δυνατόν περισσότερο ο χρόνος εκτέλεσης του βρόχου, χωρίς όμως να αλλάξει η λογική και τα αποτελέσματα της ακολουθιακής ροής εκτέλεσης. Με άλλα λόγια, κάθε χρονική δρομολόγηση πρέπει να είναι έγκυρη, δηλαδή να εντοπίζει για κάθε χρονική στιγμή όλα εκείνα τα στιγμιότυπα-επαναλήψεις του φωλιασμένου βρόχου που μπορούν να εκτελεστούν ταυτόχρονα, διατηρώντας όμως τις σχέσεις εξαρτήσεων που είχε αρχικά ο φωλιασμένος βρόχος. Έτσι, αν δύο επαναλήψεις του φωλιασμένου βρόχου $\vec{j_1}$ και $\vec{j_2}$ έχουν εξάρτηση μεταξύ τους, δηλαδή ισχύει $\vec{j_2} = \vec{j_1} + \vec{d}$, τότε η εξάρτηση αυτή πρέπει να διατηρηθεί μετά τη χρονική δρομολόγηση του βρόχου, δηλαδή το στιγμιότυπο $\vec{j_2}$ πρέπει να τοποθετηθεί χρονικά μετά το στιγμιότυπο $\vec{j_1}$. Ο τυπικός ορισμός της χρονικής δρομολόγησης δίνεται στη συνέχεια:

Ορισμός 2.9 *Χρονική Δρομολόγηση ενός φωλιασμένου βρόχου ονομάζεται η συνάρτηση* s : $J \rightarrow N$ *τέτοια ώστε, για οποιοδήποτε ζεύγος σημείων* $\vec{j_1}$, $\vec{j_2}$, *να ισχύει* $s(\vec{j_2}) > s(\vec{j_1})$, *αν* $\vec{j_2} = \vec{j_1} + \vec{d}$, *όπου* $\vec{d} \in D$.

Η *Γραμμική Χρονική Δρομολόγηση* είναι μια ειδική περίπτωση χρονικής δρομολόγησης, που προκύπτει με την εφαρμογή μιας γραμμικής συνάρτησης στα στιγμιότυπα ενός φωλιασμένου βρόχου. Η γραμμική χρονική δρομολόγηση ορίζεται ως εξής:

Ορισμός 2.10 *Γραμμική Χρονική Δρομολόγηση ενός φωλιασμένου βρόχου ορίζεται ως η χρονική δρομολόγηση* s_Π *τέτοια ώστε, για κάθε* $\vec{j} \in J:\ s_\Pi(\vec{j}) = \lfloor \frac{\Pi \vec{j}^T + c}{disp\Pi} \rfloor$, *όπου* $\Pi \in Z^{1\times n}$ *και* $disp\Pi = min\{\Pi \vec{d_i}^T : \vec{d_i} \in D\}$, $c = t_0 = -min\{\Pi \vec{j}^T : \vec{j1} \in J\}$.

Στον παραπάνω ορισμό το διάνυσμα-γραμμή Π είναι το *γραμμικό διάνυσμα δρομολόγησης* (linear scheduling vector), η σταθερά t_0 λέγεται *σταθερά έναρξης* και η σταθερά $disp\Pi$ *σταθερά μετατόπισης*.

2.8 Μοντέλο Αλγορίθμων

Συνοψίζοντας, θα παρουσιάσουμε το μοντέλο των αλγορίθμων στους οποίους αναφέρεται η παρούσα εργασία. Έτσι, οι αλγόριθμοι που μας αφορούν έχουν τις εξής ιδιότητες:

- Είναι *τέλεια φωλιασμένοι βρόχοι*. Τονίζουμε εδώ, ότι ο μετασχηματισμός υπερκόμβων έχει προταθεί για τέλεια φωλιασμένους βρόχους.
- Οι εξαρτήσεις του φωλιασμένου βρόχου είναι *σταθερές* (constant) και *ομοιόμορφες* (uniform). Ο περιορισμός αυτός τίθεται για να είναι δυνατή η αυτόματη παραγωγή των συνόλων επικοινωνίας (Κεφάλαιο 4).
- Ο αλγόριθμος έχει n γραμμικά ανεξάρτητα διανύσματα εξάρτησης. Πρόκειται δηλαδή για `DOACROSS` βρόχους. Αν τα γραμμικά ανεξάρτητα διανύσματα είναι λιγότερα από n, τότε ο φωλιασμένος βρόχος μπορεί να μετασχηματιστεί έτσι ώστε να περιέχει `DOALL` βρόχους και επομένως μπορούν να εφαρμοστούν άλλες μέθοδοι παραλληλοποίησης για την επίτευξη αποδοτικού παραλληλισμού και δεν είναι απαραίτητη η εφαρμογή του μετασχηματισμού υπερκόμβων.
- Το μέγεθος των διανυσμάτων εξάρτησης είναι μικρότερο από το μέγεθος των υπερκόμβων, επομένως τα διανύσματα εξάρτησης συνδέουν σημεία που βρίσκονται είτε στο εσωτερικό ενός υπερκόμβου ή σε υπερκόμβους που εφάπτονται. Έτσι, θεωρούμε ότι τα

μετασχηματισμένα διανύσματα εξάρτησης $\vec{d^S}$ περιέχουν μόνο μηδενικά και μονάδες. Η παραδοχή αυτή είναι απολύτως λογική, αφού στην πράξη το μέγεθος του υπερκόμβου είναι τάξεις μεγέθους μεγαλύτερο από το μέγεθος των διανυσμάτων εξάρτησης. Στη συντριπτική πλειοψηφία των περιπτώσεων, οι εξαρτήσεις αφορούν γειτονικά σημεία του χώρου επαναλήψεων. Αντίθετα, ένας υπερκόμβος προκειμένου να ενσωματώσει ικανό όγκο υπολογισμού, ώστε να μειωθεί η επιβάρυνση λόγω επικοινωνίας, περιλαμβάνει εκατοντάδες ή και χιλιάδες επαναλήψεις.

- Δεν υπάρχουν αντιεξαρτήσεις και εξαρτήσεις εξόδου. Θεωρούμε δηλαδή ότι κάθε στιγμιότυπο των δεικτών του βρόχου γράφει σε μοναδικό σημείο της μνήμης. Γενικά, οι αντιεξαρτήσεις και οι εξαρτήσεις εξόδου ενός αλγορίθμου μπορούν να απαλειφθούν [CDRV98].

- Οι βρόχοι περιέχουν μία εντολή ανάθεσης σε έναν πίνακα δεδομένων. Η παραδοχή αυτή γίνεται για να απλοποιηθούν οι συμβολισμοί και οι εκφράσεις που χρησιμοποιούνται στην παρούσα εργασία. Όλες οι τεχνικές που προτείνονται μπορούν εύκολα να επεκταθούν σε περισσότερες εντολές ανάθεσης και πίνακες δεδομένων.

Με βάση τα παραπάνω οι αλγόριθμοι που μας απασχολούν θα έχουν τη μορφή:

```
FOR (j1=l1; j1 ≤ u1; j1+=c1)
   FOR (j2=l2; j2 ≤ u2; j2+=c2)
      ...
      FOR (jn=ln; jn ≤ un; jn+=cn)
         A[fw(j⃗)]=F(A[fw(j⃗-d⃗1)],..., A[fw(j⃗-d⃗q)]);
      ENDFOR
      ...
   ENDFOR
ENDFOR
```

Στον παραπάνω κώδικα, A είναι ο πίνακας δεδομένων, $f_w : Z^n \rightarrow Z^k$ είναι η συνάρτηση εγγραφής στον πίνακα A και $\vec{d_1} \ldots \vec{d_q}$ τα διανύσματα εξάρτησης του αλγορίθμου ($q \geq n$). Στη συνέχεια δίνεται ο ορισμός του Χώρου Δεδομένων (Data Space-DS) ενός φωλιασμένου βρόχου σαν αυτόν που περιγράψαμε παραπάνω.

Ορισμός 2.11 *Ο Χώρος Δεδομένων (DS) ενός φωλιασμένου βρόχου ορίζεται ως εξής:*

$$DS = \{\vec{w} \in Z^k \mid \vec{w} = f_w(\vec{j}) \mid \vec{j} \in J\} \tag{2.10}$$

Κώδικες που υπακούουν στο μοντέλο που παρουσιάστηκε, συναντώνται πολύ συχνά κατά την αριθμητική επίλυση μερικών διαφορικών εξισώσεων. Πιο συγκεκριμένα, προβλήματα διάχυσης θερμότητας ή μετάδοσης δυναμικού, προβλήματα ρευστοδυναμικής κ.λ.π. μοντελοποιούνται σαν προβλήματα μερικών διαφορικών εξισώσεων αρχικών οριακών συνθηκών, τα οποία επιλύονται αριθμητικά με τη βοήθεια φωλιασμένων βρόχων που έχουν τα παραπάνω χαρακτηριστικά. Στο Παράρτημα Β παρουσιάζουμε πέντε τέτοιους κώδικες οι οποίοι θα χρησιμοποιηθούν και στα πειράματα του Κεφαλαίου 5.

Κεφάλαιο 3

Παραγωγή Σειριακού Μετασχηματισμένου Κώδικα

Στο κεφάλαιο αυτό θα παρουσιάσουμε μία νέα μέθοδο παραγωγής σειριακού κώδικα για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Ονομάζουμε αυτόν τον κώδικα *σειριακό μετασχηματισμένο κώδικα*. Τονίζουμε το γεγονός ότι ο κώδικας αυτός είναι σειριακός, για να τον ξεχωρίσουμε από τον τελικό παράλληλο κώδικα. Όπως δείξαμε και στο Σχήμα 1.2, η διαδικασία παραγωγής του σειριακού κώδικα προηγείται της παραλληλοποίησης. Εφ' όσον αναφερόμαστε σε μετασχηματισμούς υπερκόμβων βρόχων που θα εκτελεστούν σε παράλληλες αρχιτεκτονικές, ο σειριακός μετασχηματισμένος κώδικας δεν έχει κάποια πρακτική εφαρμογή, αλλά αποτελεί έναν ενδιάμεσο κώδικα από τον οποίο θα προκύψει ο τελικός παράλληλος SPMD κώδικας.

Όπως αναφέρθηκε και στην Ενότητα 2.6.3, ο σειριακός μετασχηματισμένος κώδικας αναδιατάσσει την αρχική σειρά εκτέλεσης του φωλιασμένου βρόχου, έτσι ώστε στο νέο κώδικα να διατρέχονται διαδοχικά οι υπερκόμβοι και για κάθε υπερκόμβο να διασχίζονται τα σημεία στο εσωτερικό του. Έτσι, ένας αρχικός n-διάστατος φωλιασμένος βρόχος μετασχηματίζεται σε ένα $2n$-διάστατο, όπου οι n εξωτερικότεροι βρόχοι διατρέχουν τους υπερκόμβους και οι n εσωτερικότεροι τα σημεία στο εσωτερικό των υπερκόμβων.

Στην Ενότητα 2.6.3 παρουσιάσαμε εν συντομία τη μέθοδο των Ancourt και Irigoin για την

παραγωγή σειριακού μετασχηματισμένου κώδικα, όπως αυτή προτάθηκε στην εργασία τους [AI91]. Η μέθοδος αυτή, όπως και πολλές άλλες που έχουν προταθεί για την αυτόματη παραγωγή παράλληλου κώδικα, στηρίζεται στην κατασκευή συστημάτων ανισοτήτων και στην απαλοιφή τους με τη μέθοδο Fourier-Motzkin Elimination (FME). Στο συγκεκριμένο πρόβλημα τα συστήματα που μας ενδιαφέρουν είναι τα (2.8) και (2.9). Τα συστήματα αυτά είναι ισοδύναμα, έτσι θα χρησιμοποιούμε μόνο το σύστημα (2.8), το οποίο ουσιαστικά ενοποιεί τις ανισότητες που περιγράφουν τον αρχικό χώρο επαναλήψεων (σύστημα (2.2)), με τις ανισότητες που επαληθεύονται από τα σημεία στο εσωτερικό ενός υπερκόμβου (σύστημα (2.7)). Η εφαρμογή της μεθόδου FME στο παραπάνω σύστημα το μετασχηματίζει σε μορφή τέτοια, ώστε οι ανισότητές του να μπορούν να χρησιμοποιηθούν για να καθορίσουν τα όρια των μεταβλητών ελέγχου ενός φωλιασμένου βρόχου. Για να συμβαίνει αυτό, θα πρέπει σε κάθε ανισότητα, να μην εμφανίζονται μεταβλητές που θα χρησιμοποιηθούν ως μεταβλητές ελέγχου σε εσωτερικότερο βρόχο μέσα στο φώλιασμα. Λεπτομερής περιγραφή της μεθόδου FME δίνεται στο Παράρτημα Α.

Το μεγάλο μειονέκτημα της μεθόδου FME είναι ότι είναι εξαιρετικά πολύπλοκη τόσο σε χρόνο όσο και σε χώρο. Συγκεκριμένα, η πολυπλοκότητά της είναι διπλά εκθετική στον αριθμό των ανισοτήτων του αρχικού συστήματος. Γενικά, η ιδέα της απαλοιφής FME μοιάζει με την ιδέα της απαλοιφής Gauss για συστήματα εξισώσεων. Και εδώ διεξάγονται γραμμοπράξεις για να μετασχηματίσουν το σύστημα σε κάτω τριγωνική μορφή. Μία γραμμοπράξη όμως ανάμεσα σε δύο ανισότητες δημιουργεί μία νέα ανισότητα η οποία προστίθεται στις ήδη υπάρχουσες. Έτσι, η μέθοδος "γεννά" διαρκώς νέες ανισότητες, γεγονός που εξηγεί διαισθητικά την ιδιαίτερα μεγάλη πολυπλοκότητά της.

Οι συνέπειες της πολύ υψηλής πολυπλοκότητας της μεθόδου FME είναι πολύ σοβαρές στη μέθοδο που πρότειναν οι Ancourt και Irigoin. Κατ' αρχήν η διαδικασία μεταγλώττισης του αρχικού προγράμματος επιβαρύνεται δραματικά. Έτσι, είναι πολύ πιθανόν ο χρόνος μεταγλώττισης να είναι τόσο μεγάλος (της τάξης των ωρών, ή ακόμα και ημερών), που να καθιστά την εφαρμογή της μεθόδου απαγορευτική. Επιπρόσθετα, είναι επίσης πιθανόν ο χώρος μνήμης που απαιτείται για την αποθήκευση των διαρκώς αυξανόμενων σε μέγεθος συστημάτων ανισοτήτων, να υπερβαίνει την εικονική μνήμη του συστήματος. Πειραματικά παραδείγματα έχουν δείξει πως σε κάποιες περιπτώσεις 1GByte εικονικής μνήμης δεν ήταν αρκετό για να καλύψει τις ανάγκες της μεθόδου σε χώρο μνήμης και έτσι η διαδικασία δεν κατέστη δυνατόν να ολοκληρωθεί.

Ακόμα σημαντικότερο όμως είναι το κόστος που επιφέρει η μέθοδος FME στην αποδοτικότητα του παραγόμενου κώδικα. Με βάση τα όσα αναφέρθηκαν παραπάνω, το τελικό σύστημα

που προκύπτει μετά την εφαρμογή της μεθόδου FME περιλαμβάνει μεγαλύτερο αριθμό ανισοτήτων από το αρχικό. Δεδομένου ότι κάθε ανισότητα αντιστοιχεί σε έναν υπολογισμό που πρέπει να γίνει κατά την αποτίμηση των ορίων του μετασχηματισμένου φωλιασμένου βρόχου, είναι μάλλον εμφανές πως όσο περισσότερες είναι οι επιπλέον ανισότητες που προστίθενται κατά την εφαρμογή της μεθόδου FME, τόσο μεγαλύτερη είναι η επιβάρυνση που υπεισέρχεται στον παραγόμενο κώδικα. Όπως θα δείξουμε και στο παράδειγμα της επόμενης ενότητας, η επιβάρυνση αυτή μπορεί να είναι πολύ σημαντική και να αποτρέπει την αποδοτική εκτέλεση του μετασχηματισμένου φωλιασμένου βρόχου.

Η μέθοδος που προτείνεται στο κεφάλαιο αυτό, λαμβάνει υπ' όψιν της τα παραπάνω προβλήματα, προσπαθεί να μειώσει το χρόνο μεταγλώττισης και να γεννήσει όσο το δυνατόν αποδοτικότερο κώδικα. Επειδή η εφαρμογή της μεθόδου FME δεν μπορεί να αποφευχθεί, η προτεινόμενη μέθοδος στοχεύει στη μείωση του μεγέθους του αρχικού συστήματος ανισοτήτων, μειώνοντας έτσι τα συνολικά βήματα που απαιτούνται για να ολοκληρωθεί η απαλοιφή. Επιπρόσθετα, για τη διάσχιση των εσωτερικών σημείων των υπερκόμβων, εφαρμόζεται ένας μη-ορθομοναδιαίος μετασχηματισμός που μετασχηματίζει τους μη-ορθογώνιους υπερκόμβους σε ορθογώνιους. Με αυτό τον τρόπο μειώνονται οι εκφράσεις που απαιτούνται για την αποτίμηση των ορίων των μεταβλητών που διατρέχουν τα εσωτερικά σημεία των υπερκόμβων και επομένως αυξάνεται η αποδοτικότητα του παραγόμενου κώδικα.

Η οργάνωση του κεφαλαίου αυτού έχει ως εξής: Στην επόμενη ενότητα θα παρουσιάσουμε ένα εισαγωγικό παράδειγμα, που δείχνει τα προβλήματα που δημιουργούνται στη μέθοδο των Ancourt και Irigoin. Στις Ενότητες 3.2 και 3.3 παρουσιάζεται η προτεινόμενη μέθοδος. Και εδώ, όπως και στην εργασία των Ancourt και Irigoin, χωρίζουμε το πρόβλημα της παραγωγής του σειριακού μετασχηματισμένου κώδικα στα υποπροβλήματα της διάσχισης των υπερκόμβων (Ενότητα 3.2) και της διάσχισης των σημείων στο εσωτερικό των υπερκόμβων (Ενότητα 3.3). Τέλος, στην Ενότητα 3.4 γίνεται μια θεωρητική σύγκριση της προτεινόμενης μεθόδου με αυτή των Ancourt και Irigoin.

3.1 Ένα Εισαγωγικό Παράδειγμα

Ας επανέλθουμε στο φωλιασμένο βρόχο του Παραδείγματος 2.7. Για να μεγαλώσουμε το μέγεθος του προβλήματος, θα προεκτείνουμε τα όρια του αρχικού χώρου, επομένως αναφερόμαστε στον παρακάτω αρχικό φωλιασμένο βρόχο:

```
FOR (j1=0; j1 ≤4999; j1++)
   FOR (j2=0; j2 ≤4999; j2++)
      A[j1,j2]=A[j1-1,j2-2]+A[j1-3,j2-1];
   ENDFOR
ENDFOR
```

Εφαρμόζοντας τη μέθοδο των Ancourt και Irigoin για τον ίδιο μετασχηματισμό υπερκόμβων όπως και στο Παράδειγμα 2.7, προκύπτει ο παρακάτω σειριακός μετασχηματισμένος κώδικας, ο οποίος ουσιαστικά είναι ο ίδιος με το Παράδειγμα 2.7, διαφέρει όμως μόνο σε μερικές σταθερές:

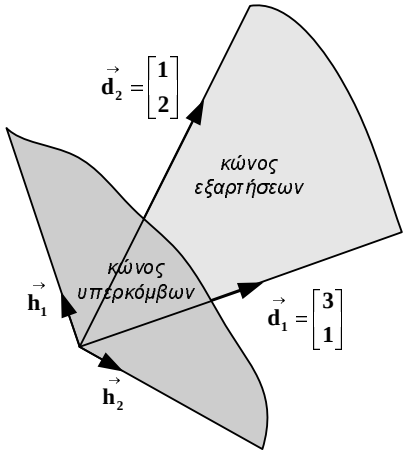
```
FOR (j1^S=500; j1^S ≤999; j1^S++)
   FOR (j2^S=max(⌈(-4-j1^S)/4⌉,⌈(-4-3j1^S)/2⌉,-250); j2^S ≤min(⌊(2499-j1^S)/4⌋,⌊(2499-3j1^S)/2⌋,749); j2^S++)
      FOR (j1=max(-19-20j2^S,5j1^S,6j1^S+4j2^S,0); j1 ≤min(2504+5j1^S,14997-20j2^S,9+6j1^S+4j2^S,4999); j1++)
         FOR (j2=max(0,-10j1^S+2j1-9,⌈(20j2^S+j1)/3⌉); j2 ≤ min(4999,⌊(19+20j2^S+j1)/3⌋,-10j1^S+2j1); j2++)
            A[j1,j2]=A[j1-1,j2-2]+A[j1-3,j2-1];
         ENDFOR
      ENDFOR
   ENDFOR
ENDFOR
```

Εφαρμόζουμε επίσης έναν ορθογώνιο μετασχηματισμό υπερκόμβων στον αρχικό χώρο, ο οποίος περιγράφεται από τον πίνακα $P = \begin{bmatrix} 8 & 0 \\ 0 & 5 \end{bmatrix}$. Το μέγεθος του υπερκόμβου είναι ίδιο με το Παράδειγμα 2.7 ($detP = 40$). Ο σειριακός μετασχηματισμένος κώδικας σ' αυτήν την περίπτωση θα είναι:

```
FOR (j1^S=0; j1^S ≤4999; j1^S+=8)
   FOR (j2^S=0; j2^S ≤4999; j2^S+=5)
      FOR (j1=j1^S; j1 ≤ j1^S+7; j1++)
         FOR (j2=j2^S; j2 ≤ j2^S+4; j2++)
            A[j1,j2]=A[j1-1,j2-2]+A[j1-3,j2-1];
         ENDFOR
      ENDFOR
   ENDFOR
ENDFOR
```

Θεωρητικά, σύμφωνα με τα συμπεράσματα των εργασιών [HS02] και [HCF03], αναμένουμε ο μη-ορθογώνιος μετασχηματισμός υπερκόμβων που περιγράφεται από τον πίνακα $P = \begin{bmatrix} 6 & 4 \\ 2 & 8 \end{bmatrix}$ να είναι καλύτερος από τον τετραγωνικό με $P = \begin{bmatrix} 8 & 0 \\ 0 & 5 \end{bmatrix}$ ως προς τον τελικό, παράλληλο χρό-

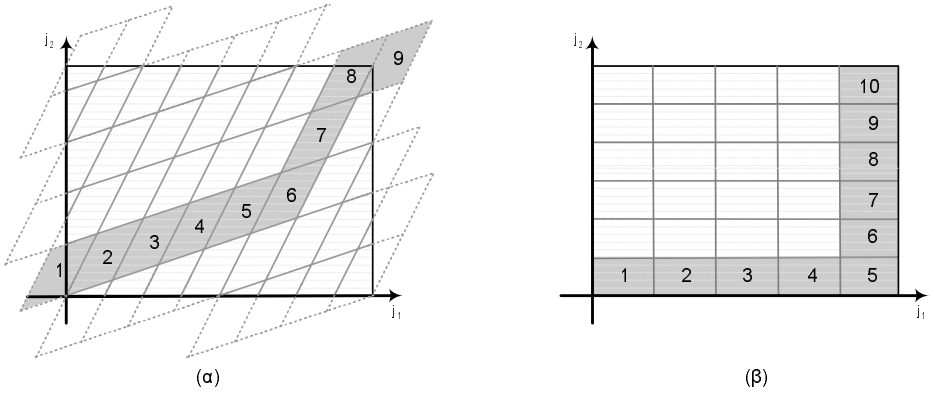
νο εκτέλεσης. Στην πρώτη περίπτωση, τα διανύσματα του P είναι παράλληλα στον κώνο των εξαρτήσεων και ισοδύναμα τα διανύσματα του πίνακα H είναι παράλληλα στον κώνο υπερκόμβων όπως φαίνεται και στο Σχήμα 3.1. Αυτό ακριβώς το γεγονός, σύμφωνα τις εργασίες [HS02] και [HCF03], θα οδηγήσει σε μικρότερο συνολικό αριθμό βημάτων για την ολοκλήρωση του αλγορίθμου. Η γενική ιδέα απεικονίζεται στο Σχήμα 3.2, όπου για οικονομία χώρου δείχνουμε τον αριθμό των βημάτων που απαιτούνται για την ολοκλήρωση του αλγορίθμου σε μικρότερο χώρο επαναλήψεων, αυτόν του Παραδείγματος 2.7. Επιβεβαιώνεται σχηματικά, ότι με το μη-ορθογώνιο μετασχηματισμό το πρόβλημα χρειάζεται εννέα βήματα για να ολοκληρωθεί, ενώ με τον ορθογώνιο χρειάζεται δέκα. Σημειώνεται, ότι μία μέθοδος για τον υπολογισμό του κώνου υπερκόμβων από τα διανύσματα ενός αλγορίθμου δίνεται στην εργασία [Xue97a].



Σχήμα 3.1: Επιλογή του μετασχηματισμού υπερκόμβων από τον κώνο των εξαρτήσεων

Προκειμένου να μελετήσουμε την αποδοτικότητα του σειριακού μετασχηματισμένου κώδικα σε κάθε περίπτωση, εκτελέσαμε τους παραπάνω κώδικες σε ένα υπολογιστή με επεξεργαστή PIII και συχνότητα 800MHz. Για να αποφύγουμε την επίδραση της λανθάνουσας μνήμης στο χρόνο εκτέλεσης, αντικαταστήσαμε την πράξη του πίνακα A με μία βαθμωτή πράξη κινητής υποδιαστολής διπλής ακρίβειας, τα δεδομένα της οποίας διατηρούνται στους καταχωρητές του επεξεργαστή. Με αυτό τον τρόπο αποκλείουμε βελτιώσεις στο χρόνο εκτέλεσης λόγω καλύτερης χρήσης της λανθάνουσας μνήμης στους σειριακούς μετασχηματισμένους κώδικες. Ο χρόνος εκτέλεσης του αρχικού προγράμματος ήταν $2.12sec$, από τα οποία μόνο τα $0.063msec$ καταναλώθηκαν στην αποτίμηση των ορίων του φωλιασμένου βρόχου. Ο χρόνος εκτέλεσης του δεύτερου κώδικα ήταν $9.18sec$ εκ των οποίων τα $7.27sec$ καταναλώθηκαν στην αποτίμηση των

νέων ορίων του βρόχου. Παρατηρούμε ότι η καθυστέρηση που επιβάλει ο μετασχηματισμός υπερκόμβων είναι πάρα πολύ μεγάλη και οφείλεται προφανώς στις ιδιαίτερα βαρείς εκφράσεις αποτίμησης των ορίων του μετασχηματισμένου βρόχου. Ακόμα και στην περίπτωση κατά την οποία βελτιστοποιήσαμε τον παραπάνω κώδικα αφαιρώντας περιττές ανισότητες με τις μεθόδους ad-hoc και exact (βλ. Παράρτημα Α) και παράγοντας διαφορετικό κώδικα για τους εσωτερικούς και τους εξωτερικούς υπερκόμβους (ώστε οι εσωτερικοί υπερκόμβοι να μην επιβαρύνονται με επιπλέον εκφράσεις που απαιτούνται για να μην ξεπεράσουμε τα όρια του αρχικού χώρου), η κατάσταση δεν βελτιώθηκε εντυπωσιακά. Ο συνολικός χρόνος εκτέλεσης σ' αυτήν την περίπτωση ήταν $7.87sec$ με τα $6.17sec$ να καταναλώνονται στην αποτίμηση των ορίων. Από την άλλη, όπως αναμενόταν, ο τρίτος κώδικας που περιγράφει ορθογώνιο μετασχηματισμό είναι πολύ πιο αποδοτικός, καθώς εκτελέστηκε σε $2.2sec$, εκ των οποίων μόνο τα $125msec$ ήταν για την αποτίμηση των ορίων του βρόχου. Παρατηρούμε ότι αυτός ο χρόνος είναι διπλάσιος του χρόνου που απαιτήθηκε κατά την αποτίμηση των ορίων του αρχικού χώρου, πράγμα αναμενόμενο καθώς στο μετασχηματισμένο κώδικα έχουμε τέσσερις αντί για δύο βρόχους, τα όρια των οποίων παραμένουν σταθερά. Τα πειραματικά αποτελέσματα συνοψίζονται στον Πίνακα 3.1.



Σχήμα 3.2: Συνολικά βήματα εκτέλεσης για τους μετασχηματισμούς υπερκόμβων του εισαγωγικού παραδείγματος: (α) Μη-Ορθογώνιος μετασχηματισμός (β) Ορθογώνιος μετασχηματισμός

Παρατηρούμε δηλαδή ότι παρ' όλο που θεωρητικά αναμένουμε ο μη-ορθογώνιος μετασχηματισμός να οδηγήσει σε μικρότερο παράλληλο χρόνο εκτέλεσης, ο σειριακός μετασχηματισμένος κώδικας είναι ιδιαίτερα αργός. Έτσι, είναι εξαιρετικά αμφίβολο αν το θεωρητικό κέρδος του μη-ορθογώνιου μετασχηματισμού θα μπορέσει να αναδειχθεί και στην πράξη. Είναι χαρακτηριστικό το γεγονός, ότι στο συγκεκριμένο παράδειγμα απαιτούνται τέσσερις επεξεργαστές απλά

	Συνολικός Χρόνος Εκτέλεσης (sec)	Χρόνος Αποτίμησης Ορίων (sec)	Χρόνος Υπολογισμών (sec)
Αρχικός Χώρος	2.12	0.063	2.06
Μη-Ορθογώνιος Μετασχηματισμός	9.18	7.27	1.7
Μη-Ορθογώνιος Μετασχηματισμός (βελτ.)	7.87	6.17	1.7
Ορθογώνιος Μετασχηματισμός	2.2	0.13	2.07

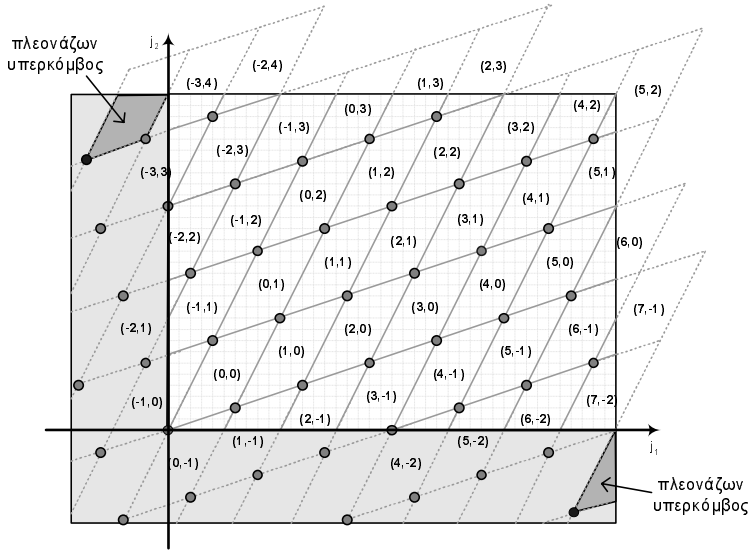
Πίνακας 3.1: Χρόνοι εκτέλεσης για τους σειριακούς μετασχηματισμένους κώδικες του εισαγωγικού παραδείγματος

για να επιτύχουμε παράλληλο χρόνο εκτέλεσης στο μη-ορθογώνια μετασχηματισμένο χώρο, που να είναι ίσος με το χρόνο εκτέλεσης του αρχικού φωλιασμένου βρόχου. Αυτή η επιβάρυνση στον σειριακό μετασχηματισμένο κώδικα είναι ο λόγος που μη-ορθογώνιοι μετασχηματισμοί υπερκόμβων δεν έχουν εφαρμοστεί στην πράξη από ερευνητικούς ή εμπορικούς μεταγλωττιστές. Επομένως, προκειμένου να εκμεταλλευτούμε τη θεωρητική υπεροχή των μη-ορθογώνιων μετασχηματισμών, είναι αναγκαίο να διατηρήσουμε την επιβάρυνση που υπεισέρχεται στο σειριακό μετασχηματισμένο κώδικα σε χαμηλά επίπεδα. Η μέθοδος που προτείνεται στη συνέχεια, επιτυγχάνει την παραγωγή αποδοτικού σειριακού μετασχηματισμένου κώδικα με ταυτόχρονη δραστική μείωση του χρόνου μεταγλώττισης.

3.2 Διάσχιση Υπερκόμβων

Στην ενότητα αυτή θα παρουσιάσουμε μία νέα μέθοδο για τη διάσχιση του Χώρου Υπερκόμβων J^S, ή, διατυπώνοντάς το διαφορετικά, θα παρουσιάσουμε μία νέα μέθοδο υπολογισμού των ορίων των n εξωτερικότερων βρόχων του $2n$-διάστατου σειριακού μετασχηματισμένου κώδικα. Όπως ειπώθηκε και στην Ενότητα 2.6.3, ο αρχικός n-διάστατος φωλιασμένος βρόχος που διατρέχεται από τις μεταβλητές ελέγχου $j_1, j_2, \ldots, j_n$, μετασχηματίζεται με βάση το μετασχηματισμό υπερκόμβων σε ένα $2n$-διάστατο βρόχο, που διατρέχεται από τις μεταβλητές ελέγχου $t_1^S, t_2^S, \ldots, t_n^S, j'_1, j'_2, \ldots, j'_n$. Σκοπός της μεθόδου για τη διάσχιση του Χώρου Υπερκόμβων J^S, είναι η εύρεση των κάτω και άνω ορίων l_k^S και u_k^S κάθε μεταβλητής ελέγχου t_k^S, $k = 1, \ldots, n$. Και εδώ, όπως και στη μέθοδο των Ancourt και Irigoin, θα κατασκευάσουμε ένα κατάλληλο

σύστημα ανισοτήτων, το οποίο και θα μετασχηματίσουμε με τη μέθοδο FME. Για να μειώσουμε όμως το χρόνο μεταγλώττισης, θα φροντίσουμε το σύστημα αυτό να είναι όσο το δυνατόν πιο μικρό, όπως θα φανεί και στη συνέχεια.



Σχήμα 3.3: Διεύρυνση του χώρου για να συμπεριληφθούν όλες οι αρχές των υπερκόμβων

Ξεκινώντας από το σύστημα (2.2) που περιγράφει τον αρχικό χώρο, έχουμε τα εξής: $B\vec{j} \leq \vec{b} \Rightarrow BH^{-1}H\vec{j} \leq \vec{b} \Rightarrow BPH\vec{j} \leq \vec{b}$. Αν συμβολίσουμε $H\vec{j} = \vec{x^S}$, τότε θα έχουμε:

$$BP\vec{x^S} \leq \vec{b} \tag{3.1}$$

Το παραπάνω σύστημα είναι πολύ μικρότερο από το (2.8), που χρησιμοποιείται από τη μέθοδο των Ancourt και Irigoin για τη διάσχιση των υπερκόμβων. Ακριβέστερα, αποτελείται από τις μισές ανισότητες και τις μισές μεταβλητές και επομένως αναμένουμε ότι η απαλοιφή του συστήματος αυτού με τη μέθοδο FME θα είναι πολύ πιο σύντομη. Δυστυχώς όμως, το σύστημα (3.1) δεν περιγράφει ακριβώς το Χώρο Υπερκόμβων J^S, πράγμα αναμενόμενο αφού $\vec{x^S} = Hj \neq \lfloor Hj \rfloor = \vec{j^S} \in J^S$. Συγκεκριμένα, το Σύστημα (3.1) επαληθεύεται μόνο από τους υπερκόμβους, των οποίων οι αρχές βρίσκονται μέσα στον αρχικό χώρο J. Όμως, όπως αναφέρθηκε και στην Ενότητα 2.5, υπάρχουν σημεία στο Χώρο Αρχών Υπερκόμβων TOS που δεν ανήκουν στον αρχικό χώρο J. Τα σημεία αυτά δεν επαληθεύουν τον σύστημα (3.1), πρέπει

όμως παρ' όλα αυτά να διασχιστούν. Στο Σχήμα 2.10, οι υπερκόμβοι (-3,3), (-2,1), (4,-2) και άλλοι στα κάτω όρια του χώρου δεν διατρέχονται από αυτό το σύστημα, επειδή οι αρχές τους βρίσκονται εκτός των ορίων του αρχικού χώρου. Παρ' όλα αυτά όμως, είναι δυνατόν να εκμεταλλευτούμε το μικρό μέγεθος του Συστήματος (3.1), τροποποιώντας το κατάλληλα, ώστε να μπορεί να περιγράψει σωστά τις αρχές όλων των υπερκόμβων. Όπως φαίνεται και στο Σχήμα 3.3, αυτό που χρειάζεται, είναι μία κατάλληλη μείωση των κάτω ορίων και/ή μία κατάλληλη αύξηση των άνω ορίων προκειμένου να συμπεριλάβουμε όλα τα σημεία του Χώρου Αρχών Υπερκόμβων TOS. Το λήμμα που ακολουθεί καθορίζει πόσο πρέπει να διευρύνουμε τον αρχικό χώρο για να συμπεριλάβουμε όλες τις αρχές των υπερκόμβων.

Λήμμα 3.1 *Αν εφαρμόσουμε το μετασχηματισμό υπερκόμβων P σε ένα Χώρο Επαναλήψεων J, του οποίου τα όρια περιγράφονται από το σύστημα ανισοτήτων $B\vec{j} \leq \vec{b}$, τότε για όλες τις αρχές υπερκόμβων $\vec{j_0} \in TOS$, θα ισχύει:*

$$B\vec{j_0} \leq \vec{b'} \tag{3.2}$$

όπου το διάνυσμα $\vec{b'}$ υπολογίζεται από την έκφραση:

$$b'_i = b_i + \frac{g-1}{g} \sum_{r=1}^{n} (\vec{\beta_i} \vec{p_r})^-, i = 1, \ldots, n \tag{3.3}$$

και $\vec{\beta_i}$ είναι η i-οστή γραμμή του πίνακα B, $\vec{p_r}$ είναι r-οστή στήλη του πίνακα P και $(\vec{\beta_i} \vec{p_r})^- = max(-\vec{\beta_i} \vec{p_r}, 0)$.

Απόδειξη 3.1 *Έστω ότι το σημείο $\vec{j} \in J$ του αρχικού χώρου επαναλήψεων ανήκει στον υπερκόμβο με αρχή το σημείο $\vec{j_0}$. Τότε το $\vec{j}$ μπορεί να γραφεί ως το άθροισμα του $\vec{j_0}$ και*

ενός γραμμικού συνδυασμού των διανυσμάτων που είναι στήλες του πίνακα υπερκόμβων P, δηλαδή θα ισχύει:

$$\vec{j} = \vec{j_0} + \sum_{l=1}^{n} \lambda_l \vec{p_l} \tag{3.4}$$

Επιπρόσθετα, όπως αναφέρθηκε και στην Ενότητα 2.6.3, ισχύει το εξής σύστημα ανισοτήτων: $\vec{0} \leq gH(\vec{j} - \vec{j_0}) \leq \vec{g} - \vec{1}$. *Η i-οστή γραμμή αυτού του συστήματος μπορεί να γραφεί και ως:* $0 \leq \vec{h_i}(\vec{j} - \vec{j_0}) \leq \frac{g-1}{g}$, *όπου* $\vec{h_i}$ *είναι το i-οστό διάνυσμα-γραμμή του πίνακα* $H = P^{-1}$. *Επομένως,* $0 \leq \vec{h_i} \sum_{l=1}^{n} \lambda_l \vec{p_l} \leq \frac{g-1}{g}$. *Αφού* $PH = I$ *θα ισχύει* $\vec{h_i}\vec{p_i} = 1$ *και* $\vec{h_i}\vec{p_l} = 0$ *αν* $i \neq l$. *Επομένως, η τελευταία έκφραση μπορεί να γραφεί ως εξής:*

$$0 \leq \lambda_i \leq \frac{g-1}{g},\ i = 1, \ldots, n \tag{3.5}$$

Για όλα τα $\vec{j} \in J$ *ισχύει το σύστημα ανισοτήτων* $B\vec{j} = \vec{b}$. *Η k-στή γραμμή του συστήματος αυτού γράφεται ως εξής:* $\vec{\beta_k}\vec{j} \leq b_k$. *Αν λάβουμε υπ' όψιν μας την αντίστοιχη αρχή του υπερκόμβου, η προηγούμενη ανισότητα γράφεται:* $\vec{\beta_k}(\vec{j_0} + \sum_{i=1}^{n} \lambda_i \vec{p_i}) \leq b_k \Rightarrow \vec{\beta_k}\vec{j_0} \leq b_k - \vec{\beta_k}(\sum_{i=1}^{n} \lambda_i \vec{p_i})$

$$\Rightarrow \vec{\beta_k}\vec{j_0} \leq b_k - \sum_{i=1}^{n} \lambda_i (\vec{\beta_k}\vec{p_i}) \tag{3.6}$$

Επιπρόσθετα, σύμφωνα με τη σχέση (3.5), θα ισχύει $0 \leq \lambda_i \leq \frac{g-1}{g}$ *για όλα τα* $i = 1, \ldots, n$. *Πολλαπλασιάζοντας αυτή την έκφραση με* $\vec{\beta_k}\vec{p_i}$ *έχουμε τα εξής:*

- *Αν* $\vec{\beta_k}\vec{p_i} \geq 0 \Rightarrow \lambda_i\vec{\beta_k}\vec{p_i} \geq 0$

- *Αν* $\vec{\beta_k}\vec{p_i} < 0 \Rightarrow \lambda_i\vec{\beta_k}\vec{p_i} \geq \frac{g-1}{g}\vec{\beta_k}\vec{p_i}$

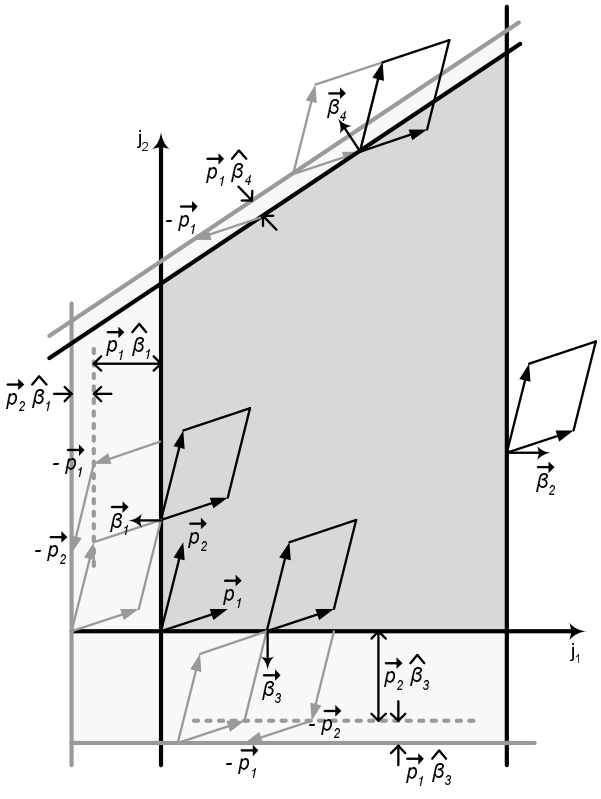
Σύμφωνα με τον ορισμό του συμβόλου $(\vec{\beta_k}\vec{p_i})^- = max(-\vec{\beta_k}\vec{p_i}, 0)$*, οι προηγούμενες ανισότητες και στις δύο περιπτώσεις μπορούν αν γραφούν:* $\lambda_i\vec{\beta_k}\vec{p_i} \geq -\frac{g-1}{g}(\vec{\beta_k}\vec{p_i})^- \Rightarrow -\lambda_i\vec{\beta_k}\vec{p_i} \leq \frac{g-1}{g}(\vec{\beta_k}\vec{p_i})^-$*. Αν αθροίσουμε για* $i = 1, \ldots, n$*, η ανισότητα αυτή δίνει:*

$$-\sum_{i=1}^{n}\lambda_i\vec{\beta_k}\vec{p_i} \leq \frac{g-1}{g}\sum_{i=1}^{n}(\vec{\beta_k}\vec{p_i})^- \tag{3.7}$$

Έτσι, από τις ανισότητες (3.6) και (3.7), συμπεραίνουμε ότι $\vec{\beta_k}\vec{j_0} \leq b_k + \frac{g-1}{g}\sum_{i=1}^{n}(\vec{\beta_k}\vec{p_i})^-$*. Συνεπώς, για κάθε υπερκόμβο με αρχή* $\vec{j_0}$*, ο οποίος έχει έστω και ένα σημείο στον αρχικό χώρο επαναλήψεων, θα ισχύει* $Bj_0 \leq \vec{b'}$*, όπου το διάνυσμα* $\vec{b'}$ *κατασκευάζεται έτσι ώστε το* k*-στό στοιχείο του να δίνεται από την έκφραση:* $b'_k = b_k + \frac{g-1}{g}\sum_{i=1}^{n}(\vec{\beta_k}\vec{p_i})^-$*.* □

Γεωμετρικά, ο παράγοντας που προστίθεται σε κάθε στοιχείο του $\vec{b}$, εκφράζει μία παράλληλη μετατόπιση του αντίστοιχου ορίου του αρχικού χώρου επαναλήψεων. Στο Σχήμα 3.4 παρουσιάζουμε τη γενική ιδέα της ολίσθησης των ορίων του αρχικού χώρου. Κάθε γραμμή $\vec{\beta_i}$ του πίνακα B είναι ένα διάνυσμα κάθετο στην αντίστοιχη οριακή επιφάνεια του χώρου, με κατεύθυνση προς το εξωτερικό του χώρου. Η εξίσωση αυτής της επιφάνειας είναι $\vec{\beta_i}\vec{x} = b_i$, επομένως μία παράλληλη μετατόπιση της επιφάνειας αυτής κατά x_0, εκφράζεται από την εξίσωση $\vec{\beta_i}(\vec{x} - \vec{x_0}) = b_i \Leftrightarrow \vec{\beta_i}\vec{x} = b_i + \vec{\beta_i}\vec{x_0}$. Όπως φαίνεται και από το Σχήμα 3.4, η επιθυμητή διεύρυνση των ορίων επιτυγχάνεται με μετατόπιση κάθε οριακής επιφάνειας κατά το διάνυσμα $-\vec{p_r}$, αν και μόνο αν το διάνυσμα $\vec{p_r}$ σχηματίζει γωνία μεγαλύτερη των 90^o με το διάνυσμα $\vec{\beta_i}$. Αυτό συμβαίνει, για παράδειγμα, με τις γωνίες που σχηματίζουν τα διανύσματα $\vec{\beta_1}$ και $\vec{p_2}$, $\vec{\beta_3}$ και $\vec{p_1}$, $\vec{\beta_3}$ και $\vec{p_2}$, $\vec{\beta_4}$ και $\vec{p_1}$ στο Σχήμα 3.4. Ισοδύναμα, η ολίσθηση πραγματοποιείται αν και μόνο αν ισχύει $\vec{p_r}\vec{\beta_i} < 0$. Επομένως, αν το εσωτερικό γινόμενο μίας στήλης του πίνακα P, έστω $\vec{p_r}$, με μία γραμμή του πίνακα B, έστω $\vec{\beta_i}$, είναι αρνητικό, τότε αφαιρούμε αυτό το εσωτερικό γινόμενο από τη σταθερά b_i για να σχηματίσουμε την κατάλληλη μετατόπιση.

Οι υπολογισμοί για τις ολισθήσεις του Σχήματος 3.4 φαίνονται στον Πίνακα 3.2. Στη Σχέση (3.1), τα παραπάνω εκφράζονται με την πρόσθεση του παράγοντα $(\vec{\beta}_i \vec{p_r})^-$ στη σταθερά b_i για όλα τα διανύσματα $\vec{p_r}$. Με τον πολλαπλασιαστικό παράγοντα $\frac{g-1}{g}$ δεν συμπεριλαμβάνουμε στη διεύρυνση τους υπερκόμβους, τα όρια των οποίων απλά "αγγίζουν" τον αρχικό χώρο, δεδομένου ότι κάθε υπερκόμβος είναι ένας ημιανοικτός χώρος.



Σχήμα 3.4: Γεωμετρική ερμηνεία του Λήμματος 3.4

Αυτή η διεύρυνση των ορίων προκειμένου να συμπεριληφθούν όλες οι αρχές των υπερκόμβων, μπορεί να οδηγήσει στη διάσχιση μερικών υπερκόμβων που οι αρχές τους ανήκουν στο διευρυμένο χώρο, αλλά κανένα από τα εσωτερικά τους σημεία δεν ανήκει στον αρχικό χώρο. Προφανώς το γεγονός αυτό είναι ανεπιθύμητο καθώς προσθέτει επιπλέον διασχίσεις υπερκόμβων κατά την εκτέλεση του τελικού κώδικα. Επειδή όμως οι πλεονάζοντες υπερκόμβοι είναι πολύ λίγοι στον αριθμό και σε συνδυασμό με το γεγονός ότι δεν θα διασχιστούν τα εσωτερικά τους σημεία (όπως θα φανεί στην επόμενη ενότητα), το κόστος της διάσχισης των επιπλέον αυτών υπερκόμβων είναι μηδαμινό. Αυτό που επιτυγχάνουμε τελικά, είναι να δημιουργήσου-

	Εσωτερικό γινόμενο με $\vec{p_1}$	Εσωτερικό γινόμενο με $\vec{p_2}$	Ολίσθηση
$\vec{\beta_1}$	< 0	< 0	$-\vec{p_1}\hat{\beta_1} - \vec{p_2}\hat{\beta_1}$
$\vec{\beta_2}$	> 0	> 0	$-$
$\vec{\beta_3}$	< 0	< 0	$-\vec{p_1}\hat{\beta_3} - \vec{p_2}\hat{\beta_3}$
$\vec{\beta_4}$	< 0	> 0	$-\vec{p_1}\hat{\beta_4}$

Πίνακας 3.2: Ολισθήσεις του αρχικού χώρου για το Σχήμα 3.4

με ένα πολύ μικρότερο σύστημα ανισοτήτων σε σχέση με το (2.8), που περιλαμβάνει όλους τους προς διάσχιση υπερκόμβους, με μοναδικό μειονέκτημα τη διάσχιση πολύ μικρού αριθμού επιπλέον υπερκόμβων.

Παράδειγμα 3.1 Θα διασχίσουμε το Χώρο Υπερκόμβων J^S που δημιουργείται από το μετασχηματισμό υπερκόμβων που περιγράφηκε στα Παραδείγματα 2.3 και 2.4. Με βάση την προσέγγιση που περιγράψαμε, πρέπει να κατασκευάσουμε το σύστημα ανισοτήτων (3.3) χρησιμοποιώντας την έκφραση (3.1). Η έκφραση (3.1) στην περίπτωσή μας δίνει $\vec{b'} = \begin{bmatrix} 39 & 29 & 9.5 & 9.5 \end{bmatrix}^T$ και έτσι το σύστημα (3.2) γίνεται:

$$\begin{pmatrix} 6 & 4 \\ 2 & 8 \\ -6 & -4 \\ -2 & -8 \end{pmatrix} \begin{pmatrix} j_1^S \\ j_2^S \end{pmatrix} \leq \begin{pmatrix} 39 \\ 29 \\ 9.5 \\ 9.5 \end{pmatrix}$$

Η διεύρυνση των ορίων του παραδείγματος αυτού φαίνεται στο Σχήμα 3.3. Μία πρόχειρη εφαρμογή της μεθόδου FME πολλαπλασιάζει τη γραμμή 1 με τη 2 και τη προσθέτει στη γραμμή 4. Έτσι, παίρνουμε $10j_1^S \leq 87.5 \Rightarrow j_1^S \leq 8$. Όμοια παίρνουμε $j_1^S \geq -4$. Επομένως ο βρόχος που θα διασχίζει τις αρχές των υπερκόμβων για το παράδειγμά μας θα έχει τη μορφή:

```
FOR (j1S=-4; j1S ≤8; j1S++)
   FOR (j2S=max(⌈(−9.5−6j1S)/4⌉, ⌈(−9.5−2j1S)/8⌉); j2S ≤min(⌊(39−6j1S)/4⌋, ⌊(29−2j1S)/8⌋); j2S++)
      ...
```

```
    ENDFOR
  ENDFOR
```

Οι υπερκόμβοι $(8,-3)$ και $(-4,4)$ είναι πλεονάζοντες (Σχήμα 3.3). ⊣

3.3 Διάσχιση Σημείων στο Εσωτερικό των Υπερκόμβων

Στην ενότητα αυτή θα παρουσιάσουμε μία νέα μέθοδο για την παραγωγή κώδικα που διασχίζει τα εσωτερικά σημεία των υπερκόμβων. Έτσι, με βάση τα όσα έχουν ειπωθεί ως τώρα, σκοπός της μεθόδου είναι ο καθορισμός των ορίων των n εσωτερικότερων βρόχων του σειριακού μετασχηματισμένου κώδικα. Επομένως, για κάθε μεταβλητή ελέγχου j'_k, πρέπει να υπολογίσουμε τα κάτω και άνω όριά της l'_k και u'_k. Οι Ancourt και Irigoin για το συγκεκριμένο πρόβλημα προτείνουν την κατασκευή του συστήματος (2.7), το οποίο επαληθεύεται από τα σημεία στο εσωτερικό των υπερκόμβων, και το συνδυασμό του με το σύστημα (2.2), που περιγράφει τον αρχικό χώρο.

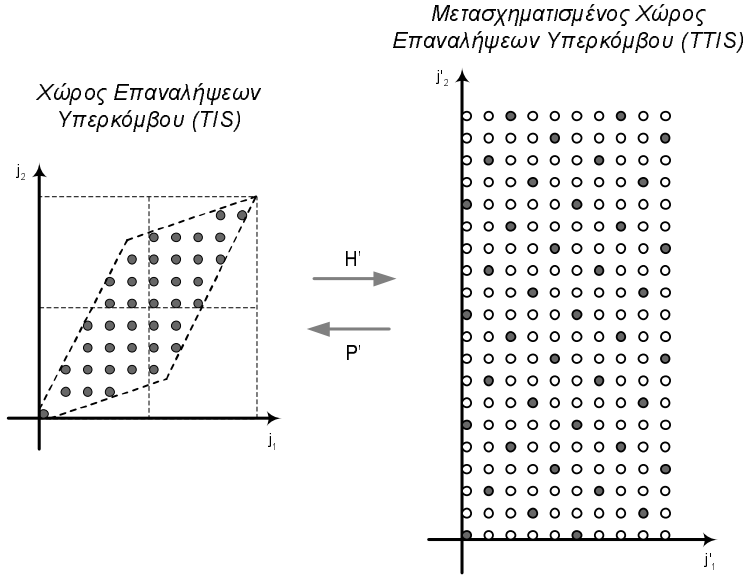
Όπως αναφέρθηκε και προηγουμένως, η παραπάνω διαδικασία δημιουργεί μεγάλο αριθμό πράξεων στην αποτίμηση των ορίων του βρόχου. Η προτεινόμενη μέθοδος απλοποιεί τις εκφράσεις αποτίμησης των ορίων του βρόχου, μετασχηματίζοντας τους μη-ορθογώνιους υπερκόμβους σε ορθογώνιους. Έτσι, για τη διάσχιση ενός μη-ορθογώνιου υπερκόμβου, διασχίζουμε τον αντίστοιχο ορθογώνιο και μετασχηματίζουμε τις μεταβλητές ελέγχου ώστε να προσπελάσουμε το κατάλληλο σημείο. Χρησιμοποιώντας την ορολογία που εισήχθη στην Ενότητα 2.5, ουσιαστικά μετασχηματίζουμε το Χώρο Επαναλήψεων Υπερκόμβου (TIS) σε ένα ορθογώνιο χώρο, το Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου (Transformed Tile Iteration Space-TTIS). Διασχίζουμε τον ορθογώνιο TTIS και μετασχηματίζουμε τις μεταβλητές ελέγχου ώστε να μεταφερθούμε στο κατάλληλο σημείο του TIS. Η παραπάνω διαδικασία τροποποιείται κατάλληλα ώστε να διασχίσουμε τα εσωτερικά σημεία οποιουδήποτε υπερκόμβου και ειδικά αυτών που δεν είναι πλήρεις, επειδή βρίσκονται στα όρια του αρχικού χώρου επαναλήψεων. Η μέθοδος μπορεί να χωριστεί σε τρία βήματα: τον προσδιορισμό του μετασχηματισμού, τη διάσχιση του Χώρου Επαναλήψεων Υπερκόμβου και τη διάσχιση όλων των σημείων του αρχικού χώρου επαναλήψεων. Τα βήματα αυτά θα παρουσιαστούν στις ενότητες που ακολουθούν.

3.3.1 Εύρεση Μετασχηματισμού

Για να μετασχηματίσουμε το Χώρο Επαναλήψεων Υπερκόμβου σε ένα ορθογώνιο χώρο και αντιστρόφως, αναζητούμε ένα ζεύγος μετασχηματισμών (P', H'): $TTIS \xrightarrow{P'} TIS$ και $TIS \xrightarrow{H'} TTIS$ (Σχήμα 3.5). Διαισθητικά, απαιτούμε ο P' να είναι παράλληλος στις πλευρές του υπερκόμβου, δηλαδή τα διανύσματα-στήλες του P' να είναι παράλληλα σε αυτά του P. Ισοδύναμα, με βάση τον ορισμό των πινάκων H και P, απαιτούμε τα διανύσματα-γραμμές του πίνακα H' να είναι παράλληλα στα διανύσματα-γραμμές του H. Επιπρόσθετα, απαιτούμε το πλέγμα του H' να είναι χώρος ακεραίων, ούτως ώστε να μπορεί να διασχιστεί από ένα φωλιασμένο βρόχο. Τυπικά, αναζητούμε ένα n-διάστατο μετασχηματισμό H', ο οποίος θα ικανοποιεί τις εξής δύο απαιτήσεις:

1. $\mathcal{L}(H') \subseteq Z^n$
2. $H' = VH$, όπου V είναι ένας $n \times n$ διαγώνιος πίνακας

Το λήμμα που ακολουθεί αποδεικνύει ότι η πρώτη απαίτηση ισχύει αν και μόνο αν ο H' είναι ακέραιος.



Σχήμα 3.5: Μετασχηματισμός του Χώρου Επαναλήψεων Υπερκόμβου σε ορθογώνιο χώρο

Λήμμα 3.2 *$\vec{j'} = A\vec{j} \in Z^n \ \forall \vec{j} \in Z^n$ αν και μόνο αν ο A είναι πίνακας ακεραίων.*

Απόδειξη 3.2 *Αν ο A είναι πίνακας ακεραίων τότε είναι προφανές ότι $\vec{j'} \in Z^n \; \forall \; \vec{j} \in Z^n$. Για την απόδειξη του αντιστρόφου υποθέτουμε ότι $\vec{j'} \in Z^n \; \forall \; \vec{j} \in Z^n$. Θα αποδείξουμε ότι ο A είναι πίνακας ακεραίων. Χωρίς βλάβη της γενικότητας, επιλέγουμε το $\vec{j}$ να είναι $\vec{j} = \hat{u}_k$, όπου $\hat{u}_k$ είναι το k-στό μοναδιαίο διάνυσμα, $\hat{u}_k = (u_{k1}, \ldots, u_{kn})$, $u_{kk} = 1, u_{kj} = 0, \; j \neq k$. Τότε, σύμφωνα με τα παραπάνω, θα ισχύει $A\hat{u}_k = [\sum_{i=1}^{n} a_{1i}u_{k_i}, \sum_{i=1}^{n} a_{2i}u_{k_i}, \ldots, \; \sum_{i=1}^{n} a_{ni}u_{k_i}]^T = [a_{1k}, a_{2k}, \ldots, a_{nk}]^T \in Z^n$. Επειδή αυτό ισχύει για όλα τα $\hat{u}_k, k = 1 \ldots n$, συμπεραίνουμε πως ο A είναι πίνακας ακεραίων.* □

Μπορούμε να κατασκευάσουμε τον πίνακα V με τον εξής τρόπο: Κάθε διαγώνιο στοιχείο v_{kk} είναι ο μικρότερος θετικός ακέραιος τέτοιος ώστε ο $v_{kk}\vec{h_k}$ να είναι ακέραιος, όπου $\vec{h_k}$ είναι η k-στή γραμμή του πίνακα H. Μ' αυτόν τον τρόπο, και οι δύο απαιτήσεις για τον πίνακα H' θα ικανοποιούνται. Είναι προφανές ότι στη γενική περίπτωση ο H' είναι ένας μη-ορθομοναδιαίος μετασχηματισμός. Αυτό σημαίνει ότι ο Μετασχηματισμένος Χώρος Επαναλήψεων Υπερκόμβου θα περιέχει οπές. Οι οπές στο Σχήμα 3.5 απεικονίζονται με λευκά σημεία.

3.3.2 Διάσχιση του Χώρου Επαναλήψεων Υπερκόμβου (TIS)

Όπως αναφέρθηκε και προηγουμένως, για να διασχίσουμε το Χώρο Επαναλήψεων Υπερκόμβου, θα διασχίσουμε το Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου και θα επιστρέψουμε στα αρχικά σημεία με τον αντίστροφο μετασχηματισμό. Ουσιαστικά, πρέπει να γεννήσουμε ένα n-διάστατο φωλιασμένο βρόχο με μεταβλητές ελέγχου $j'_1, j'_2, \ldots, j'_n$, για να διασχίσουμε το χώρο που προκύπτει από την εφαρμογή του μη-ορθομοναδιαίου μετασχηματισμού H' στο Χώρο Επαναλήψεων Υπερκόμβου. Θα μπορούσε να εφαρμοστεί η μεθοδολογία που παρουσιάστηκε στην Ενότητα 2.6.2, ή κάποια από τις μεθοδολογίες των εργασιών [Ram92], [Ram95], [Xue94], [Li93], [FLV95]. Στην περίπτωσή μας, λόγω της κανονικότητας των υπερκόμβων, τα πράγματα είναι πιο απλά και έτσι η παραγωγή του ζητούμενου κώδικα επιτυγχάνεται ακολουθώντας τα παρακάτω τρία βήματα και μάλιστα χωρίς να είναι αναγκαία η εφαρμογή της μεθόδου FME.

- **Βήμα 1:** *Υπολογισμός των ορίων του Μετασχηματισμένου Χώρου Επαναλήψεων Υπερκόμβου.*

Το λήμμα που ακολουθεί δίνει τα ζητούμενα όρια του Μετασχηματισμένου Χώρου Επαναλήψεων Υπερκόμβου.

Λήμμα 3.3 *Αν* $\vec{j'} \in TTIS$ *τότε θα ισχύει* $0 \leq j'_k \leq v_{kk} - 1, \;\; k = 1, \ldots, n.$

Απόδειξη 3.3 *Από τον ορισμό του Χώρου Επαναλήψεων Υπερκόμβου γνωρίζουμε ότι για τα σημεία* $\vec{j} \in TIS$ *ισχύει* $\lfloor H\vec{j} \rfloor = 0$, *ή ισοδύναμα* $0 \leq \sum_{i=1}^{n} h_{ik} j_k < 1, \;\; k = 1 \ldots, n.$ *Πολλαπλασιάζοντας με* v_{kk} *θα έχουμε* $0 \leq v_{kk} \sum_{i=1}^{n} h_{ik} j_k < v_{kk}$. *Από τον ορισμό του Μετασχηματισμένου Χώρου Επαναλήψεων Υπερκόμβου ισχύει* $j'_k = v_{kk} \sum_{i=1}^{n} h_{ik} j_k$, *άρα* $0 \leq j'_k < v_{kk} \Rightarrow 0 \leq j'_k \leq v_{kk} - 1.$ □

- **Βήμα 2:** *Υπολογισμός βημάτων και αρχικών τιμών για τις μεταβλητές ελέγχου.*
  Τα βήματα και οι αρχικές τιμές των μεταβλητών ελέγχου δεν είναι απαραίτητα μοναδιαία και μηδενικά αντίστοιχα. Αυτό σημαίνει, πως αν μία μεταβλητή ελέγχου αυξηθεί κατά βήμα c_k, όλες οι μεταβλητές $j'_{k+1}, \ldots, j'_n$ θα πρέπει να αρχικοποιηθούν σε κάποιες αρχικές τιμές $a_{(k+1)k}, \ldots, a_{nk}$.

  Υποθέτουμε πως για ένα διάνυσμα επαναλήψεων $\vec{j'}$ στο νέο βρόχο ισχύει $P'\vec{j'} \in Z^n$. Το πρώτο ερώτημα είναι πόσο πρέπει να αυξήσουμε την εσωτερικότερη μεταβλητή ελέγχου j'_n, έτσι ώστε το επόμενο σημείο που θα σαρωθεί να είναι ακέραιο. Τυπικά, αναζητούμε το ελάχιστο $c_n \in Z$, έτσι ώστε $P' \left[\begin{array}{cccc} j'_1 & j'_2 & \ldots & j'_n + c_n \end{array} \right]^T \in Z^n$. Μετά τον καθορισμό του c_n, το επόμενο βήμα είναι να υπολογίσουμε το βήμα της μεταβλητής ελέγχου j'_{n-1} έτσι ώστε το επόμενο σημείο να είναι ακέραιο. Σ' αυτή την περίπτωση, είναι δυνατόν η μεταβλητή ελέγχου j'_n να πρέπει επίσης να τεθεί σε μία αρχική τιμή $a_{n(n-1)} : 0 \leq a_{n(n-1)} < c_n$. Στη γενική περίπτωση του δείκτη j'_k, πρέπει να καθορίσουμε τα $c_k, a_{(k+1)k}, \ldots, a_{nk}$ έτσι ώστε: $P' \left[\begin{array}{cccccc} j'_1 & \ldots & j'_k + c_k & j'_{k+1} + a_{(k+1)k} & \ldots & j'_n + a_{nk} \end{array} \right]^T \in Z^n$. Κάθε μεταβλητή ελέγχου j'_k έχει $k-1$ διαφορετικές αρχικές τιμές a_{ki}, ανάλογα με το ποια από τις $k-1$ εξωτερικότερες μεταβλητές ελέγχου έχει μόλις αυξηθεί. Αυτές οι αρχικές τιμές είναι $a_{k1}, \ldots, a_{k(k-1)}$. Το λήμμα που ακολουθεί αποδεικνύει ότι τα βήματα c_k και οι αρχικές τιμές a_{kl}, $(k = 1 \ldots n \text{ and } l = 1 \ldots k-1)$, μπορούν αν προκύψουν απευθείας από την Ερμητιανή Κανονική Μορφή του πίνακα H', που συμβολίζεται με $\widetilde{H'}$ (Σχήμα 3.6).

Λήμμα 3.4 *Αν $\widetilde{H'}$ είναι η ΕΚΜ του πίνακα H' και $\vec{j'} = (j'_1, j'_2, \ldots, j'_n)$ είναι το διάνυσμα επαναλήψεων που χρησιμοποιείται για να διασχίσει τα ακριβή σημεία του $\mathcal{L}(H')$, τότε το βήμα του δείκτη j'_k θα είναι $c_k = \widetilde{h'}_{kk}$ και οι αρχικές τιμές θα είναι $a_{kl} = \widetilde{h'}_{kl}$, $(k = 1 \ldots n$ και $l = 1 \ldots k-1)$.*

Απόδειξη 3.4 *Από τα Θεωρήματα 2.1 και 2.2 γνωρίζουμε ότι $\mathcal{L}(H') = \mathcal{L}(\widetilde{H'})$. Έτσι, $\vec{0} \in \mathcal{L}(H')$ και οι στήλες του $\widetilde{H'}$ επίσης ανήκουν στο $\mathcal{L}(H')$. Θα δείξουμε πως αν ξεκινήσουμε από ένα σημείο που ανήκει στο $\mathcal{L}(H')$, τότε το σημείο που θα προκύψει προσθέτοντας οποιοδήποτε μη-μηδενικό, λεξικογραφικά μικρότερο διάνυσμα από τα διανύσματα-στήλες του $\widetilde{H'}$ δεν θα ανήκει στο $\mathcal{L}(H')$. Με άλλα λόγια, το βήματα και οι αρχικές τιμές του ζητούμενου βρόχου δεν θα μπορούν να προκύψουν από διανύσματα λεξικογραφικά μικρότερα των στηλών του $\widetilde{H'}$. Χωρίς βλάβη της γενικότητας, θα θεωρήσουμε πως το τρέχον σημείο που ήδη ανήκει στο $\mathcal{L}(H')$ είναι το $\vec{0}$. Έστω το διάνυσμα $\vec{x} \in Z^n/\vec{0}$ με τις ακόλουθες ιδιότητες: $x_i = 0$ για $i < k$ και $0 \leq x_i \leq \widetilde{h'}_{ik}$ για $k \leq i \leq n$. Προφανώς ισχύει $\vec{x} \leq \widetilde{h'}_k$, όπου $\widetilde{h'}_k$ το k-στό διάνυσμα στήλη του $\widetilde{H'}$. Αρκεί να αποδείξουμε πως $\vec{x} \notin \mathcal{L}(H')$. Αν υποθέσουμε ότι $\vec{x} \in \mathcal{L}(H')$, τότε θα ισχύει: $\exists\ \vec{j} \in Z^n : \widetilde{H'}\vec{j} = \vec{x}$. Επειδή ο $\widetilde{H'}$ είναι ένας κάτω-τριγωνικός μη-αρνητικός πίνακας θα ισχύει: $x_1 = \widetilde{h'}_{11} j_1 = 0 \Rightarrow j_1 = 0$. Όμοια, $j_i = 0$ για $i < k$. Έτσι θα ισχύει: $x_k = \widetilde{h'}_{kk} j_k$. Σύμφωνα με τα παραπάνω θα ισχύει επίσης: $0 \leq x_k = \widetilde{h'}_{kk} j_k \leq \widetilde{h'}_{kk} \Rightarrow 0 \leq j_k \leq 1$. Επιπρόσθετα, $0 \leq x_{k+1} = \widetilde{h'}_{(k+1)k} j_k + \widetilde{h'}_{(k+1)(k+1)} j_{k+1} \leq \widetilde{h'}_{(k+1)k}$. Από τον ορισμό της ΕΚΜ του πίνακα γνωρίζουμε πως $\widetilde{h'}_{(k+1)(k+1)} > \widetilde{h'}_{(k+1)k} \Rightarrow j_{k+1} = 0$.*

Όμοια, $j_i = 0$ *για* $i > k+1$. *Επομένως* $\vec{x} = \vec{0}$, *ή* $\vec{x}$ *είναι η* k*-στή στήλη* $\widetilde{H'}$, *δηλαδή δεν μπορεί να υπάρξει μη-μηδενικό λεξικογραφικά μικρότερο διάνυσμα από την* k*-στή στήλη του* $\widetilde{H'}$ *που να ανήκει στο* $\mathcal{L}(H')$. *Προφανώς, δεν μπορεί να υπάρξουν λεξικογραφικά μεγαλύτερα διανύσματα από τις στήλες του* $\widetilde{H'}$ *που να ανήκουν στο* $\mathcal{L}(H')$, *εκτός από γραμμικούς συνδυασμούς των στηλών του* $\widetilde{H'}$. *Αυτό συμβαίνει γιατί κάθε λεξικογραφικά μεγαλύτερο του* $\widetilde{h'}_k$ *διάνυσμα έστω* $\vec{x'}$ *θα μπορεί να γραφεί σαν* $\vec{x'} = \lambda\widetilde{h'}_k + \vec{x}$, *όπου* $\vec{x}$ *λεξικογραφικά μικρότερο του* $\widetilde{h'}_k$. *Προφανώς* $\lambda\widetilde{h'}_k \in \mathcal{L}(H')$, *άρα σύμφωνα με τα παραπάνω* $\vec{x'} = \lambda\widetilde{h'}_k + \vec{x} \notin \mathcal{L}(H')$. □

Με βάση την παραπάνω ανάλυση, το σημείο που θα διασχιστεί από το επόμενο στιγμιότυπο των μεταβλητών ελέγχου, υπολογίζεται με βάση το τρέχον στιγμιότυπο, αφού τα βήματα και οι αρχικές τιμές προστίθενται στις τρέχουσες μεταβλητές. Ειδική μέριμνα πρέπει να ληφθεί, ώστε κάθε φορά που μεταβάλλεται το διάνυσμα $\vec{j'} = (j'_1, \ldots, j'_n)$, το νέο διάνυσμα $\vec{j'}$ να υπολογίζεται σαν άθροισμα του τρέχοντος $\vec{j'}$ και ενός πολλαπλάσιου του ενός διανύσματος-στήλη του $\widetilde{H'}$. Έτσι, εφόσον για το τρέχον στιγμιότυπο ισχύει $\vec{j'} \in \mathcal{L}(H')$, εξασφαλίζουμε πως το επόμενο στιγμιότυπο θα παραμένει στο $\mathcal{L}(H')$.

- **Βήμα 3:** *Αντικατάσταση των μεταβλητών ελέγχου.*
  Οι μεταβλητές ελέγχου αντικαθίστανται με βάση τη σχέση $\vec{j} = P'\vec{j'}$.

Το θεώρημα που ακολουθεί συνοψίζει την παραπάνω διαδικασία.

Θεώρημα 3.1 Ο παρακάτω n-διάστατος φωλιασμένος βρόχος διασχίζει όλα τα σημεία $\vec{j} \in TIS$ και μόνον αυτά.

```
FOR (j'_1=0, ..., j'_n=0; j'_1 ≤ v_11-1; j'_1+=h̃'_11, ..., j'_n+=h̃'_n1)
   FOR (j'_n+=⌈-j'_2/h̃'_22⌉ * h̃'_n2, ..., j'_2+=⌈-j'_2/h̃'_22⌉ * h̃'_22; j'_2 ≤ v_22-1; j'_2+=h̃'_22, ..., j'_n+=h̃'_n2)
      ...
      FOR (j'_n+=⌈-j'_n/h̃'_nn⌉ * h̃'_nn; j'_n ≤ v_nn-1; j'_n+=h̃'_nn)
         j⃗ = P'j⃗';
         ...
      ENDFOR
      ...
   ENDFOR
ENDFOR
```

Απόδειξη 3.1 *Αρκεί να δείξουμε ότι ο παραπάνω φωλιασμένος βρόχος διασχίζει όλα τα σημεία του TTIS, καθώς λόγω της σχέσης $\vec{j} = P'\vec{j'}$ ισοδύναμα θα διασχίζονται και όλα τα σημεία του TIS. Για να διασχίζει ο βρόχος όλα τα σημεία του TTIS και μόνον αυτά, θα πρέπει να αποδείξουμε τα εξής:*

1. *Σύμφωνα με το Λήμμα 3.3, για κάθε μεταβλητή ελέγχου θα πρέπει να ισχύει $0 \leq j'_k \leq v_{kk} - 1$.*

2. *Τα σημεία που διασχίζονται είναι ακριβή σημεία.*

3. *Δεν υπάρχει σημείο του TTIS που δεν σαρώνεται από τον παραπάνω βρόχο.*

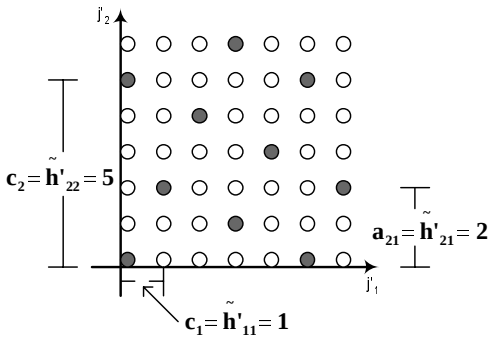
1. Από τα άνω όρια του φωλιασμένου βρόχου είναι προφανές πως η μεταβλητή ελέγχου j_k δεν υπερβαίνει το $v_{kk} - 1$. Και για τη σχέση $j'_k += \lceil \frac{-j'_k}{\tilde{h}'_{kk}} \rceil * \tilde{h}'_{kk}$ που καθορίζει το κάτω όριο της μεταβλητής ελέγχου ισχύει $j'_k + \lceil \frac{-j'_k}{\tilde{h}'_{kk}} \rceil * \tilde{h}'_{kk} \geq 0$. Άρα, $0 \leq j'_k \leq v_{kk} - 1$.

2. Ο παραπάνω φωλιασμένος βρόχος ξεκινά από το σημείο $\vec{0}$ που προφανώς είναι ακριβές

σημείο. Το επόμενο σημείο που διασχίζεται κάθε φορά προκύπτει με βάση τις εκφράσεις $j'_k += \widetilde{h'}_{kk}, \ldots, j'_n += \widetilde{h'}_{nk}$ *και* $j'_n += \lceil \frac{-j'_k}{\widetilde{h'}_{kk}} \rceil * \widetilde{h'}_{nk}, \ldots, j'_k += \lceil \frac{-j'_k}{\widetilde{h'}_{kk}} \rceil * \widetilde{h'}_{kk}$*. Η έκφραση* $j'_k += \lceil \frac{-j'_k}{\widetilde{h'}_{kk}} \rceil * \widetilde{h'}_{kk}$ *θέτει τη μεταβλητή ελέγχου* j'_k *στην ελάχιστη θετική τιμή, ώστε να παραμείνουμε σε ακριβές σημείο και οι εκφράσεις* $j'_n += \lceil \frac{-j'_k}{\widetilde{h'}_{kk}} \rceil * \widetilde{h'}_{nk}, \ldots, j'_{k+1} += \lceil \frac{-j'_k}{\widetilde{h'}_{(k)(k)}} \rceil * \widetilde{h'}_{(k+1)(k)}$ *μεταβάλουν κατάλληλα τις μεταβλητές* $j'_n, \ldots, j'_{k-1}$ *για να προσαρμοστούν στην παραπάνω τροποποίηση. Και οι δύο παραπάνω εκφράσεις μεταβάλλουν το* j' *κατά ένα πολλαπλάσιο μίας στήλης του* $\widetilde{H'}$*. Δεδομένου ότι ξεκινάμε πάντα από ακριβές σημείο, θα καταλήγουμε πάντα σε επίσης ακριβές σημείο (βλ. και Λήμμα 3.4).*

3. Αρκεί να δείξουμε ότι $\forall k = 1, \ldots, n$*, ο φωλιασμένος βρόχος με βάθος* k *διασχίζει όλα τα υπερεπίπεδα διάστασης* $n-k$*. Για παράδειγμα ο εξωτερικότερος βρόχος (βάθος 1) θα διασχίζει όλα τα υπερεπίπεδα διάστασης* $n-1$ *που ανήκουν στον TTIS, ο αμέσως εσωτερικότερος όλα τα υπερεπίπεδα διάστασης* $n-2$ *που ανήκουν στον TTIS κ.ο.κ.*

Έστω ο τυχαίος βρόχος με βάθος k*. Με βάση το φωλιασμένο βρόχο του θεωρήματος, κάθε υπερεπίπεδο διάστασης* $k-1$ *που θα διασχιστεί προκύπτει με βάση τις εκφράσεις* $j'_k += \widetilde{h'}_{kk}, \ldots, j'_n += \widetilde{h'}_{nk}$*, που ουσιαστικά αποτελούν την πρόσθεση του* k*-στού διανύσματος στήλης του* $\widetilde{H'}$*. Από το Λήμμα 3.4 γνωρίζουμε πως δεν μπορεί να υπάρχουν υπερεπίπεδα που να προκύπτουν από κάποιο διάνυσμα λεξικογραφικά μικρότερο από το* $\widetilde{h'}_k$*. Άρα για όλα τα* $k = 1, \ldots, n$ *ο παραπάνω φωλιασμένος βρόχος διασχίζει ολα τα υπερεπίπεδα διάστασης* $n-k$ *που ανήκουν στον TTIS, επομένως διασχίζει όλα τα σημεία του TTIS.* □



Σχήμα 3.6: Βήματα και αρχικές τιμές στον TTIS όπως προκύπτουν από την Ερμητιανή Κανονική Μορφή του H'

Παράδειγμα 3.2 Θα διασχίσουμε τα σημεία που ανήκουν στο Χώρο Επαναλήψεων Υπερκόμβου των προηγούμενων παραδειγμάτων. Υπενθυμίζουμε, ότι $H = \begin{bmatrix} \frac{1}{5} & -\frac{1}{10} \\ -\frac{1}{20} & \frac{3}{20} \end{bmatrix}$. Οι ΜΚΔ των παρονομαστών της πρώτης γραμμής του H είναι 10 και της δεύτερης 20, και έτσι, με βάση την ανάλυση της Ενότητας 3.3.1, θα έχουμε: $H' = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}$ και $V = \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}$. Ανάλογα, $P' = \begin{bmatrix} \frac{3}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{2}{5} \end{bmatrix}$. Η Ερμητιανή Κανονική Μορφή του πίνακα H' είναι $\widetilde{H'} = \begin{bmatrix} 1 & 0 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ και έτσι, όπως φαίνεται και στο Σχήμα 3.6, $c_1 = \widetilde{h'}_{11} = 1$, $c_2 = \widetilde{h'}_{22} = 5$, $a_{21} = \widetilde{h'}_{21} = 2$. Επομένως, ο κώδικας που διασχίζει τα σημεία του Χώρου Επαναλήψεων Υπερκόμβου, σύμφωνα με το Θεώρημα 3.1, είναι:

```
FOR (j'1=0, j'2=0; j'1 ≤9; j'1+=1, j'2+=2)
   FOR (j'2+=⌈−j'2/5⌉*5; j'2 ≤19; j'2+=5)
      ( j1 ) = [ 3/5  1/5 ] ( j'1 );
      ( j2 )   [ 1/5  2/5 ] ( j'2 )
      A[j1,j2]=A[j1-1,j2-2]+A[j1-3,j2-1];
```

```
    ENDFOR
  ENDFOR                                                    ⊣
```

3.3.3 Διάσχιση των Σημείων του Χώρου Επαναλήψεων

Αυτό που απομένει, είναι να προσαρμόσουμε κατάλληλα το φωλιασμένο βρόχο του Θεωρήματος 3.1, που διασχίζει τα σημεία του Χώρου Επαναλήψεων Υπερκόμβου, ώστε να μπορεί να διασχίσει τα εσωτερικά σημεία σε οποιονδήποτε υπερκόμβο. Αν $\vec{j'} \in TTIS$ είναι το σημείο που προκύπτει από τις μεταβλητές ελέγχου του φωλιασμένου βρόχου του Θεωρήματος 3.1 και $\vec{j^S} \in J^S$ είναι ο υπερκόμβος του οποίου τα εσωτερικά σημεία θέλουμε να διασχίσουμε, θα ισχύει $\vec{j} = P\vec{j^S} + P'\vec{j'} \Rightarrow$

$$\vec{j} = \vec{j_0} + P'\vec{j'} \tag{3.8}$$

όπου $\vec{j_0} = P\vec{j^S} \in TOS$, είναι η αρχή του υπερκόμβου και $P'\vec{j'} \in TIS$ είναι το αντίστοιχο του $\vec{j'}$ σημείο στον TIS. Τέλος, ειδική προσοχή πρέπει να δωθεί, ώστε τα σημεία που διασχίζονται να μην προσπερνούν τα όρια του αρχικού χώρου. Όπως αναφέρθηκε προηγουμένως, ένα σημείο $\vec{j} \in J$ ικανοποιεί το σύστημα: $B\vec{j} \leq \vec{b}$. Αντικαθιστώντας το $\vec{j}$ από την παραπάνω εξίσωση έχουμε: $B(\vec{j_0} + P'\vec{j'}) \leq \vec{b} \Rightarrow$

$$BP'\vec{j'} \leq \vec{b} - B\vec{j_0} \tag{3.9}$$

Εφαρμόζοντας τη μέθοδο FME στο παραπάνω σύστημα ανισοτήτων, προκύπτουν κατάλληλες εκφράσεις για το $\vec{j'}$, έτσι ώστε να μην προσπερνώνται τα όρια του αρχικού χώρου. Μ' αυτόν τον τρόπο αντιμετωπίζεται και το πρόβλημα των πλεοναζόντων υπερκόμβων της προηγούμενης ενότητας, καθώς τα σημεία αυτών των υπερκόμβων δεν επαληθεύουν το παραπάνω σύστημα και επομένως δεν θα διασχιστούν.

Παράδειγμα 3.3 Προκειμένου να διασχίσουμε τα εσωτερικά σημεία οποιουδήποτε υπερκόμβου κατασκευάζουμε τον πίνακα $[BP'|\vec{b}|B] = \begin{pmatrix} \frac{3}{5} & \frac{1}{5} & 39 & 1 & 0 \\ \frac{1}{5} & \frac{2}{5} & 29 & 0 & 1 \\ -\frac{3}{5} & -\frac{1}{5} & 0 & -1 & 0 \\ -\frac{1}{5} & -\frac{2}{5} & 0 & 0 & -1 \end{pmatrix}$. Η μέθοδος FME

σ' αυτόν τον πίνακα δίνει $\begin{pmatrix} 1 & 0 & 78 & 2 & -1 \\ \frac{3}{5} & \frac{1}{5} & 39 & 1 & 0 \\ \frac{1}{5} & \frac{2}{5} & 29 & 0 & 1 \\ -1 & 0 & 29 & -2 & 1 \\ -\frac{3}{5} & -\frac{1}{5} & 0 & -1 & 0 \\ -\frac{1}{5} & -\frac{2}{5} & 0 & 0 & -1 \end{pmatrix}$. Επομένως, ο κώδικας που θα διασχίζει τα σημεία για οποιονδήποτε υπερκόμβο του χώρου θα είναι:

$\begin{pmatrix} j_{0_1} \\ j_{0_2} \end{pmatrix} = \begin{bmatrix} 6 & 4 \\ 2 & 8 \end{bmatrix} \begin{pmatrix} j_1^S \\ j_2^S \end{pmatrix}$;
lb_1=max(0,-29-2j_{0_1}+j_{0_2});
ub_1=min(9 /* v_{11}-1 */,78-2j_{0_1}+j_{0_2});
FOR (j'_1=lb_1, j'_2=lb_1*2; $j'_1 \leq ub_1$; j'_1+=1, j'_2+=2)
  lb_2=max(0,-3j'_1-5j_{0_1},$\lceil \frac{-j'_1-5j_{0_2}}{2} \rceil$);
  ub_2=min(19 /* v_{22}-1 */,-3j'_1-5j_{0_1}+195,$\lfloor \frac{-j'_1-5j_{0_2}+145}{2} \rfloor$);
  FOR (j'_2+=$\lceil \frac{lb_2-j'_2}{5} \rceil * 5$; $j'_2 \leq ub_2$; j'_2+=5)
    $\begin{pmatrix} j_1 \\ j_2 \end{pmatrix} = \begin{pmatrix} j_{0_1} \\ j_{0_2} \end{pmatrix} + \begin{bmatrix} \frac{3}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{2}{5} \end{bmatrix} \begin{pmatrix} j'_1 \\ j'_2 \end{pmatrix}$;
    A[j_1,j_2]=A[j_1-1,j_2-2]+A[j_1-3,j_2-1];
  ENDFOR
ENDFOR ⊣

3.4 Σύγκριση με τη Μέθοδο των Ancourt και Irigoin

Η μέθοδος που παρουσιάστηκε, πλεονεκτεί σε σύγκριση με αυτή των Ancourt και Irigoin, τόσο στο χρόνο μεταγλώττισης όσο και στην αποδοτικότητα του παραγόμενου κώδικα. Η διαφορά στο χρόνο μεταγλώττισης προκύπτει από το διαφορετικό μέγεθος των συστημάτων ανισοτήτων που χρησιμοποιούνται για τον υπολογισμό των ορίων του φωλιασμένου βρόχου που διασχίζει τους υπερκόμβους. Όπως αναφέρθηκε και στην Ενότητα 3.2, ο σύστημα ανισοτήτων που προτείνουμε για τη διάσχιση των υπερκόμβων (σύστημα (3.2)), είναι πολύ μικρότερο από αυτό που προτείνουν οι Ancourt και Irigoin (σύστημα (2.8)). Συγκεκριμένα, αν ο αρχικός χώρος είναι n-διάστατος και περιγράφεται από p ανισότητες (οι γραμμές του πίνακα B είναι p), τότε το σύστημα (2.8) αποτελείται από $2n+p$ ανισότητες των $2n$ μεταβλητών, ενώ το προτεινόμενο σύστημα (3.2) είναι πολύ μικρότερο, καθώς αποτελείται από p ανισότητες των n μεταβλητών. Αυτή η μεγάλη διαφορά στο μέγεθος των δύο συστημάτων, οδηγεί σε πάρα πολύ μεγάλη διαφορά

γραμμοπράξεων κατά την απαλοιφή των συστημάτων με τη μέθοδο FME. Τελικά, ο χρόνος μεταγλώττισης με την προτεινόμενη μέθοδο είναι πάρα πολύ μικρότερος, όπως φαίνεται και από τα πειραματικά αποτελέσματα που παρουσιάζονται στο Κεφάλαιο 5.

Όσον αφορά την αποδοτικότητα του παραγόμενου κώδικα, η προτεινόμενη μέθοδος υπερέχει αυτής των Ancourt και Irigoin. Αυτό συμβαίνει, διότι προκειμένου να διασχίσει τα εσωτερικά σημεία ενός υπερκόμβου, η προτεινόμενη μέθοδος υπολογίζει τα όρια των μεταβλητών ελέγχου με βάση το Θεώρημα 3.1, που είναι πολύ πιο απλές από αυτές που χρησιμοποιεί η μέθοδος των Ancourt και Irigoin $\begin{pmatrix} gH \\ -gH \end{pmatrix} \vec{j} \leq \begin{pmatrix} (g-1)\vec{1} \\ \vec{0} \end{pmatrix}$. Το γεγονός αυτό μπορεί να γίνει εύκολα αντιληπτό συγκρίνοντας τους δύο εσωτερικότερους βρόχους στον κώδικα του Παραδείγματος 2.7 που έχει προκύψει με την εφαρμογή της μεθόδου των Ancourt και Irigoin, με τον κώδικα του Παραδείγματος 3.3 που έχει προκύψει με την εφαρμογή της μεθόδου που παρουσιάστηκε στο παρών κεφάλαιο. Παρατηρούμε ότι για τη διάσχιση των εσωτερικών σημείων των υπερκόμβων στην πρώτη περίπτωση, χρησιμοποιούνται ιδιαίτερα πολύπλοκες εκφράσεις για την αποτίμηση των μεταβλητών ελέγχου του φωλιασμένου βρόχου. Αντίθετα, στον κώδικα του Παραδείγματος 3.3 οι εκφράσεις αποτίμησης είναι πολύ πιο απλές. Επιπρόσθετα βέβαια σ' αυτή την περίπτωση υπάρχουν αρκετές πράξεις πολλαπλασιασμού, οι οποίες όμως εκτελούνται πολύ πιο αποδοτικά στους σύγχρονους επεξεργαστές από τις εκφράσεις `min` και `max`.

Ποσοτικά μπορούμε να συγκρίνουμε την αποδοτικότητα των δύο μεθόδων ως εξής: Αν εφαρμόσουμε τη μέθοδο FME στο σύστημα ανισοτήτων $\begin{pmatrix} gH \\ -gH \end{pmatrix} \vec{j} \leq \begin{pmatrix} (g-1)\vec{1} \\ \vec{0} \end{pmatrix}$ που περιέχει 2n ανισότητες με n αγνώστους, τότε σύμφωνα με την υλοποίηση της μεθόδου (βλ. Παράρτημα Α) θα δημιουργηθούν n ανισότητες με n αγνώστους για τον υπολογισμό του κάτω ορίου της μεταβλητής ελέγχου σε βάθος n, $O(n^2)$ ανισότητες με $n-1$ αγνώστους για τον υπολογισμό του κάτω ορίου της μεταβλητής ελέγχου σε βάθος $n-1$ και γενικά $O(n^{2^{n-k}})$ ανισότητες με k αγνώστους για τον προσδιορισμό του κάτω ορίου της μεταβλητής σε βάθος k. Ομοίως και για τα άνω όρια. Ο προσδιορισμός των ορίων της μεταβλητής x_k από μία ανισότητα με k μεταβλητές της μορφής $a_1x_1 + a_2x_2 + \ldots, a_kx_k \leq d$, απαιτεί k πολλαπλασιασμούς, $k-1$ προσθέσεις και μία διαίρεση. Επίσης, για τον προσδιορισμό των κάτω (άνω) ορίων της μεταβλητής σε βάθος k απαιτούνται $O(n^{2^{n-k}})$ συγκρίσεις για την εύρεση της μέγιστης (ελάχιστης) τιμής των ανισοτήτων που συμμετέχουν. Έτσι, ενώ για την αρχικοποίηση ενός βρόχου σε βάθος k στο φωλιασμένο βρόχο του Θεωρήματος 3.1 απαιτούνται μόνο $n-k+1$ προσθέσεις, $n-k+1$ πολλαπλασιασμοί και μία διαίρεση, για την αντίστοιχη αρχικοποίηση στο βρόχο που προκύπτει για τη μέθοδο των Ancourt και Irigoin, απαιτούνται $O(kn^{2^{n-k}})$ προσθέσεις, $O(kn^{2^{n-k}})$ πολλαπλασιασμοί, μία διαίρεση και $O(n^{2^{n-k}})$ συγκρίσεις. Κατά την αύξηση της

μεταβλητής ελέγχου σε βάθος k η προτεινόμενη μέθοδος απαιτεί $n-k+1$ προσθέσεις και η μέθοδος των Ancourt και Irigoin μία πρόσθεση, διαφορά που όμως δεν επιβαρύνει αισθητά τον κώδικα του Θεωρήματος 3.1.

Για να αποκτήσουμε μία καλύτερη ιδέα για την αποδοτικότητα του παραγόμενου κώδικα, εκτελέσαμε τον σειριακό μετασχηματισμένο κώδικα που προκύπτει από την εφαρμογή της προτεινόμενης μεθόδου στο παράδειγμα της Ενότητας 3.1. Ο χρόνος εκτέλεσης σ' αυτή την περίπτωση ήταν $3.97sec$ εκ των οποίων τα $1.73sec$ καταναλώθηκαν στην αποτίμηση των ορίων του μετασχηματισμένου φωλιασμένου βρόχου. Σε σύγκριση με τα αποτελέσματα που παρουσιάζονται στον Πίνακα 3.1, η προτεινόμενη μέθοδος παράγει σειριακό μετασχηματισμένο κώδικα που εκτελείται σχεδόν στο μισό χρόνο από αυτόν που χρειάζεται ο κώδικας που έχει παραχθεί με τη μέθοδο των Ancourt και Irigoin. Βέβαια, όπως αναμενόταν, ο παραγόμενος κώδικας ακόμα υστερεί σε σχέση με τον κώδικα που προκύπτει από την εφαρμογή ορθογώνιου μετασχηματισμού υπερκόμβων. Στο Κεφάλαιο 5, παρουσιάζονται επίσης συγκριτικά αποτελέσματα των δύο μεθόδων σε σχέση και με την αποδοτικότητα του σειριακού μετασχηματισμένου κώδικα.

Κεφάλαιο 4

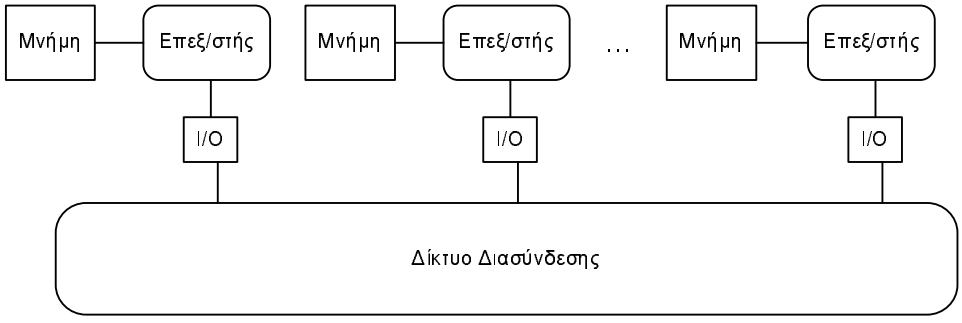
Παραλληλοποίηση

Στο κεφάλαιο αυτό θα ασχοληθούμε με την παραλληλοποίηση του σειριακού μετασχηματισμένου κώδικα, ο οποίος έχει παραχθεί ακολουθώντας τη μέθοδο που παρουσιάστηκε στο Κεφάλαιο 3. Η παραλληλοποίηση περιλαμβάνει ενδιάμεσες διαδικασίες όπως η ανάθεση των επαναλήψεων σε επεξεργαστές, η κατανομή των δεδομένων και η παραγωγή πρωτογενών κλήσεων επικοινωνίας (Σχήμα 1.2). Στόχος της μεθόδου που θα προταθεί, είναι η απλοποίηση της παραλληλοποίησης, ώστε ο τελικός κώδικας να είναι όσο το δυνατόν πιο απλός και αποδοτικός. Η γενική ιδέα στη διαδικασία παραλληλοποίησης στηρίζεται στην αξιοποίηση των ιδιοτήτων του ορθογώνιου Μετασχηματισμένου Χώρου Επαναλήψεων Υπερκόμβου. Έτσι, κάθε επεξεργαστής τοπικά έχει τη δυνατότητα να διατρέχει ορθογώνιους υπερκόμβους και να προσπελαύνει ορθογώνιους χώρους μνήμης. Στο τέλος των υπολογισμών, αν είναι απαραίτητο από τη φύση του προς επίλυση προβλήματος, ο επεξεργαστής θα μπορεί να αντιστοιχίζει τα τοπικά υπολογισμένα δεδομένα με τα καθολικά δεδομένα.

Θα ξεκινήσουμε την παρουσίαση της μεθόδου παραλληλοποίησης με την περιγραφή του γενικού μοντέλου της παράλληλης αρχιτεκτονικής στην οποία θα εκτελεστεί ο παραγόμενος παράλληλος κώδικας. Στη συνέχεια θα αναφερθούμε σε ζητήματα που έχουν να κάνουν με την αποδοτική ανάθεση των επαναλήψεων του μετασχηματισμένου βρόχου σε επεξεργαστές και την κατανομή των δεδομένων. Τέλος, με βάση τα παραπάνω, θα δείξουμε πώς είναι δυνατόν να παραχθούν αυτόματα πρωτογενείς κλήσεις επικοινωνίας που θα εξασφαλίζουν την ορθή εκτέλεση του παραλληλοποιημένου προγράμματος.

4.1 Τελική Παράλληλη Αρχιτεκτονική

Τα ιδιαίτερα χαρακτηριστικά της παράλληλης αρχιτεκτονικής στην οποία θα εκτελεστεί ο τελικός κώδικας, προφανώς επηρεάζουν σε πολύ μεγάλο βαθμό τη διαδικασία παραγωγής του. Στην παρούσα εργασία αναφερόμαστε σε αρχιτεκτονικές κατανεμημένης μνήμης, στις οποίες ο κάθε επεξεργαστής έχει πρόσβαση αποκλειστικά και μόνο στη δική του τοπική μνήμη, ενώ διασυνδέεται με τους υπόλοιπους επεξεργαστές του συστήματος με ένα δίκτυο διασύνδεσης (Σχήμα 4.1). Θεωρείται ότι δεν έχει υλοποιηθεί στην εν λόγω αρχιτεκτονική κάποιο σχήμα Κατανεμημένης Κοινής Μνήμης (Distributed Shared Memory-DSM) ούτε στο υλικό ούτε στο λογισμικό. Σε ένα σύστημα σαν αυτό, οι επεξεργαστές επικοινωνούν με ανταλλαγή μηνυμάτων (message-passing), με τη βοήθεια δηλαδή βιβλιοθηκών επικοινωνίας που υλοποιούνται πάνω από το υλικό του δικτύου διασύνδεσης. Χαρακτηριστικά παραδείγματα τέτοιας αρχιτεκτονικής είναι ο Intel Paragon, ο IBM SP2, ο Cray T3D/T3E, ο SGI PowerChallenge και ο Convex Exemplar, καθώς και όλες οι συστοιχίες προσωπικών υπολογιστών (PC Clusters), που τα τελευταία χρόνια αποτελούν μία πολύ δυναμικά ανερχόμενη πλατφόρμα για την εκτέλεση υπολογιστικά απαιτητικών εφαρμογών.



Σχήμα 4.1: Τελική αρχιτεκτονική κατανεμημένης μνήμης

Ο λόγος για τον οποίο επιλέξαμε την παραπάνω τελική αρχιτεκτονική είναι απλός. Τα συστήματα κατανεμημένης μνήμης που δεν διαθέτουν DSM, αποτελούν την δυσκολότερη τελική αρχιτεκτονική για αυτόματη παραγωγή παράλληλου κώδικα. Εφ' όσον οι επεξεργαστές δεν μοιράζονται χώρο μνήμης, δεν μπορούν να συγχρονιστούν ή να ανταλλάξουν δεδομένα με απλές εντολές ανάγνωσης/εγγραφής στη μοιραζόμενη μνήμη. Αντίθετα, είναι αναγκαίο να γεννηθούν αυτόματα πρωτογενείς κλήσεις επικοινωνίας, που θα ορίζουν σαφώς τους επεξεργαστές οι οποίοι θα συμμετάσχουν και ακριβώς τα δεδομένα που πρέπει να αποσταλούν και να ληφθούν.

Επιπρόσθετα, πρέπει να ληφθεί ειδική μέριμνα, ώστε τα δεδομένα να διαμοιράζονται στις τοπικές μνήμες των επεξεργαστών του συστήματος. Είναι μάλλον εμφανές, πως ο κώδικας για ένα παράλληλο σύστημα με μοιραζόμενη μνήμη (είτε φυσικά κοινή, είτε μέσω κάποιου συστήματος DSM), μπορεί εύκολα να προκύψει από τον κώδικα για ένα σύστημα με κατανεμημένη μνήμη, με αντικατάσταση των κλήσεων αποστολής/λήψης δεδομένων από εγγραφές/αναγνώσεις στη μνήμη και προσθήκη κατάλληλου συγχρονισμού.

Ο τελικός παράλληλος κώδικας θα είναι της μορφής SPMD (Single Program Multiple Data), δηλαδή όλοι οι επεξεργαστές θα εκτελούν κοινό κώδικα, ο οποίος διαφοροποιείται κατά το χρόνο εκτέλεσης, ανάλογα με το μοναδικό αναγνωριστικό κάθε επεξεργαστή. Στην παρούσα εργασία, θεωρούμε ότι οι επεξεργαστές είναι οργανωμένοι σε ένα $(n-1)$-διάστατο πλέγμα, έτσι κάθε επεξεργαστής χαρακτηρίζεται από ένα διάνυσμα $n-1$ διαστάσεων που θα συμβολίζουμε με $\vec{pid}$. Αυτή η παραδοχή δεν αποτελεί φυσικό περιορισμό για τη μέθοδο που θα παρουσιαστεί, είναι απλά μία σύμβαση για την ονοματολογία των επεξεργαστών. Όλες οι δημοφιλείς τοπολογίες συστημάτων κατανεμημένης μνήμης μπορούν εύκολα να απεικονιστούν σε ένα πλέγμα k διαστάσεων.

4.2 Ανάθεση Επαναλήψεων σε Επεξεργαστές και Δρομολόγηση

Στην ενότητα αυτή θα ασχοληθούμε με την ανάθεση των υπολογισμών στους επεξεργαστές της αρχιτεκτονικής που περιγράφηκε παραπάνω. Από τους $2n$ βρόχους που περιέχει ο σειριακός μετασχηματισμένος κώδικας, οι n εσωτερικότεροι διασχίζουν τα εσωτερικά σημεία των υπερκόμβων και, όπως έχει ειπωθεί εξ αρχής, δεν παραλληλοποιούνται αλλά αντίθετα αντιμετωπίζονται σαν μία μεγάλη μονάδα υπολογισμών. Οι n εξωτερικότεροι βρόχοι, που αποτελούν ένα `DOACROSS` φώλιασμα, θα ανατεθούν σε επεξεργαστές και θα εκτελεστούν παράλληλα. Συγκεκριμένα, κάθε επεξεργαστής θα αναλάβει την εκτέλεση υπολογισμών σε μία γραμμή από υπερκόμβους. Η διάσταση κατά μήκος της οποίας γίνεται απεικόνιση των υπερκόμβων στον ίδιο επεξεργαστή λέγεται *διάσταση απεικόνισης*.

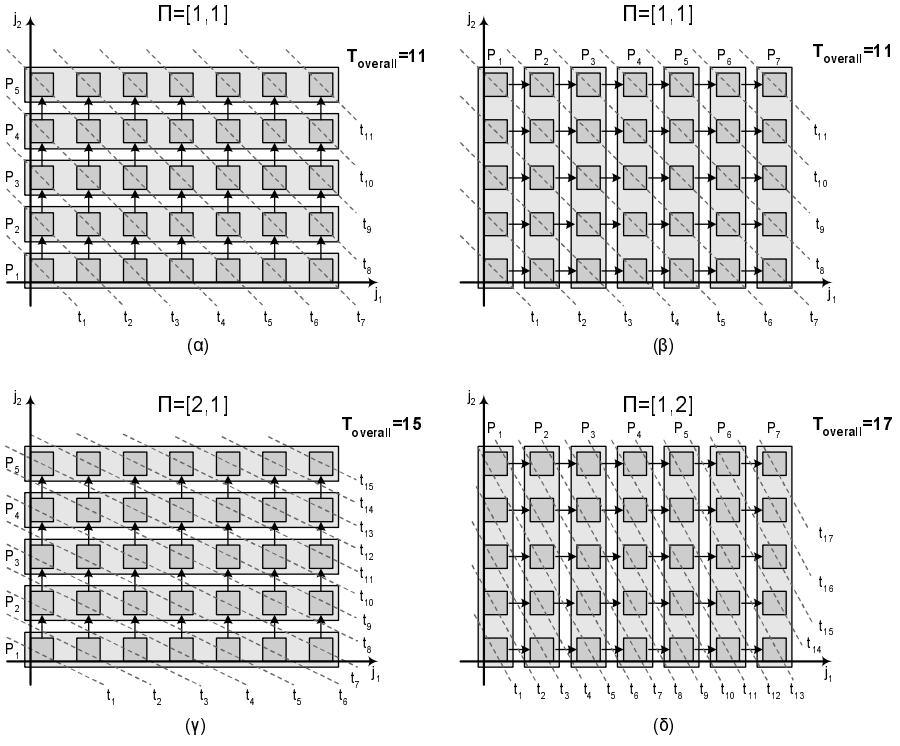
Η επιλογή της διάστασης απεικόνισης αλλά και του σχήματος δρομολόγησης επηρεάζει σε μεγάλο βαθμό την απόδοση του παραλληλοποιημένου προγράμματος. Σε σχέση με τη δρομολόγηση των υπερκόμβων σε επεξεργαστές μας απασχολούν δύο βασικά σχήματα: Κατά το πρώτο σχήμα, η επικοινωνία και ο υπολογισμός γίνονται με την κλασσική διαδοχή *λήψη-υπολογισμός-αποστολή*, κατά την όποια ο επεξεργαστής αναμένει τα δεδομένα που θα χρειαστεί για τη διεξαγωγή των υπολογισμών του τρέχοντος υπερκόμβου, διεξάγει τους υπολογισμούς και τέλος

στέλνει τα δεδομένα που μόλις υπολόγισε στους επεξεργαστές που τα χρειάζονται. Υπάρχει όμως και ένα εναλλακτικό, εξελιγμένο σχήμα δρομολόγησης, που επιτρέπει στον επεξεργαστή να διεξάγει ταυτόχρονα επικοινωνία και υπολογισμούς. Στο σχήμα αυτό, ο επεξεργαστής πραγματοποιεί ταυτόχρονα τις εξής τρεις λειτουργίες: (α) λαμβάνει τα δεδομένα που θα χρειαστεί στον επόμενο υπερκόμβο (τα δεδομένα που χρειάζονται για τον τρέχοντα υπερκόμβο έχουν ληφθεί κατά την προηγούμενη φάση) (β) στέλνει τα δεδομένα που έχει υπολογίσει κατά τον προηγούμενο υπερκόμβο και (γ) διεξάγει υπολογισμούς για τον τρέχοντα υπερκόμβο.

Το παραπάνω εξελιγμένο σχήμα δρομολόγησης είναι ανάλογο με τη δρομολόγηση φωλιασμένων βρόχων UET-UCT (Unit Execution Time-Unit Communication Time), όπως αυτό παρουσιάζεται στην εργασία [AKPT99]. Στη συγκεκριμένα εργασία αποδεικνύεται, ότι για να υλοποιηθεί το σχήμα δρομολόγησης που επιτρέπει την επικάλυψη επικοινωνίας και υπολογισμών, τότε το διάνυσμα δρομολόγησης Π θα πρέπει να είναι της μορφής $\Pi = \{2, 2, \ldots, 1, \ldots, 2, 2\}$, με τη μονάδα στη διάσταση απεικόνισης. Αντίθετα, το κλασσικό σχήμα δρομολόγησης χωρίς επικάλυψη υπολογισμών και επικοινωνίας υλοποιείται με το διάνυσμα δρομολόγησης $\Pi = \{1, 1, \ldots, 1\}$.

Στο Σχήμα 4.2 βλέπουμε πώς επηρεάζει το συνολικό χρόνο εκτέλεσης και τον αριθμό των απαιτούμενων επεξεργαστών η διάσταση της απεικόνισης και το σχήμα της δρομολόγησης. Οι δύο περιπτώσεις στο πάνω μέρος του σχήματος ((α) και (β)), αντιστοιχούν σε δρομολόγηση χωρίς επικάλυψη επικοινωνίας και υπολογισμών. Παρατηρούμε πως σ' αυτή την περίπτωση η επιλογή της διάστασης απεικόνισης δεν επηρεάζει το συνολικό χρόνο εκτέλεσης, επηρεάζει όμως τον αριθμό των επεξεργαστών που απαιτούνται. Αν απεικονίσουμε στον ίδιο επεξεργαστή υπερκόμβους κατά μήκος της διάστασης με το μεγαλύτερο μήκος, τότε μειώνουμε τον αριθμό των απαιτούμενων επεξεργαστών. Οι δύο περιπτώσεις στο κάτω μέρος του σχήματος ((γ) και (δ)), αντιστοιχούν σε δρομολόγηση με επικάλυψη υπολογισμού και επικοινωνίας. Τα βέλη στο σχήμα υποδηλώνουν επικοινωνία, ενώ οι γκρι γραμμές που τέμνουν υπερκόμβους και βέλη υποδηλώνουν την ταυτόχρονη διεξαγωγή υπολογισμών στους υπερκόμβους και επικοινωνίας στα αντίστοιχα βέλη. Σ' αυτή την περίπτωση, η επιλογή της διάστασης απεικόνισης επηρεάζει τον αριθμό των απαιτούμενων επεξεργαστών, αλλά κυρίως επηρεάζει το συνολικό χρόνο εκτέλεσης. Παρατηρούμε ότι αν η απεικόνιση γίνει κατά μήκος της διάστασης με το μεγαλύτερο μήκος, τότε έχουμε ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης. Πρέπει επίσης να σημειωθεί, ότι τα χρονικά βήματα στις περιπτώσεις (α) και (β) είναι διαφορετικά από αυτά των περιπτώσεων (γ) και (δ), αφού στην πρώτη περίπτωση μεταξύ δύο χρονικών στιγμών μεσολαβεί και χρόνος επικοινωνίας, ενώ στη δεύτερη περίπτωση, κάθε χρονικό βήμα περιλαμβάνει και την επικοινωνία. Αυτό σημαίνει ότι τα 15 βήματα της περίπτωσης (γ) θα ολοκληρωθούν πιο γρή-

γορα από τα 11 των περιπτώσεων (α) και (β). Η υλοποίηση του σχήματος δρομολόγησης που επιτρέπει επικάλυψη υπολογισμών και επικοινωνίας μελετάται στις εργασίες [GSK01], [STK01] και [STK02].



Σχήμα 4.2: Συνολικοί χρόνοι εκτέλεσης για διαφορετικά σχήματα δρομολόγησης και διαστάσεις απεικόνισης: (α) Δρομολόγηση χωρίς επικάλυψη κατά μήκος της διάστασης j_1 (β) Δρομολόγηση χωρίς επικάλυψη κατά μήκος της διάστασης j_2 (γ) Δρομολόγηση με επικάλυψη κατά μήκος της διάστασης j_1 (δ) Δρομολόγηση με επικάλυψη κατά μήκος της διάστασης j_2

Αν συμβολίσουμε με m τη διάσταση απεικόνισης, σύμφωνα με τα παραπάνω όλοι οι υπερκόμβοι $\vec{j^S} = (j_1^S, \ldots, j_m^S, \ \ldots, j_n^S)$, όπου $j_i^S = \sigma\tau\alpha\theta.$, $i = 1, \ldots, m-1, m+1, \ldots, n$ και $l_m^S \leq j_m^S \leq u_m^S$ θα εκτελεστούν στον ίδιο επεξεργαστή. Οι $n-1$ συντεταγμένες του υπερκόμβου (αφαιρώντας την j_m^S) καθορίζουν τον επεξεργαστή, στον οποίο θα εκτελεστεί ο συγκεκριμένος υπερκόμβος ($\vec{pid}$). Όλοι οι υπερκόμβοι κατά μήκος της j_m^S (συμβολίζεται επίσης και ως t^S) εκτελούνται σειριακά από τον ίδιο επεξεργαστή, διαδοχικά, με τη σειρά που υπαγορεύει η χρονική δρομολόγηση που έχει επιλεγεί. Μετά την επιλογή του δείκτη j_m^S, μεταθέτουμε τους n εξωτερικότερους βρόχους του σειριακού μετασχηματισμένου κώδικα, ούτως ώστε ο βρόχος

j_m^S να βρεθεί σε βάθος n. Επειδή όλα τα μετασχηματισμένα διανύσματα εξάρτησης $\vec{d^S}$ στο J^S αποτελούνται από μη αρνητικές συντεταγμένες, η παραπάνω μετάθεση είναι έγκυρη (βλ. Ενότητες 2.4 και 2.8). Τέλος, αξίζει να σημειωθεί, ότι θεωρούμε την ύπαρξη άπειρων επεξεργαστών για την απεικόνιση των υπερκόμβων. Ουσιαστικά, κατά την παραγωγή κώδικα θεωρούμε ότι οι επεξεργαστές αντιστοιχούν σε διεργασίες, οι οποίες στη φάση της εκτέλεσης μπορούν να ομαδοποιηθούν και να εκτελεστούν σε πεπερασμένο αριθμό επεξεργαστών. Τεχνικές για τη βέλτιστη απεικόνιση των διεργασιών αυτών σε επεξεργαστές είναι πέραν του αντικειμένου της παρούσας εργασίας και αποτελούν αντικείμενο μελλοντικής ερευνητικής εργασίας.

4.3 Κατανομή Δεδομένων

Ένα από τα σημαντικότερα ζητήματα κατά την παραλληλοποίηση ενός αλγορίθμου σε μια αρχιτεκτονική κατανεμημένης μνήμης, είναι η κατανομή των δεδομένων στις τοπικές μνήμες των επεξεργαστών του συστήματος. Με βάση το μοντέλο των αλγορίθμων που μας αφορούν, όπως το παρουσιάσαμε στην Ενότητα 2.8, κατά την κατανομή των δεδομένων θα πρέπει ουσιαστικά να δεσμευτεί κατάλληλος χώρος μνήμης σε κάθε επεξεργαστή, προκειμένου να αποθηκευτούν τα δεδομένα που υπολογίζονται σε κάθε επανάληψη του φωλιασμένου βρόχου. Επιπρόσθετα, λόγω των αλγοριθμικών εξαρτήσεων, κάποια από τα δεδομένα που υπολογίζονται σε έναν επεξεργαστή και αποθηκεύονται στην τοπική του μνήμη, θα χρειαστούν για τον υπολογισμό σε κάποιο γειτονικό επεξεργαστή. Έτσι, η κατανομή των δεδομένων περιλαμβάνει και τη δέσμευση χώρου για την αποθήκευση των δεδομένων που θα ληφθούν από άλλους επεξεργαστές.

Στην προσέγγισή μας θα ακολουθήσουμε τον κανόνα *ο υπολογίζων κατέχει* (computer-owns rule), ο οποίος υπαγορεύει ότι κάθε επεξεργαστής θα πρέπει να κατέχει τοπικά τα δεδομένα τα οποία υπολογίζει. Επομένως, επικοινωνία θα απαιτείται κάθε φορά που ένας επεξεργαστής χρειάζεται να διαβάσει δεδομένα που έχουν υπολογιστεί σε απομακρυσμένο επεξεργαστή. Άρα, δεδομένου ότι κάθε επεξεργαστής διατρέχει μία σειρά από ολισθημένους Μετασχηματισμένους Χώρους Επαναλήψεων Υπερκόμβου, οι οποίοι με βάση τα όσα ειπώθηκαν στο Κεφάλαιο 3 είναι ορθογώνιοι χώροι, τότε θα μπορεί να προσπελαύνει και χώρους δεδομένων οι οποίοι είναι επίσης ορθογώνιοι. Μετά το πέρας όλων των υπολογισμών, κάθε επεξεργαστής θα μπορεί να τοποθετήσει τα τοπικά υπολογισμένα δεδομένα στις κατάλληλες θέσεις του Χώρου Δεδομένων DS.

Ο χώρος δεδομένων που θα δεσμευτεί τοπικά σε έναν επεξεργαστή, για την αποθήκευση των δεδομένων που υπολογίζονται σε έναν υπερκόμβο, θα μπορούσε να είναι μια ακριβής εικόνα του

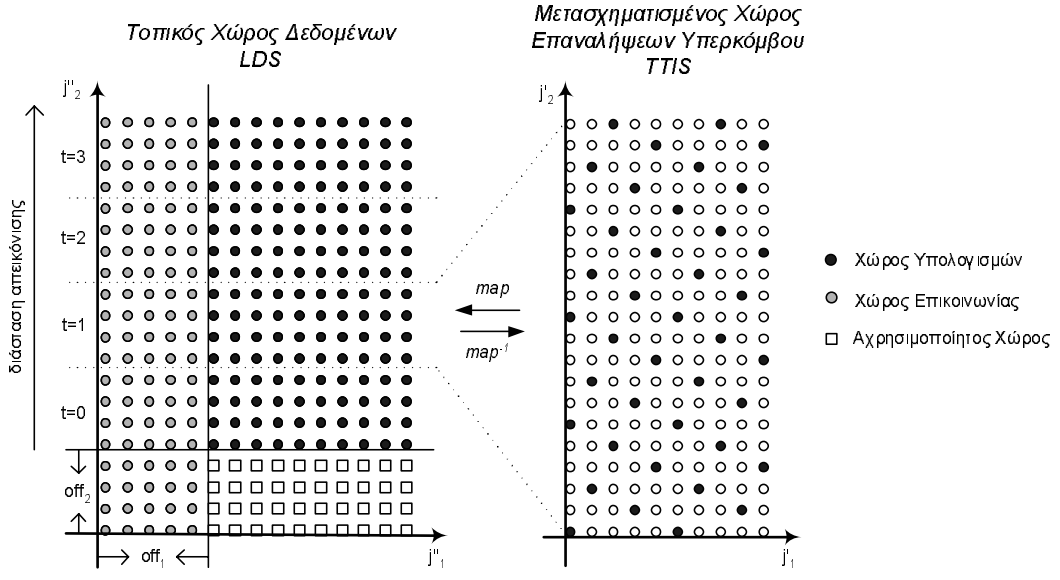
Μετασχηματισμένου Χώρου Επαναλήψεων Υπερκόμβου, αλλά σ' αυτή την περίπτωση οι οπές του χώρου αυτού θα αντιπροσώπευαν χώρο μνήμης που δεν θα χρησιμοποιηθεί. Επιπρόσθετα, κάθε επεξεργαστής θα πρέπει να δεσμεύσει και κατάλληλο χώρο μνήμης για την επικοινωνία, δηλαδή για τα δεδομένα που θα λάβει από γειτονικούς επεξεργαστές. Άρα ο χώρος μνήμης που απαιτείται από κάθε υπερκόμβο, προκύπτει αν συμπυκνωθεί κατάλληλα ο Μετασχηματισμένος Χώρος Επαναλήψεων Υπερκόμβου και στη συνέχεια επεκταθεί για να συμπεριλάβει και τα δεδομένα επικοινωνίας. Τελικά, ο Τοπικός Χώρος Δεδομένων (Local Data Space-LDS) δίνεται από τον ορισμό που ακολουθεί.

Ορισμός 4.1 *Ο Τοπικός Χώρος Δεδομένων (LDS) ορίζεται ως:*

$$\begin{aligned} LDS &= \{\vec{j''} \in Z^n \mid 0 \leq j''_k < off_k + v_{kk}/\widetilde{h'}_{kk},\ k = 1, \ldots, n, k \neq m \\ &\wedge \quad 0 \leq j''_m < off_m + |t| v_{mm}/\widetilde{h'}_{mm}\} \end{aligned} \tag{4.1}$$

όπου το $|t|$ υποδηλώνει τον αριθμό των υπερκόμβων που ανατίθενται στο συγκεκριμένο επεξεργαστή.

Όπως φαίνεται και στο Σχήμα 4.3, ο Τοπικός Χώρος Δεδομένων ενός επεξεργαστή αποτελείται από το χώρο που απαιτείται για να αποθηκευτούν τα δεδομένα που υπολογίζονται (μαύρα σημεία) και από το χώρο που χρησιμοποιείται για την αποθήκευση των δεδομένων επικοινωνίας (γκρι σημεία) ενός υπερκόμβου, πολλαπλασιασμένους με τον αριθμό των υπερκόμβων που έχουν ανατεθεί στο συγκεκριμένο επεξεργαστή. Τα λευκά τετράγωνα παριστάνουν χώρο που δεν θα χρησιμοποιηθεί. Οι παράγοντες off_k, με βάση τους οποίους διευρύνουμε τον Τοπικό Χώρο Δεδομένων, προκύπτουν από τα κριτήρια επικοινωνίας του αλγορίθμου, όπως θα φανεί στην επόμενη ενότητα. Αξίζει να σημειωθεί, ότι το $|t|$ που υποδηλώνει τον αριθμό των υπερκόμβων που ανατίθενται σε έναν επεξεργαστή, μπορεί να υπολογιστεί κατά τη διάρκεια της μεταγλώττισης με κατάλληλη εφαρμογή της μεθόδου FME. Το Λήμμα 4.1 ορίζει τη συνάρτηση μετάβασης από το Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου στον Τοπικό Χώρο Δεδομένων, ενώ το Λήμμα 4.2 ορίζει την αντίστροφη συνάρτηση μετάβασης από τον Τοπικό Χώρο Δεδομένων στο Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου.



Σχήμα 4.3: Τοπικός Χώρος Δεδομένων (LDS) και Μετασχηματισμένος Χώρος Επαναλήψεων Υπερκόμβου ($TTIS$)

Λήμμα 4.1 *Αν $\vec{j'} \in TTIS$, τότε το αντίστοιχο σημείο του στον Τοπικό Χώρο Δεδομένων LDS δίνεται από τη συνάρτηση $\vec{j''} = map(\vec{j'}, t)$ ως εξής:*

$$\vec{j''} = map(\vec{j'}, t) = \begin{cases} j''_k = j'_k / \widetilde{h'}_{kk} + off_k, k \neq m \\ j''_m = (tv_{mm} + j'_m)/\widetilde{h'}_{mm} + off_m \end{cases}$$

όπου το t συμβολίζει τον τρέχοντα υπερκόμβο ($t \geq 0$).

Απόδειξη 4.1 *Προκειμένου να αποδείξουμε την ισχύ του λήμματος, αρκεί να αποδείξουμε ότι το σημείο που προκύπτει από τη συνάρτηση $map(\vec{j'}, t) \in LDS$. Από το Λήμμα 3.3 γνωρίζουμε ότι για κάθε $k \neq m$ ισχύει $0 \leq j'_k < v_{kk} \Rightarrow 0 \leq \lfloor \frac{j'_k}{h'_{kk}} \rfloor < \frac{v_{kk}}{h'_{kk}} \Rightarrow off_k \leq \lfloor \frac{j'_k}{h'_{kk}} \rfloor + off_k <$*

$\frac{v_{kk}}{\widetilde{h}'_{kk}} + off_k \Rightarrow off_k \leq j''_k < \frac{v_{kk}}{\widetilde{h}'_{kk}} + off_k$. *Επιπρόσθετα,* $0 \leq j'_m < v_{mm} \Rightarrow 0 \leq \lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor <$ $\frac{v_{mm}}{\widetilde{h}'_{mm}} \Rightarrow \frac{tv_{mm}}{\widetilde{h}'_{mm}} + off_m \leq \frac{tv_{mm}}{\widetilde{h}'_{mm}} + \lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor + off_m < \frac{(t+1)v_{mm}}{\widetilde{h}'_{mm}} + off_m$. *Αν λάβουμε υπ' όψιν ότι* $0 \leq t \leq |t|-1$, *η προηγούμενη ανισότητα δίνει* $off_m \leq j''_m < \frac{|t|v_{mm}}{\widetilde{h}'_{mm}} + off_m$. *Επομένως ισχύει* $\vec{j''} = map(\vec{j'}, t) \in LDS$. □

Λήμμα 4.2 *Αν* $\vec{j''} \in LDS$, *τότε το αντίστοιχο σημείο του στο χώρο* $TTIS$ *θα δίνεται από τη συνάρτηση* $\vec{j'} = map^{-1}(\vec{j''})$ *ως εξής:*

$$\vec{j'} = map^{-1}(\vec{j''}) = \widetilde{H'}\vec{x}$$

όπου $\vec{x}$ *δίνεται από την έκφραση:*

$$x_k = j''_k - off_k - (\textstyle\sum_{l=1}^{k-1} x_l \widetilde{h}'_{kl})/\widetilde{h}'_{kk},\ k \neq m$$

$$x_m = j''_m - off_m - tv_{mm}/\widetilde{h}'_{mm} - (\textstyle\sum_{l=1}^{m-1} x_l \widetilde{h}'_{ml})/\widetilde{h}'_{mm}$$

όπου το t *συμβολίζει τον τρέχοντα υπερκόμβο και υπολογίζεται από την έκφραση* $t = (j''_m - off_m)\widetilde{h}'_{mm}/v_{mm}$.

Απόδειξη 4.2 *Αρκεί να δείξουμε ότι οι συναρτήσεις* $map()$ *και* $map^{-1}()$ *είναι αντίστροφες συναρτήσεις. Ισοδύναμα, πρέπει να αποδείξουμε ότι:*

1. $(\vec{j'}, t) = map^{-1}(map(\vec{j'}, t))$.

2. $\vec{j''} = map(map^{-1}(\vec{j''}))$.

1. $(\vec{j'}, t) = map^{-1}(map(\vec{j'}, t)) \Leftrightarrow \begin{cases} t = \lfloor \frac{((\lfloor \frac{tv_{mm}+j'_m}{\widetilde{h}'_{mm}} \rfloor + off_m) - off_m)\widetilde{h}'_{mm}}{v_{mm}} \rfloor & (\alpha) \\ j'_k = \sum_{i=1}^{k} \widetilde{h}'_{ki} y_i & (\beta) \end{cases}$

όπου:

$$\left.\begin{array}{c} y_i = ((\lfloor \frac{j'_i}{\widetilde{h}'_{ii}} \rfloor + off_i) - off_i) - \lfloor \frac{\sum_{l=1}^{i-1} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor, i \neq m \\ y_m = ((\lfloor \frac{tv_{mm}+j'_m}{\widetilde{h}'_{mm}} \rfloor + off_m) - off_m - \frac{tv_{mm}}{\widetilde{h}'_{mm}}) - \lfloor \frac{\sum_{l=1}^{m-1} \widetilde{h}'_{ml} y_l}{\widetilde{h}'_{mm}} \rfloor \end{array}\right\} \Leftrightarrow y_i = \lfloor \frac{j'_i}{\widetilde{h}'_{ii}} \rfloor - \lfloor \frac{\sum_{l=1}^{i-1} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor$$

- *(α) Πρέπει να δείξουμε ότι ισχύει* $t = \lfloor \frac{((\lfloor \frac{tv_{mm}+j'_m}{\widetilde{h}'_{mm}} \rfloor + off_m) - off_m)\widetilde{h}'_{mm}}{v_{mm}} \rfloor \Leftrightarrow t = t + \lfloor \frac{\lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor \widetilde{h}'_{mm}}{v_{mm}} \rfloor$. *Από την ανισότητα* $0 \leq j'_m < v_{mm} \Rightarrow 0 \leq \lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor < \frac{v_{mm}}{\widetilde{h}'_{mm}} \Rightarrow 0 \leq \lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor \widetilde{h}'_{mm} < v_{mm} \Rightarrow 0 \leq \lfloor \frac{\lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor \widetilde{h}'_{mm}}{v_{mm}} \rfloor < 1 \Rightarrow \lfloor \frac{\lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor \widetilde{h}'_{mm}}{v_{mm}} \rfloor = 0$. *Έτσι,* $t = t + \lfloor \frac{\lfloor \frac{j'_m}{\widetilde{h}'_{mm}} \rfloor \widetilde{h}'_{mm}}{v_{mm}} \rfloor \Leftrightarrow t = t + 0$, *που ισχύει πάντα.*
- *(β) Από την έκφραση* $y_i = \lfloor \frac{j'_i}{\widetilde{h}'_{ii}} \rfloor - \lfloor \frac{\sum_{l=1}^{i-1} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor \Rightarrow \lfloor \frac{j'_i}{\widetilde{h}'_{ii}} \rfloor = y_i + \lfloor \frac{\sum_{l=1}^{i-1} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor = \lfloor \frac{\sum_{l=1}^{i} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor \Rightarrow \widetilde{h}'_{ii} \lfloor \frac{\sum_{l=1}^{i} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor \leq j'_i \leq \widetilde{h}'_{ii} \lfloor \frac{\sum_{l=1}^{i} \widetilde{h}'_{il} y_l}{\widetilde{h}'_{ii}} \rfloor + \widetilde{h}'_{ii} - 1$. *Στο διάστημα αυτό υπάρχει μόνο ένα ακριβές σημείο* $\vec{j'}$ *(αφού* $\widetilde{h}'_{ii}$ *είναι το βήμα του* j'_i *προκειμένου να προκύψει επίσης ακριβές σημείο), το οποίο είναι το* $\sum_{l=1}^{i} \widetilde{h}'_{il} y_l$. *Επομένως, ισχύει ότι* $j'_i = \sum_{l=1}^{i} \widetilde{h}'_{il} y_l$.

2.

$$\vec{j''} = map(map^{-1}(\vec{j''})) \Leftrightarrow \left\{\begin{array}{l} j''_k = \lfloor \frac{\sum_{l=1}^{k} \widetilde{h}'_{kl} z_l}{\widetilde{h}'_{kk}} \rfloor + off_k, k \neq m \\ j''_m = \lfloor \frac{tv_{mm} + \sum_{l=1}^{m} \widetilde{h}'_{ml} z_l}{\widetilde{h}'_{mm}} \rfloor + off_m \end{array}\right\} \tag{4.2}$$

όπου:

$$\left.\begin{array}{c} z_l = j''_l - off_l - \lfloor \frac{\sum_{i=1}^{l-1} \widetilde{h}'_{li} z_i}{\widetilde{h}'_{ll}} \rfloor, l \neq m \\ z_m = j''_m - off_m - \frac{tv_{mm}}{\widetilde{h}'_{mm}} - \lfloor \frac{\sum_{i=1}^{m-1} \widetilde{h}'_{mi} z_i}{\widetilde{h}'_{mm}} \rfloor \end{array}\right\} \Rightarrow \left\{\begin{array}{c} j''_l = off_l + \lfloor \frac{\sum_{i=1}^{l} \widetilde{h}'_{li} z_i}{\widetilde{h}'_{ll}} \rfloor, l \neq m \\ j''_m = off_m + \frac{tv_{mm}}{\widetilde{h}'_{mm}} + \lfloor \frac{\sum_{i=1}^{m} \widetilde{h}'_{mi} z_i}{\widetilde{h}'_{mm}} \rfloor \end{array}\right. \tag{4.3}$$

Με τη βοήθεια της (4.3), η (4.2) μετασχηματίζεται ισοδύναμα ως εξής:

$$\begin{cases} off_k + \lfloor \frac{\sum_{i=1}^{k} \widetilde{h}'_{ki} z_i}{\widetilde{h}'_{kk}} \rfloor = \lfloor \frac{\sum_{l=1}^{k} \widetilde{h}'_{kl} z_l}{\widetilde{h}'_{kk}} \rfloor + off_k, k \neq m \\ off_m + \frac{tv_{mm}}{\widetilde{h}'_{mm}} + \lfloor \frac{\sum_{i=1}^{m} \widetilde{h}'_{mi} z_i}{\widetilde{h}'_{mm}} \rfloor = \lfloor \frac{tv_{mm} + \sum_{l=1}^{m} \widetilde{h}'_{ml} z_l}{\widetilde{h}'_{mm}} \rfloor + off_m \end{cases}$$

το οποίο προφανώς ισχύει πάντα, αν λάβουμε υπ' όψιν ότι το v_{mm} είναι πολλαπλάσιο του $\widetilde{h}'_{mm}$.

Μετά την απόδειξη των (1) και (2), προκύπτει ότι το Λήμμα 4.2 ισχύει πάντα. □

Η συνάρτηση $map(\vec{j'}, t)$ του Λήμματος 4.1, καθορίζει τη θέση μνήμης στον Τοπικό Χώρο Δεδομένων, όπου θα αποθηκευτούν τα δεδομένα που υπολογίζονται στην επανάληψη $\vec{j'} \in TTIS$ (Σχήμα 4.3). Προφανώς, η συνάρτηση $map^{-1}(\vec{j''})$ του Λήμματος 4.2 υπολογίζει το σημείο $\vec{j'} \in TTIS$ κατά το οποίο υπολογίστηκε το $\vec{j''} \in LDS$. Αξίζει να σημειωθεί, ότι όλες οι διαιρέσεις στις παραπάνω εκφράσεις είναι ακέραιες διαιρέσεις. Η συνάρτηση $loc(\vec{j})$ στον Πίνακα 4.1 χρησιμοποιεί την $map(\vec{j'}, t)$ για να εντοπίσει τον επεξεργαστή $\vec{pid}$ και τη θέση μνήμης $\vec{j''} \in LDS$ όπου αποθηκεύονται τα δεδομένα που θα εκτελούντο κατά την επανάληψη $\vec{j} \in J$ του αρχικού χώρου. Αντιστρόφως, ο Πίνακας 4.2 δείχνει τη σειρά των βημάτων που απαιτούνται, προκειμένου να εντοπίσουμε σε ποιό σημείο του αρχικού χώρου $\vec{j} \in J$, αντιστοιχεί το σημείο $\vec{j''} \in LDS$ που βρίσκεται στην τοπική μνήμη του επεξεργαστή $\vec{pid}$. Επομένως, η συνάρτηση $loc^{-1}()$ καλείται από τους επεξεργαστές στο τέλος των υπολογισμών τους για να μεταβούν από τον Τοπικό Χώρο Δεδομένων τους στον αρχικό χώρο επαναλήψεων J. Στη συνέχεια το αντίστοιχο σημείο του αρχικού Χώρου Δεδομένων DS προκύπτει με τη βοήθεια της συνάρτησης f_w (Σχήμα 4.4).

Με βάση το σχήμα κατανομής δεδομένων που παρουσιάστηκε, κάθε επεξεργαστής δεσμεύει ακριβώς το μέγεθος της μνήμης που θα χρησιμοποιήσει για την αποθήκευση των υπολογισμένων δεδομένων και των δεδομένων επικοινωνίας. Αξίζει να τονιστεί, ότι η απευθείας δέσμευση μνήμης στον αρχικό Χώρο Δεδομένων DS θα οδηγούσε σε σπατάλη χώρου μνήμης, καθώς για να συμβεί αυτό, θα έπρεπε να δεσμευτεί ολόκληρη η ορθογώνια περιοχή που περικλείει ένα γενικά μη-ορθογώνιο χώρο δεδομένων (Σχήμα 4.5). Ακόμα και αν χρησιμοποιηθεί ορθογώνιος μετασχηματισμός υπερκόμβων, στη γενική περίπτωση λόγω της f_w, ο χώρος δεδομένων που χρησιμοποιεί ένας υπερκόμβος δεν είναι ορθογώνιος [AKN95]. Με τη μέθοδο όμως που παρουσιάστηκε, υποχρεώνουμε κάθε επεξεργαστή να δεσμεύει ορθογώνιο χώρο μνήμης, ανε-

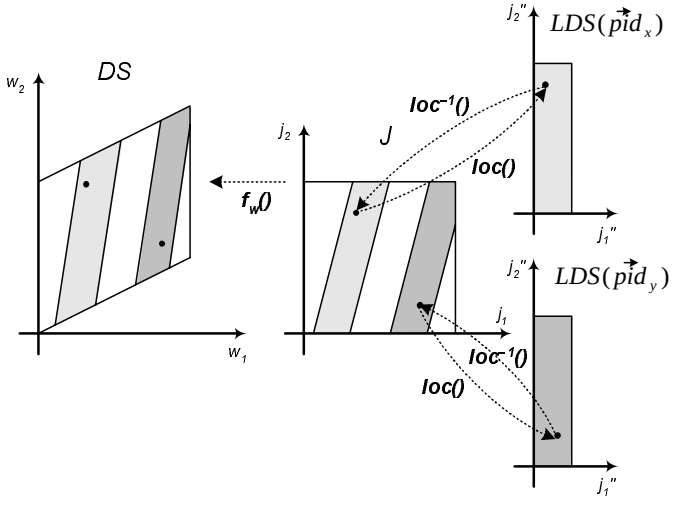
$\tilde{\mathbf{j}}'' = \mathbf{map}(\tilde{\mathbf{j}}', \mathbf{t})$
$j''_k = j'_k / \widetilde{h'}_{kk} + off_k, k \neq m$
$j''_m = (tv_{mm} + j'_m) / \widetilde{h'}_{mm} + off_m$
$(\tilde{\mathbf{j}}'', \tilde{\mathbf{pid}}) = \mathbf{loc}(\tilde{\mathbf{j}})$
$\vec{j^S} = \lfloor H\vec{j} \rfloor$
$\vec{j'} = H'(\vec{j} - P\vec{j^S})$
$\vec{j''} = map(\vec{j'}, j^S_m - l^S_m)$
$\vec{pid} = (j^S_1, \ldots, j^S_{m-1}, j^S_{m+1}, \ldots, j^S_n)$

Πίνακας 4.1: Εύρεση της εικόνας του σημείου $\vec{j} \in J$ στον Τοπικό Χώρο Δεδομένων (LDS) ενός επεξεργαστή με τη βοήθεια της συνάρτησης $loc()$

$\tilde{\mathbf{j}}' = \mathbf{map^{-1}}(\tilde{\mathbf{j}}'')$
$t = (j''_m - off_m) \widetilde{h'}_{mm} / v_{mm}$
$x_k = j''_k - off_k - (\sum_{l=1}^{k-1} x_l \widetilde{h'}_{kl}) / \widetilde{h'}_{kk}, k \neq m$
$x_m = j''_m - off_m - tv_{mm} / \widetilde{h'}_{mm} - (\sum_{l=1}^{m-1} x_l \widetilde{h'}_{ml}) / \widetilde{h'}_{mm}$
$\vec{j'} = \widetilde{H'}\vec{x}$
$\tilde{\mathbf{j}} = \mathbf{loc^{-1}}(\tilde{\mathbf{j}}'', \tilde{\mathbf{pid}})$
$\vec{j'} = map^{-1}(\vec{j''})$
$\vec{j^S} = (pid_1, \ldots, pid_{m-1}, t + l^S_m, pid_{m+1}, \ldots, pid_n)$
$\vec{j} = P\vec{j^S} + P'\vec{j'}$

Πίνακας 4.2: Εύρεση της εικόνας του σημείου $\vec{j''} \in LDS$ (του επεξεργαστή $\vec{pid}$) στον αρχικό χώρο J με τη βοήθεια της συνάρτησης $loc^{-1}()$

ξάρτητα από το σχήμα του μετασχηματισμού υπερκόμβων που έχει χρησιμοποιηθεί. Τέλος, δεδομένου ότι οι χώροι δεδομένων των αλγορίθμων που μας απασχολούν είναι γενικά πολύ μεγάλοι, η συμπίεση του τοπικού χώρου δεδομένων ώστε να προκύψει ο LDS κρίνεται γενικά αναγκαία. Τελικά, αυτό οδηγεί σε ένα συμβιβασμό ανάμεσα στο χώρο δεδομένων και την υπολογιστική πολυπλοκότητα, καθώς πρέπει να χρησιμοποιηθούν οι σχετικές εκφράσεις για να μεταβούμε στον LDS. Το κόστος αυτών των εκφράσεων όμως θεωρείται αμελητέο και δεν επηρεάζει σημαντικά το χρόνο εκτέλεσης του παραλληλοποιημένου αλγορίθμου.



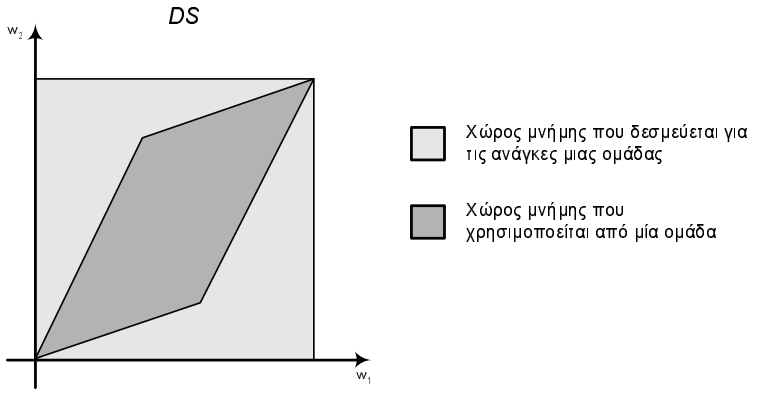
Σχήμα 4.4: Μεταβάσεις μεταξύ του Χώρου Δεδομένων (DS), του Χώρου Επαναλήψεων (J) και των Τοπικών Χώρων Δεδομένων (LDS)

4.4 Παραγωγή Πρωτογενών Κλήσεων Επικοινωνίας

Με την ανάθεση των υπολογισμών σε επεξεργαστές και την κατανομή των δεδομένων που παρουσιάστηκε στις προηγούμενες ενότητες, είναι δυνατόν κάποια δεδομένα που έχουν εγγραφεί στην τοπική μνήμη ενός επεξεργαστή να χρειάζονται για τους υπολογισμούς σε κάποιον γειτονικό επεξεργαστή. Πιο συγκεκριμένα, με βάση το μοντέλο των εξαρτήσεων που ακολουθούν οι αλγόριθμοι που μας απασχολούν, τα δεδομένα που διαβάζει ένας επεξεργαστής μπορεί να υπολογίζονται από κάποιον άλλο. Σ' αυτές τις περιπτώσεις, οι επεξεργαστές πρέπει να επικοινωνήσουν ανταλλάσσοντας μηνύματα. Τα δύο βασικά ζητήματα που πρέπει να αντιμετωπιστούν κατά την αυτόματη παραγωγή των πρωτογενών κλήσεων ανταλλαγής μηνυμάτων, είναι ο καθορισμός των δεδομένων που θα αποσταλούν σε κάθε μήνυμα (σύνολα επικοινωνίας) και των επεξεργαστών που θα επικοινωνήσουν.

4.4.1 Σύνολα Επικοινωνίας

Τα σημεία επικοινωνίας από τα οποία προκύπτουν τα αντίστοιχα δεδομένα που πρέπει να αποσταλούν προκύπτουν από τον ακόλουθο ορισμό.



Σχήμα 4.5: Σπατάλη σε χώρο μνήμης για τη δέσμευση μνήμης ενός μη-ορθογώνιου υπερκόμβου

Ορισμός 4.2 *Έστω m η διάσταση απεικόνισης και η $\vec{d^S} \in D^S$ είναι μία εξάρτηση υπερκόμβων που προκαλεί και εξάρτηση (σύνδεση) μεταξύ επεξεργαστών, δηλαδή ισχύει $\exists\ l \neq m$: $d_l^S \neq 0$. Ένα σημείο $\vec{j'} \in TTIS$ είναι σημείο επικοινωνίας κατά τη διεύθυνση του $\vec{d^S}$, αν και μόνο αν τα δεδομένα που υπολογίστηκαν στην επανάληψη $\vec{j} = P\vec{j^S} + P'\vec{j'}$ είναι αναγκαία για τον υπολογισμό κάποιου σημείου που ανήκει στον υπερκόμβο $\vec{j^S} + \vec{d^S}$, όπου $\vec{j^S} \in J^S$ και $\vec{j^S} + \vec{d^S} \in J^S$.*

Αξίζει να σημειωθεί, πως εφ' όσον το διάνυσμα $\vec{d^S}$ υποδηλώνει εξάρτηση μεταξύ επεξεργαστών, οι υπερκόμβοι $\vec{j^S}$ και $\vec{j^S} + \vec{d^S}$ θα υπολογίζονται σε διαφορετικούς επεξεργαστές. Επίσης, κάθε σημείο επικοινωνίας ορίζεται σε σχέση με κάποιο διάνυσμα εξάρτησης μεταξύ υπερκόμβων $\vec{d^S}$. Με άλλα λόγια, τα σημεία επικοινωνίας στον $TTIS$ αντιστοιχούν σε επαναλήψεις στις οποίες υπολογίζονται δεδομένα που πρέπει να αποσταλούν σε κάποιον άλλο επεξεργαστή στη διεύθυνση του $\vec{d^S}$. Και σ' αυτήν την περίπτωση εκμεταλλευόμαστε την κανονικότητα του $TTIS$ για να εξάγουμε απλά κριτήρια υπολογισμού των σημείων επικοινωνίας, όπως προκύπτει από το λήμμα που ακολουθεί.

Λήμμα 4.3 *Ένα σημείο $\vec{j'} = (j'_1, \ldots, j'_n) \in TTIS$ αντιστοιχεί σε ένα σημείο επικοινωνίας*

στη διεύθυνση της εξάρτησης υπερκόμβων $\vec{d^S} = (d_1^S, \ldots, d_n^S) \in D^S$ *αν και μόνο αν ισχύει:*

$$j'_k \geq d_k^S(v_{kk} - \max_{\vec{d'} \in D'}\{d'_k\})$$

όπου $k = 1, \ldots, n, \vec{d'} \in D', D' = H'D$.

Απόδειξη 4.3 *Για να είναι το* $\vec{j'}$ *σημείο επικοινωνίας κατά την* k*-στή διάσταση, διακρίνουμε δύο περιπτώσεις:*

1. $d_k^S = 0$. *Καθώς δεν υπάρχει εξάρτηση υπερκόμβων σε αυτή τη διεύθυνση, δεν ορίζεται περιορισμός για το* j'_k. *Έτσι θα ισχύει* $0 \leq j'_k \leq v_{kk} - 1$.

2. $d_k^S = 1$. *Σ' αυτή την περίπτωση, θα πρέπει να υπάρχει μία εξάρτηση στον* $TTIS$ $\vec{d'} \in D'$ *τέτοια ώστε το* k*-στό στοιχείο του* $\vec{j'} + \vec{d'}$ *να ξεπερνά το αντίστοιχο όριο του* $TTIS$, *προκαλώντας έτσι ανάγκη για επικοινωνία στην* k*-στή διάσταση.*

   Σύμφωνα με τα παραπάνω θα πρέπει να ισχύει

$$j'_k + d'_k > v_{kk} - 1 \Rightarrow j'_k + d'_k \geq v_{kk} \Rightarrow j'_k \geq v_{kk} - d'_k$$

   για κάποιο $\vec{d'} \in D'$ *ή ισοδύναμα*

$$j'_k \geq v_{kk} - \max_{d' \in D'}\{d'_k\}$$

Η ένωση των δύο παραπάνω προτάσεων δίνει το ζητούμενο αποτέλεσμα. □

Ορισμός 4.3 *Το διάνυσμα επικοινωνίας* $\vec{CC}$ *ορίζεται ως* $\vec{CC} = (cc_1, \ldots, cc_n)$ *όπου*

$$cc_k = v_{kk} - \max_{d' \in D'} \{d'_k\}$$

και $k = 1, \ldots, n$, $d' \in D'$, $D' = H'D$.

Είναι προφανές ότι το διάνυσμα $\vec{CC}$ μπορεί εύκολα να υπολογιστεί κατά τη διάρκεια της μεταγλώττισης. Σύμφωνα με το Λήμμα 4.3, το διάνυσμα $\vec{CC}$ μπορεί να χρησιμοποιηθεί για να καθορίσει τα σημεία επικοινωνίας:

Πόρισμα 4.1 *Έστω το διάνυσμα επικοινωνίας* $\vec{CC} = (cc_1, \ldots, cc_n)$. *Ένα σημείο* $\vec{j'} = (j'_1, \ldots, j'_n) \in TTIS$ *αντιστοιχεί σε σημείο επικοινωνίας στη διεύθυνση της εξάρτησης* $\vec{d^S} = (d^S_1, \ldots, d^S_n) \in D^S$ *αν και μόνο αν ισχύει:*

$$j'_k \geq d^S_k cc_k$$

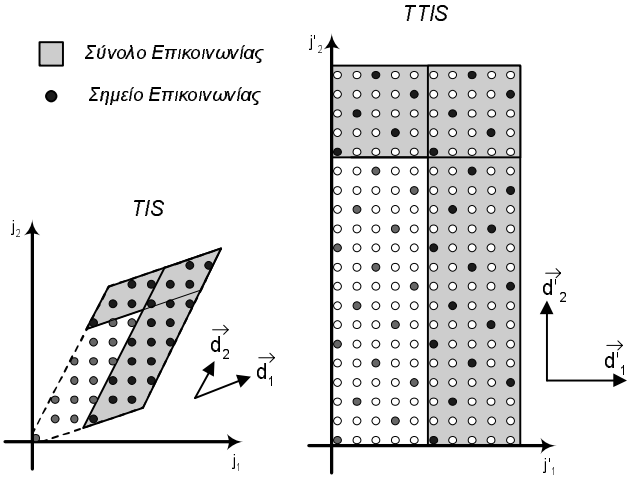
για $k = 1, \ldots, n$.

Θα εφαρμόσουμε το Πόρισμα 4.1 στην παραγωγή του παράλληλου κώδικα για τον καθορισμό των συνόλων επικοινωνίας. Συγκεκριμένα, τα δεδομένα που προκύπτουν από το Πόρισμα 4.1, πρέπει να συγκεντρωθούν και να "πακεταριστούν" στον αποστολέα, και ανάλογα να "ξεπακεταριστούν" στον παραλήπτη. Είναι μάλλον εμφανές, ότι ο καθορισμός των συνόλων επικοινωνίας στον $TTIS$ είναι πολύ πιο απλός από άλλες εναλλακτικές λύσεις, όπως για παράδειγμα στον TIS. Στο Σχήμα 4.6 φαίνονται τα σύνολα επικοινωνίας όπως υπολογίζονται στο TIS και στον $TTIS$. Στη δεύτερη περίπτωση, τα σύνολα επικοινωνίας είναι πολύ πιο εύκολο να καθοριστούν, καθώς σε κάθε περίπτωση είναι ορθογώνιοι χώροι. Αξίζει επίσης να σημειωθεί, ότι οι παράγοντες off_k που χρησιμοποιήθηκαν στην Ενότητα 4.3 για τον ορισμό του Τοπικού Χώρου

Δεδομένων, δίνονται από τις σχέσεις:

$$off_k = \lceil \max_{d' \in D'} \{d'_k\}/c_k \rceil, k \neq m \tag{4.4}$$

$$off_m = v_{mm}/c_m \tag{4.5}$$

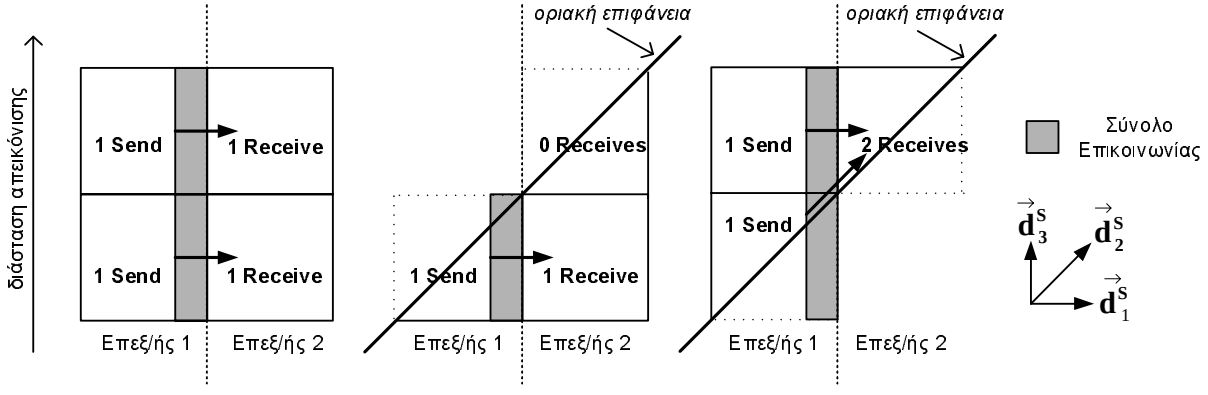


Σχήμα 4.6: Προσδιορισμός των συνόλων επικοινωνίας στο Χώρο Επαναλήψεων Υπερκόμβου (TIS) και στο Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου ($TTIS$)

4.4.2 Επικοινωνία Μεταξύ Επεξεργαστών

Θα αναφερθούμε στο σχήμα επικοινωνίας που λαμβάνει χώρα, όταν η δρομολόγηση των υπερκόμβων δεν περιλαμβάνει επικάλυψη υπολογισμού και επικοινωνίας. Με μικρές τροποποιήσεις οι τεχνικές που θα παρουσιαστούν, εφαρμόζονται και στην περίπτωση επικάλυψης υπολογισμών και επικοινωνίας. Η επικοινωνία σ' αυτή την περίπτωση λαμβάνει χώρα πριν και μετά την εκτέλεση των υπολογισμών στα εσωτερικά σημεία ενός υπερκόμβου. Πριν την εκτέλεση των υπολογισμών, ο επεξεργαστής πρέπει να λάβει τα δεδομένα που χρειάζεται για την διεξαγωγή των υπολογισμών και τα οποία έχουν υπολογιστεί σε κάποιον απομακρυσμένο επεξεργαστή. Στη συνέχεια "ξεπακετάρει" τα δεδομένα αυτά στις κατάλληλες θέσεις του Τοπικού Χώρου Δεδομένων. Δυαδικά, μετά την ολοκλήρωση των υπολογισμών στο εσωτερικό ενός υπερκόμβου, ο επεξεργαστής πρέπει να στείλει τα δεδομένα που έχει υπολογίσει και που είναι αναγκαία

για τον υπολογισμό σε γειτονικούς επεξεργαστές. Έστω D^m η προβολή του D^S στις $n-1$ διαστάσεις-αφαιρώντας τη διάσταση m. Ο πίνακας D^m εκφράζει εξαρτήσεις επεξεργαστών, κάτι που σημαίνει ότι, γενικά, ο επεξεργαστής $\vec{pid}$ πρέπει να λάβει δεδομένα από τους επεξεργαστές $\vec{pid} - \vec{d^m}$ και να στείλει δεδομένα στους επεξεργαστές $\vec{pid} + \vec{d^m}$ για όλα τα $\vec{d^m} \in D^m$.

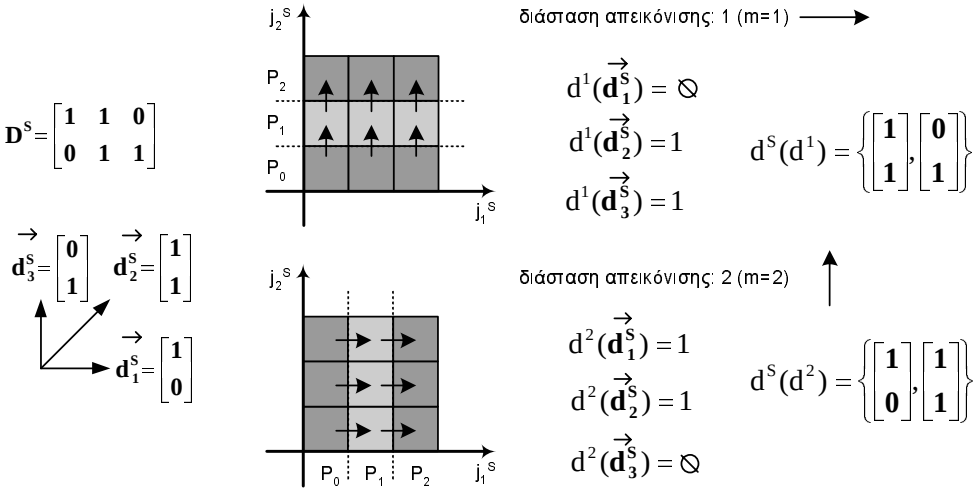


Σχήμα 4.7: Ασυμμετρία στην επικοινωνία μεταξύ δύο επεξεργαστών

Υιοθετούμε το σχήμα επικοινωνίας των Tang και Xue [TX00], το οποίο υπαγορεύει μία απλή υλοποίηση της διαδικασίας αποστολής και μία περισσότερο πολύπλοκη διαδικασία για τη λήψη των δεδομένων. Η ασυμμετρία ανάμεσα στις δύο αυτές διαδικασίες (αποστολή-λήψη) προκύπτει από το γεγονός, ότι ένας υπερκόμβος μπορεί να λάβει από περισσότερους του ενός υπερκόμβου, του ίδιου απομακρυσμένου επεξεργαστή, αλλά θα στείλει τα δεδομένα του μόνο μία φορά σε κάθε γειτονικό επεξεργαστή. Με αυτό τον τρόπο ικανοποιούνται όλες οι εξαρτήσεις μεταξύ υπερκόμβων που καταλήγουν στον ίδιο επεξεργαστή, με το ίδιο μήνυμα. Με άλλα λόγια, ένας υπερκόμβος *λαμβάνει από υπερκόμβους, αλλά στέλνει σε επεξεργαστές*. Στο Σχήμα 4.7 φαίνεται η ασυμμετρία αυτή. Στην πρώτη περίπτωση αριστερά του σχήματος, η επικοινωνία είναι απλή, καθώς κάθε επεξεργαστής κάνει μία αποστολή και μία λήψη για κάθε υπερκόμβο. Όταν όμως βρισκόμαστε σε οριακές επιφάνειες, τα πράγματα είναι κάπως διαφορετικά. Στο μέσον του σχήματος, ο Επεξεργαστής 1 πραγματοποιεί μία αποστολή, αν και υπάρχουν δύο υπερκόμβοι στον Επεξεργαστή 2 που χρειάζονται τα συγκεκριμένα δεδομένα. Ο λεξικογραφικά μικρότερος υπερκόμβος στον Επεξεργαστή 2 κάνει την απαραίτητη λήψη δεδομένων. Στην τρίτη περίπτωση φαίνεται πιο ξεκάθαρα η ασυμμετρία αποστολής-λήψης ανά υπερκόμβο. Και εδώ έχουμε το γνωστό σχήμα, όπου κάθε υπερκόμβος κάνει ακριβώς μία αποστολή (Επεξεργαστής 1). Στον Επεξεργαστή 2 όμως, ένας υπερκόμβος πραγματοποιεί δύο λήψεις για να εξασφαλίσει την ορθή λειτουργία του σχήματος αποστολής-λήψης.

4.5 Τελικός SPMD Κώδικας

Ο τελικός SPMD κώδικας προκύπτει με την εφαρμογή των τεχνικών που παρουσιάστηκαν στο κεφάλαιο αυτό. Στον κώδικα που θα παρουσιάσουμε, κάνουμε χρήση των εκφράσεων $d^m(\vec{d^S})$ και $d^S(\vec{d^m})$. Το $d^m(\vec{d^S})$ υποδηλώνει την εξάρτηση ανάμεσα σε επεξεργαστές που αντιστοιχεί στην εξάρτηση υπερκόμβων $\vec{d^S}$, ενώ το $d^S(\vec{d^m})$ υποδηλώνει όλες τις εξαρτήσεις υπερκόμβων που γεννούν την εξάρτηση επεξεργαστών $\vec{d^m}$. Στο Σχήμα 4.8 παρουσιάζουμε ένα παράδειγμα υπολογισμού των παραπάνω εκφράσεων σε ένα δισδιάστατο πρόβλημα με τρία διανύσματα εξάρτησης μεταξύ των υπερκόμβων, $\vec{d_1^S} = (1,0)^T$, $\vec{d_2^S} = (1,1)^T$ και $\vec{d_3^S} = (0,1)^T$. Στην πρώτη περίπτωση θεωρούμε σαν διάσταση απεικόνισης την πρώτη διάσταση $(m = 1)$ και επομένως, όπως φαίνεται και στο πάνω μέρος του σχήματος, η εξάρτηση μεταξύ των επεξεργαστών είναι κατά τη διεύθυνση του j_2^S. Ουσιαστικά, η έκφραση $d^1(\vec{d^S})$ φανερώνει αν το διάνυσμα $\vec{d^S}$, προκαλεί επικοινωνία μεταξύ επεξεργαστών. Προφανώς, για $m = 1$, τα διανύσματα $\vec{d_2^S}$ και $\vec{d_3^S}$ προκαλούν επικοινωνία και γι' αυτό το λόγο οι αντίστοιχες εκφράσεις είναι ίσες με τη μονάδα. Αντίθετα, το διάνυσμα εξάρτησης $\vec{d_1^S}$ παραμένει μέσα στον επεξεργαστή και έτσι $d^1(\vec{d_1^S}) = \emptyset$. Η έκφραση $d^S(\vec{d^1})$ υπολογίζει από ποια διανύσματα εξάρτησης υπερκόμβων προέκυψε το διάνυσμα εξάρτησης μεταξύ των επεξεργαστών, άρα $d^S(\vec{d^1}) = \{(1,1)^T, (0,1)^T\}$. Ανάλογα υπολογίζονται οι εκφράσεις για $m = 2$.



Σχήμα 4.8: Παράδειγμα προσδιορισμού των εκφράσεων $d^m(\vec{d^S})$ και $d^S(\vec{d^m})$

Υλοποιήσαμε τα σχήματα για την λήψη και την αποστολή δεδομένων χρησιμοποιώντας τη βιβλιοθήκη ανταλλαγής μηνυμάτων MPI (Message Passing Interface) [MPI94] [MPI96]. Επει-

δή το MPI δίνει στις διεργασίες του αναγνωριστικά μίας διάστασης (`MPI_rank`), ενώ η δική μας μέθοδος απεικόνισης θεωρεί $n-1$ διαστάσεων αναγνωριστικά των διεργασιών, χρησιμοποιούμε τη συνάρτηση `MPI_rank=Rank(`$\vec{pid}$`)` για να απεικονίσουμε τα αναγνωριστικά $\vec{pid}$ σε αναγνωριστικά του MPI. Η συνάρτηση $\vec{pid}$`=Rank`$^{-1}$`(MPI_rank)` προφανώς υλοποιεί την αντίστροφη διαδικασία. Η συνάρτηση που υλοποιεί τη διαδικασία λήψης δεδομένων είναι ως εξής:

```
 1:  RECEIVE(pid, t^S, D^S, CC)
 2:     FOR d^S ∈ D^S  /* για όλες τις εξαρτήσεις υπερκόμβων */
 3:        /*αν ο προηγούμενος υπερκόμβος είναι έγκυρος
 4:          και ο τρέχων υπερκόμβος είναι ο λεξικογραφικά μικρότερος επόμενός του */
 5:        IF(valid((pid, t^S) − d^S) ∧ (pid, t^S)=minsucc((pid, t^S) − d^S, d^m(d^S)))
 6:           /* λήψη δεδομένων από τον προηγούμενο επεξεργαστή */
 7:           MPI_Recv(buffer,Rank(pid − d^m(d^S)),...);
 8:           /* τοποθέτηση των δεδομένων που έχουν ληφθεί στον LDS. */
 9:           count=0;
10:           FOR (j'_1=max(l'_1, d^S_1 cc_1); j'_1<u'_1; j'_1+=c_1)
11:              ...
12:              FOR (j'_m=l'_m; j'_m<u'_m; j'_m+=c_m)
13:                 ...
14:                 FOR (j'_n=max(l'_n, d^S_n cc_n); j'_n<u'_n; j'_n+=c_n)
15:                    LA[map(j', t^S − l^S_n)-(d^S_1 v_11/c_1, ..., d^S_n v_nn/c_n)]=buffer[count++];
16:                 ENDFOR
17:                 ...
18:              ENDFOR
19:              ...
20:           ENDFOR
21:        ENDIF
22:     ENDFOR
```

Ο κώδικας της συνάρτησης `RECEIVE()` ελέγχει όλες τις εξαρτήσεις μεταξύ υπερκόμβων (γραμμή 2) και αν υπάρχει κάποιος έγκυρος υπερκόμβος σε άλλον επεξεργαστή από τον οποίο πρέπει να λάβει δεδομένα, τότε πραγματοποιεί λήψη δεδομένων (γραμμή 7). Η συνάρτηση `minsucc(`$\vec{s}, \vec{d^m}$`)` υπολογίζει το λεξικογραφικά μικρότερο προηγούμενο υπερκόμβο του υπερκόμβου $\vec{s}$ στη διεύθυνση $\vec{d^m}$, ενώ η συνάρτηση `valid(`$\vec{s}$`)` επιστρέφει "αληθές" αν ο υπερκόμβος $\vec{s}$ ανήκει στον αρχικό χώρο (γραμμή 5). Οι γραμμές 10-20 διασχίζουν τον τοπικό χώρο δεδομένων και τοποθετούν τα δεδομένα που έχουν ληφθεί στις κατάλληλες θέσεις. Ο LA είναι ένας τοπικός πίνακας δεδομένων που υλοποιεί τον Τοπικό Χώρο Δεδομένων LDS. Η αποστολή δεδομένων υλοποιείται με τη συνάρτηση:

```
 1:  SEND(p⃗id, t^S, D^m, C⃗C)
 2:     FOR d⃗^m ∈ D^m /* Για όλες τις εξαρτήσεις μεταξύ επεξεργαστών */
 3:        /* αν υπάρχει έγκυρος λεξικογραφικά επόμενος υπερκόμβος */
 4:        IF(∃ d^S(d⃗^m) ∈ D^S :valid((p⃗id, t^S) + d^S(d⃗^m)))
 5:           count=0;
 6:           /*"πακετάρισε" τα δεδομένα επικοινωνίας */
 7:           FOR (j'_1=max(l'_1,d^m_1 cc_1); j'_1<u'_1; j'_1+=c_1)
 8:              ...
 9:              FOR (j'_m=l'_m; j'_m<u'_m; j'_m+=c_m)
10:                 ...
11:                 FOR (j'_n=max(l'_n,d^m_n cc_n); j'_n<u'_n; j'_n+=c_n)
12:                    buffer[count++]=LA[map(j⃗',t^S − l^S_n)];
13:                 ENDFOR
14:                 ...
15:              ENDFOR
16:              ...
17:           ENDFOR
18:           /* αποστολή των δεδομένων στον επόμενο επεξεργαστή. */
19:           MPI_Send(buffer,Rank(p⃗id + d⃗^m),...);
20:        ENDIF
21:     ENDFOR
```

Ο κώδικας της συνάρτησης SEND() ελέγχει για κάθε διάνυσμα εξάρτησης μεταξύ επεξεργαστών (γραμμή 2), αν υπάρχει έγκυρος λεξικογραφικά επόμενος υπερκόμβος (γραμμή 4) και αν συμβαίνει αυτό, διατρέχει το σύνολο επικοινωνίας (γραμμές 7-17), πακετάρει τα δεδομένα και τα στέλνει στον κατάλληλο επεξεργαστή (γραμμή 19). Συνοψίζοντας, ο τελικός παράλληλος SPMD κώδικας θα έχει τη μορφή:

```
DOACROSS (pid_1=l_1^S; pid_1 ≤ u_1^S; pid_1++);
  ...
  DOACROSS (pid_{n-1}=l_{n-1}^S; pid_{n-1} ≤ u_{n-1}^S; pid_{n-1}++)
    /*Σειριακή Εκτέλεση Υπερκόμβων*/
    FOR (t^S=l_n^S; t^S ≤ u_n^S; t^S++)
      /*Λήψη δεδομένων από γειτονικούς υπερκόμβους*/
      RECEIVE(p⃗id, t^S, D^S, C⃗C);
      /*Διάσχιση του εσωτερικού του υπερκόμβου*/
      FOR (j'_1=l'_1; j'_1 ≤ u'_1; j'_1+=c_1, j'_2+=a_21, ..., j'_n+=a_n1)
        ...
        FOR (j'_n=l'_n; j'_n ≤ u'_n; j'_n+=c_n)
          /*Διεξαγωγή υπολογισμών στον τοπικό χώρο δεδομένων LDS*/
          t=t^S-l_n^S;
          LA[map(j⃗',t)]=F(LA[map(j⃗'-d⃗'_1,t)], ..., LA[map(j⃗'-d⃗'_q,t)]);
        ENDFOR
        ...
      ENDFOR
      /*Αποστολή δεδομένων σε γειτονικούς επεξεργαστές*/
      SEND(p⃗id, t^S, D^m, C⃗C);
    ENDFOR
    /*Εγγραφή αποτελεσμάτων στον αρχικό Χώρο Δεδομένων DS*/
    FOR (j⃗'' ∈ LDS) WRITE(A[f_w(loc^{-1}(j⃗'', p⃗id))], LA[j⃗'']);
  ENDDOACROSS
  ...
ENDDOACROSS
```

Παράδειγμα 4.1 Θα εφαρμόσουμε τη διαδικασία παραλληλοποίησης, που παρουσιάστηκε σ' αυτό το κεφάλαιο, στον αλγόριθμο των Παραδειγμάτων 2.3 και 3.1-3.3. Αρχικά, πρέπει να επιλεγεί η διάσταση απεικόνισης. Όπως φαίνεται και από τα Σχήματα 2.9 και 2.10, η διάσταση με το μεγαλύτερο μήκος είναι η j_1^S. Με βάση τα όσα ειπώθηκαν στην Ενότητα 4.2, θα ήταν πιο αποδοτικό να απεικονίσουμε στον ίδιο επεξεργαστή υπερκόμβους κατά μήκος της πρώτης διάστασης. Το γεγονός αυτό όμως θα απαιτούσε μετάθεση των βρόχων πριν από το στάδιο της παραγωγής του σειριακού μετασχηματισμένου κώδικα. Για να διατηρήσουμε μία ακολουθία με τα προηγούμενα παραδείγματα και για καθαρά εποπτικούς λόγους, θα απεικονίσουμε τους

υπερκόμβους κατά μήκος της διάστασης j_2^S στον ίδιο επεξεργαστή. Έτσι, $m = 2$.

$\texttt{map}(\vec{j'},t)$	$\texttt{map}^{-1}(\vec{j''})$	$\texttt{loc}^{-1}(\vec{j''},pid)$
$j_1''=j_1'+5;$ $j_2''=(20\text{t}+j_2')/5+4;$	$t=j_2''/5;$ $x_1=j_1''\text{-}5;$ $x_2=j_2''\text{-}4\text{-}4t\text{-}\frac{2}{5}x_1;$ $\vec{j'}=\begin{bmatrix}1 & 0\\ 2 & 5\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix};$	$\vec{j'}=\text{map}^{-1}(\vec{j''});$ $j_1^S=pid;$ $j_2^S=t+\max(\lceil\frac{-9.5-6pid}{4}\rceil,\lceil\frac{-9.5-2pid}{8}\rceil);$ $\vec{j}=\begin{bmatrix}6 & 4\\ 2 & 8\end{bmatrix}\begin{pmatrix}j_1^S\\ j_2^S\end{pmatrix}+\begin{bmatrix}\frac{3}{5} & \frac{1}{5}\\ \frac{1}{5} & \frac{2}{5}\end{bmatrix}\begin{pmatrix}j_1'\\ j_2'\end{pmatrix};$

Πίνακας 4.3: Κώδικας των συναρτήσεων $\texttt{map}()$, $\texttt{map}^{-1}()$ και $\texttt{loc}^{-1}()$

Ο πίνακας εξαρτήσεων στο Μετασχηματισμένο Χώρο Επαναλήψεων Υπερκόμβου θα είναι: $D' = H'D \Rightarrow D' = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 5 \\ 5 & 0 \end{bmatrix}$. Επομένως, με βάση τις Σχέσεις 4.4-4.5, $off_1 = \frac{5}{1} = 5$ και $off_2 = \frac{20}{5} = 4$. Έτσι, από τον Ορισμό 4.1 του Τοπικού Χώρου Δεδομένων, θα έχουμε: $LDS = \{\vec{j''} \in Z^n \mid 0 \leq j_1'' < 15 \mid 0 \leq j_2'' < 4 + 4|t|\}$. Οι συναρτήσεις $\texttt{map}()$, $\texttt{map}^{-1}()$ και $\texttt{loc}^{-1}()$ δίνονται στον Πίνακα 4.1. Το διάνυσμα επικοινωνίας $\vec{CC}$, με βάση τον Ορισμό 4.3, θα είναι $\vec{CC} = (5, 15)$. Τέλος, ο Πίνακας Εξαρτήσεων Υπερκόμβων D^S είναι $D^S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Ο τελικός SPMD κώδικας για το συγκεκριμένο παράδειγμα προκύπτει από τη συνένωση του κώδικα των Παραδειγμάτων 3.1 και 3.3, και με την προσθήκη των κατάλληλων κλήσεων που μετασχηματίζουν το κώδικα αυτό σε παράλληλο. Τα j_1^S και j_2^S αντικαθίστανται από τα pid και t^S αντίστοιχα. Ο κώδικας θα έχει τη μορφή:

```
1:  FOR (pid=-4; pid ≤8; pid++)
2:    IF (MPI_rank==Rank(pid))
3:      FOR (t^S=max(⌈(-9.5-6pid)/4⌉,⌈(-9.5-2pid)/8⌉); t^S ≤ min(⌊(39-6pid)/4⌋,⌊(29-2pid)/8⌋); t^S++)
4:        (j01; j02) = [6 4; 2 8] (pid; t^S);
5:        t=t^S-max(⌈(-9.5-6pid)/4⌉,⌈(-9.5-2pid)/8⌉);
6:        RECEIVE(pid,t^S,D^S,CC⃗);
7:        lb1=max(0,-29-2j01+j02);
8:        ub1=min(9,78-2j01+j02);
9:        FOR (j'1=lb1, j'2=lb1*2; j'1 ≤ ub1; j'1+=1, j'2+=2)
10:         lb2=max(0,-3j'1-5j01,⌈(-j'1-5j02)/2⌉);
11:         ub2=min(19,-3j'1-5j01+195, ⌊(-j'1-5j02+145)/2⌋);
12:         FOR (j'2+=⌈(lb2-j'2)/5⌉*5; j'2 ≤ ub2; j'2+=5)
13:           LA[map(j'⃗,t)]=LA[map(j'⃗-(5,0),t)]+LA[map(j'⃗-(0,5),t)];
14:         ENDFOR
15:       ENDFOR
16:       SEND(pid,t^S,D^m,CC⃗);
17:     ENDFOR
18:   ENDIF
19:   WRITE_RESULTS();
20:  ENDFOR
```

Στον παραπάνω κώδικα, οι γραμμές 1-2 μεταβάλουν την εκτέλεση του προγράμματος ανάλογα με το αναγνωριστικό του επεξεργαστή. Με αυτό τον τρόπο επιτυγχάνουμε τη μορφή SPMD (Single Program Multiple Data). Η συνάρτηση `Rank()`, όπως ειπώθηκε και προηγουμένως, μετατρέπει το pid σε αναγνωριστικό του MPI (rank). Ουσιαστικά, στο συγκεκριμένο παράδειγμα απεικονίζει το $pid = -4$ σε MPI_rank=0, το $pid = -3$ σε `MPI_rank`=1 κ.ο.κ. Η γραμμή 3 διασχίζει τους υπερκόμβους σε κάθε επεξεργαστή. Ο κώδικας για τη διάσχιση των σημείων στο εσωτερικό των υπερκόμβων είναι ίδιος με αυτόν του Παραδείγματος 3.3 (γραμμές 4, 7-12). Η εντολή ανάθεσης (γραμμή 13) έχει αλλάξει ώστε να αποθηκεύει τα δεδομένα στον Τοπικό Χώρο Δεδομένων LDS κάνοντας κατάλληλη χρήση της συνάρτησης `map()`. Ο κώδικας για τις

συναρτήσεις RECEIVE() και SEND() που υλοποιούν τη λήψη και αποστολή δεδομένων, αλλά και της WRITE_RESULTS() που γράφει τα δεδομένα στον αρχικό Χώρο Δεδομένων DS, δίνονται στη συνέχεια.

```
1:  RECEIVE(pid, t^S, D^S, C⃗C)
2:    FOR d⃗^S ∈ D^S = [ 1 0 ]
                      [ 0 1 ]
3:      IF(valid((pid,t^S)-d⃗^S) ∧ (pid,t^S)=minsucc((pid,t^S)-d⃗^S, d^m(d⃗^S)))
4:        MPI_Recv(buffer,Rank(pid-1),...);
5:        count=0;
6:        t=t^S-max(⌈(-9.5-6pid)/4⌉, ⌈(-9.5-2pid)/8⌉);
7:        lb_1=max(0,-29-2j_{0_1}+j_{0_2},5);
8:        ub_1=min(9, 78-2j_{0_1}+j_{0_2});
9:        FOR (j'_1=lb_1, j'_2=lb_1*2; j'_1 ≤ ub_1; j'_1+=1, j'_2+=2)
10:         lb_2=max(0,-3j'_1-5j_{0_1},⌈(-j'_1-5j_{0_2})/2⌉);
11:         ub_2=min(19, -3j'_1-5j_{0_1}+195, ⌊(-j'_1-5j_{0_2}+145)/2⌋);
12:         FOR (j'_2+=⌈(lb_2-j'_2)/5⌉ * 5; j'_2 ≤ ub_2; j'_2+=5)
13:           LA[map(j⃗',t)-(10,4)]=buffer[count++];
14:         ENDFOR
15:       ENDFOR
16:     ENDIF
17:   ENDFOR
```

Η έκφραση στη γραμμή 3 επαληθεύεται μόνο για $\vec{d^S} = (1,0)^T$ και έτσι ο κώδικας που παρουσιάζουμε στις γραμμές 4-17, ισχύει μόνο γι' αυτή την περίπτωση. Ουσιαστικά, στις γραμμές αυτές διασχίζεται το σύνολο επικοινωνίας που αντιστοιχεί στον προηγούμενο υπερκόμβο και τοποθετούνται στις κατάλληλες θέσεις τα δεδομένα που έχουν ληφθεί από την κλήση της MPI_Recv. Αξίζει να τονιστεί, ότι το σύνολο επικοινωνίας διασχίζεται όπως ακριβώς ένας πλήρης υπερκόμβος, με τη μόνη διαφορά της ύπαρξης του παράγοντα 5 στην έκφραση max της γραμμής 7. Στη γραμμή 13, ο παράγοντας $(10,4)$ που αφαιρείται από το map($\vec{j'}$,t) μας μεταφέρει ακριβώς

στον προηγούμενο υπερκόμβο.

```
 1:  SEND(pid, t^S, D^m, CC⃗)
 2:  FOR d⃗^m ∈ D^m
 3:     IF(∃ d^S(d⃗^m) ∈ D^S:valid((pid, t^S) + d^S(d⃗^m)))
 4:        count=0;
 5:        t=t^S-max(⌈(-9.5-6pid)/4⌉, ⌈(-9.5-2pid)/8⌉);
 6:        lb_1=max(0,-29-2j_{0_1}+j_{0_2},5);
 7:        ub_1=min(9,78-2j_{0_1}+j_{0_2});
 8:        FOR (j'_1=lb_1, j'_2=lb_1*2; j'_1 ≤ ub_1; j'_1+=1, j'_2+=2)
 9:           lb_2=max(0,-3j'_1-5j_{0_1},⌈(-j'_1-5j_{0_2})/2⌉);
10:           ub_2=min(19,-3j'_1-5j_{0_1}+195, ⌊(-j'_1-5j_{0_2}+145)/2⌋);
11:           FOR (j'_2+=⌈(lb_2-j'_2)/5⌉*5; j'_2 ≤ ub_2; j'_2+=5)
12:              buffer[count++]=LA[map(j⃗',t)];
13:           ENDFOR
14:        ENDFOR
15:        MPI_Send(buffer,Rank(pid+1),...);
16:     ENDIF
17:  ENDFOR
```

Αντίστοιχα και στην περίπτωση αποστολής δεδομένων, η έκφραση στη γραμμή 3 επαληθεύεται μόνο για $\vec{d^S} = (1,0)^T$ και ο κώδικας έχει γραφτεί για αυτή την περίπτωση. Και εδώ, η διάσχιση του συνόλου επικοινωνίας γίνεται όπως και στην περίπτωση της λήψης, μόνο που το σύνολο επικοινωνίας που διατρέχεται είναι προφανώς του τρέχοντος υπερκόμβου.

```
 1:  WRITE_RESULTS()
 2:     IF (MPI_rank!=0)
 3:        count=0;
 4:        FOR (j1''=5; j1''<15; j1''++)
 5:           FOR (j2''=4;j2'' <4+4|t|; j2''++)
 6:              buffer[count++]=LA[j1'',j2''];
 7:           ENDFOR
 8:        ENDFOR
 9:        MPI_Send(buffer,0,...);
10:     ELSE
11:        FOR (i=0;i<proc_num;i++)
12:           MPI_Recv(recv_buffer[i],i,...);
13:           count=0;
14:           FOR (j1''=5; j1''<15; j1''++)
15:              FOR (j2''=4; j2''<4+4|t|; j2''++)
16:                 A[fw(loc^-1(j'',Rank^-1(i)))]=recv_buffer[i][count++];
17:              ENDFOR
18:           ENDFOR
19:        ENDFOR
20:     ENDIF
```

Στην παραπάνω συνάρτηση έχει θεωρηθεί ότι η διεργασία με `MPI_rank`=0 είναι αυτή που αναλαμβάνει να συγκεντρώσει τα αποτελέσματα από όλες τις υπόλοιπες. Έτσι, στις γραμμές 2-9, τα δεδομένα που έχουν υπολογιστεί σε κάθε διεργασία, "πακετάρονται" και αποστέλλονται στη διεργασία με `MPI_rank`=0. Στις γραμμές 4-8 διασχίζονται όλα τα σημεία του Τοπικού Χώρου Δεδομένων LDS που περιέχουν δεδομένα που έχουν υπολογιστεί από το συγκεκριμένο επεξεργαστή. Στις γραμμές 11-20, η διεργασία με `MPI_rank`=0, λαμβάνει τα δεδομένα από όλες τις υπόλοιπες και με τη βοήθεια των συναρτήσεων $\mathtt{f}_w$, $\mathtt{loc}^{-1}$ και $\mathtt{Rank}^{-1}$ τα τοποθετεί στις κατάλληλες θέσεις του αρχικού Χώρου Δεδομένων DS. ⊣

Κεφάλαιο 5

Πειραματικά Αποτελέσματα

Στο κεφάλαιο αυτό θα παρουσιάσουμε πειραματικά αποτελέσματα από την εφαρμογή της μεθόδου αυτόματης παραγωγής παράλληλου κώδικα, που προτάθηκε στα Κεφάλαια 3 και 4. Στα πλαίσια των πειραμάτων πραγματοποιήθηκαν δύο διαφορετικές σειρές μετρήσεων. Η πρώτη σειρά αφορούσε τη διαδικασία παραγωγής και την αποδοτικότητα του σειριακού μετασχηματισμένου κώδικα, και η δεύτερη αφορούσε την επίδοση του τελικού παράλληλου κώδικα. Όσον αφορά το σειριακό μετασχηματισμένο κώδικα, στόχος των σχετικών πειραματικών μετρήσεων ήταν η σύγκριση της προτεινόμενης μεθόδου με αυτή των Ancourt και Irigoin, τόσο σε σχέση με το χρόνο μεταγλώττισης όσο και σε σχέση με την ποιότητα του παραγόμενου κώδικα. Στόχος των πειραματικών μετρήσεων που έγιναν στον τελικό παράλληλο κώδικα, ήταν η ανάδειξη της χρησιμότητας ενός εργαλείου αυτόματης παραλληλοποίησης για μη-ορθογώνια μετασχηματισμένους φωλιασμένους βρόχους. Επιπρόσθετα, με τη βοήθεια της υλοποιημένης μεθόδου, είμαστε σε θέση να παρουσιάσουμε για πρώτη φορά πειραματικά αποτελέσματα που ελέγχουν στην πράξη τη θεωρία περί επιλογής του "βέλτιστου" σχήματος για το μετασχηματισμό υπερκόμβων, που παρουσιάστηκε στις εργασίες [HS02] και [HCF03].

5.1 Πειραματικά Αποτελέσματα Σειριακού Μετασχηματισμένου Κώδικα

Όπως αναφέρθηκε και προηγουμένως, στην ενότητα αυτή θα παρουσιάσουμε πειραματικά αποτελέσματα που συγκρίνουν την προτεινόμενη μέθοδο με αυτή των Ancourt και Irigoin [AI91]. Στο εξής θα αναφερόμαστε στη μέθοδο που παρουσιάστηκε στο Κεφάλαιο 3 ως μέθοδο RI (Reduced Inequalities), ενώ στη μέθοδο των Ancourt και Irigoin ως μέθοδο AI. Υλοποιήσαμε τις μεθόδους RI και AI σε ένα εργαλείο αυτόματης παραγωγής σειριακού μετασχηματισμένου κώδικα. Εφαρμόσαμε διάφορους μετασχηματισμούς υπερκόμβων σε φωλιασμένους βρόχους και μετρήσαμε το χρόνο μεταγλώττισης αλλά και το χρόνο εκτέλεσης του παραγόμενου κώδικα. Κατ' αρχήν επιλέχθηκαν τυχαίοι χώροι επαναλήψεων και μετασχηματισμοί υπερκόμβων για να ελεγχθούν οι δύο μέθοδοι στη γενική περίπτωση. Στη συνέχεια εφαρμόσαμε τις δύο μεθόδους σε πραγματικές εφαρμογές. Επιπρόσθετα χρησιμοποιήσαμε και ένα γνωστό εργαλείο, τον Omega Calculator [KMP+95], για να παράγουμε σειριακό μετασχηματισμένο κώδικα για τα ίδια ακριβώς προβλήματα. Δώσαμε σαν είσοδο στον Omega τις ανισότητες της μεθόδου AI. Τα σχετικά αποτελέσματα αναφέρονται ως AI-Omega. Τα πειράματα εκτελέστηκαν σε έναν υπολογιστή με επεξεργαστή PIII και συχνότητα 800MHz, με μνήμη 128MB RAM. Το λειτουργικό σύστημα ήταν Linux με πυρήνα 2.4.18. Το εργαλείο παράγει C κώδικα ο οποίος με τη σειρά του μεταγλωττίζεται σε γλώσσα μηχανής με τον gcc v.2.95.4. Εφαρμόσαμε το τρίτο επίπεδο βελτιστοποίησης (optimization flag=-O3). Έγιναν δοκιμές και με χαμηλότερα επίπεδα βελτιστοποίησης όπου οι τελικοί χρόνοι εκτέλεσης ήταν μεγαλύτεροι, αλλά τα σχετικά αποτελέσματα ήταν γενικά τα ίδια. Ακολούθως παρουσιάζουμε πειραματικά αποτελέσματα για γενικούς, τυχαίους χώρους και στη συνέχεια για πραγματικές εφαρμογές.

5.1.1 Τυχαίοι Χώροι Επαναλήψεων

Στην πρώτη σειρά μετρήσεων που πραγματοποιήθηκαν, επιλέξαμε τυχαίους χώρους επαναλήψεων και τυχαίους μετασχηματισμούς υπερκόμβων. Οι χώροι αυτοί προφανώς δεν αντιστοιχούν σε πραγματικά προβλήματα, τα πειραματικά αποτελέσματα όμως δίνουν μια ιδέα της συμπεριφοράς των μεθόδων AI και RI στη γενική περίπτωση. Αξίζει να τονιστεί επιπρόσθετα, ότι ο ρόλος ενός αυτόματου παραλληλοποιητή-μεταγλωττιστή είναι να γεννήσει αποδοτικό κώδικα για οποιαδήποτε περίπτωση.

Ο Πίνακας 5.1 δείχνει τους χώρους επαναλήψεων που χρησιμοποιήθηκαν για τα πειράματα.

Οι τρεις πρώτοι χώροι (Space1-Space3) είναι δισδιάστατοι, ενώ οι επόμενοι επτά (Space4-Space10) είναι τρισδιάστατοι. Υπάρχουν διάφορα σχήματα χώρων, για παράδειγμα οι χώροι Space1 και Space4 είναι ορθογώνιοι, ενώ οι υπόλοιποι χώροι σε κάθε διάσταση έχουν προοδευτικά πιο περίπλοκο σχήμα. Κάθε ένας από τους παραπάνω χώρους μετασχηματίστηκε χρησιμοποιώντας διάφορους μετασχηματισμούς υπερκόμβων που ορίζονται από κάποιο πίνακα P. Γενικά παρατηρήσαμε ότι ο χρόνος μεταγλώττισης και ο παραγόμενος κώδικας επηρεάζονται πάρα πολύ από το σχήμα του χώρου επαναλήψεων και το σχήμα του μετασχηματισμού υπερκόμβων. Συγκεκριμένα, όσο αυξάνουμε τον αριθμό των μη-μηδενικών στοιχείων στους παράγοντες των πινάκων B και P, τόσο περισσότερο χρονοβόρα γίνεται η διαδικασία μεταγλώττισης. Αυτό εξηγείται από το γεγονός ότι κατά την απαλοιφή των συστημάτων (2.8) και (3.2) με τη μέθοδο FME, η παρουσία μεγάλου αριθμού μηδενικών στους πίνακες B και P, καθιστά τη διαδικασία πολύ πιο σύντομη. Ένα άλλο στοιχείο που παρατηρήθηκε κατά την εφαρμογή διάφορων μετασχηματισμών υπερκόμβων στους χώρους επαναλήψεων, είναι η μεγάλη μεταβολή του αριθμού των γραμμοπράξεων και του συνολικού χρόνου μεταγλώττισης, που προκαλεί μια μικρή μεταβολή στους συντελεστές του πίνακα P. Και αυτό το γεγονός εξηγείται από τη φύση της μεθόδου FME. Εδώ παρουσιάζουμε ενδεικτικά αποτελέσματα από την εφαρμογή τριών μετασχηματισμών υπερκόμβων $(P_1 - P_3)$ στους δισδιάστατους χώρους και την εφαρμογή επτά μετασχηματισμών υπερκόμβων $(P_4 - P_{10})$ στους τρισδιάστατους χώρους. Οι πίνακες $P_1 - P_{10}$ δίνονται στο Παράρτημα Γ.

	j_1		j_2		j_3		
	κάτω όριο	άνω όριο	κάτω όριο	άνω όριο	κάτω όριο	άνω όριο	# επαναλήψεων
Space1	-1999	4999	-1999	4999	-	-	48986001
Space2	-1999	4999	-1999	$4999 + 2i_1$	-	-	69983001
Space3	-4999	4999	$-4999 + 3i_1$	$4999 + 2i_1$	-	-	99980001
Space4	0	399	0	399	0	399	64000000
Space5	0	399	0	$399 + i_1$	0	399	95920000
Space6	0	399	$-i_1$	$399 + i_1$	0	399	127840000
Space7	-99	149	$-99 - i_1$	$149 + i_1$	-99	$149 + 2i_2$	22904099
Space8	0	399	$-i_1$	$399 + i_1$	i_1	$79 + 2i_2$	117635018
Space9	-99	149	$-99 - i_1$	$149 + i_1$	$-99 - i_1$	$149 + i_1 + 2i_2$	31129399
Space10	0	59	$-i_1$	$59 + i_1$	$-i_1 - 3i_2$	$59 + i_1 + 2i_2$	1994462

Πίνακας 5.1: Χώροι επαναλήψεων που χρησιμοποιήθηκαν στα πειράματα

	AI	RI	AI-Omega	AI	RI
Γραμμοπράξεις (P_2)			Χρόνος Μεταγλώττισης (P_2)		
Space1	37	10	22.56	0.28	0.27
Space2	33	10	21.56	0.28	0.27
Space3	34	10	22.78	0.29	0.26
Μέσες Γραμμοπράξεις			Μέσος Χρόνος Μεταγλώττισης		
P_1	31	10	18.88	0.27	0.26
P_2	35	10	22.30	0.28	0.27
P_3	55	12	37.63	0.36	0.3

Πίνακας 5.2: Γραμμοπράξεις της μεθόδου FME και χρόνος μεταγλώττισης (ms) για δισδιάστατους αλγορίθμους

Χρόνοι Μεταγλώττισης

Όπως αναφέρθηκε και προηγουμένως, σε κάθε χώρο επαναλήψεων του Πίνακα 5.1 εφαρμόστηκαν οι αντίστοιχοι μετασχηματισμοί υπερκόμβων. Επειδή η διαδικασία μεταγλώττισης κυριαρχείται από τις γραμμοπράξεις της μεθόδου FME, μετρήσαμε τις γραμμοπράξεις και το συνολικό χρόνο μεταγλώττισης των μεθόδων AI και RI. Επιπρόσθετα, μετρήσαμε και το χρόνο μεταγλώττισης για τον Omega Calculator, κυρίως σαν μέτρο σύγκρισης για την αποτελεσματικότητα του εργαλείου που υλοποιεί τις μεθόδους AI και RI. Τα αποτελέσματα για τους δισδιάστατους χώρους φαίνονται στον Πίνακα 5.2 και για τους τρισδιάστατους χώρους στον Πίνακα 5.3. Παρουσιάζουμε τον αριθμό των γραμμοπράξεων και το συνολικό χρόνο μεταγλώττισης για τον μετασχηματισμό P_2 στους δισδιάστατους χώρους και για το μετασχηματισμό P_3 στους τρισδιάστατους, ενώ παρουσιάζουμε το μέσο αριθμό γραμμοπράξεων και το μέσο χρόνο μεταγλώττισης για κάθε πίνακα μετασχηματισμού P_i σε όλους τους χώρους.

Παρατηρούμε ότι σε όλες τις περιπτώσεις η μέθοδος RI αποδίδει καλύτερα από τη μέθοδο AI σε σχέση με το χρόνο μεταγλώττισης. Μάλιστα η διαφορά από το χρόνο μεταγλώττισης του Omega Calculator είναι εμφανής, κάτι που αποδεικνύει ότι και σε απόλυτες τιμές, η απόδοση του εργαλείου αυτόματης παραγωγής κώδικα είναι πολύ καλή. Η διαφορά στο χρόνο μεταγλώττισης δικαιολογείται, όπως αναφέρθηκε και στην Ενότητα 3.4, από τη διαφορά στον αριθμό των ανισοτήτων και των αγνώστων των εξισώσεων (2.8) και (3.2) που χρησιμοποιούνται στη μέθοδο AI και RI αντίστοιχα. Η διαφορά αυτή στα συστήματα οδηγεί και σε πολύ μεγάλη διαφορά στον απαιτούμενο αριθμό γραμμοπράξεων κατά την απαλοιφή FME, όπως φαίνεται και στους Πίνακες 5.2 και 5.3. Αξίζει να σημειωθεί, ότι στα αποτελέσματα που παρουσιάζονται δεν έχει εφαρμοστεί η exact μέθοδος απλοποίησης (βλ. Παράρτημα Α). Η διαφορά στο χρόνο μεταγλώττισης με την εφαρμογή της απλοποίησης αυτής είναι δραματική στην περίπτωση

	AI	RI	AI-Omega	AI	RI
Γραμμοπράξεις (P_7)			Χρόνος Μεταγλώττισης (P_7)		
Space4	264	28	235.55	1.33	0.49
Space5	578	34	367.78	6.0	0.52
Space6	508	42	1,188.72	4.24	0.55
Space7	1411	38	911.38	40.78	0.54
Space8	1522	42	2,099.32	51.31	0.56
Space9	379	38	370.47	2.61	0.55
Space10	419	42	527.3	3.08	0.56
Μέσες Γραμμοπράξεις			Μέσος Χρόνος Μεταγλώττισης		
P_4	88	22	51.87	0.51	0.43
P_5	105	22	67.2	0.58	0.44
P_6	265	38	276.14	1.67	0.53
P_7	726	38	814.36	15.62	0.54
P_8	5382	36	2,901.14	1,746.64	0.53
P_9	3767	35	2,921.1	781.04	0.53
P_{10}	59563	41	2,531.58	356,508.91	0.56

Πίνακας 5.3: Γραμμοπράξεις της μεθόδου FME και χρόνος μεταγλώττισης (ms) για τρισδιάστατους αλγορίθμους

της μεθόδου AI (ο χρόνος μεταγλώττισης φτάνει περίπου τη μία ώρα), ενώ το κέρδος στην απόδοση του παραγόμενου κώδικα είναι πολύ μικρό (3% κατά μέσο όρο και 10% το μέγιστο). Ακόμα όμως και με την εφαρμογή της exact απλοποίησης, ο χρόνος μεταγλώττισης της μεθόδου RI παραμένει της τάξης των χιλιοστών του δευτερολέπτου. Επομένως, η exact μέθοδος απλοποίησης μπορεί να εφαρμοστεί πρακτικά στη μέθοδο RI, ώστε να βελτιωθεί περαιτέρω η αποδοτικότητα του τελικού κώδικα.

Παρά το γεγονός ότι η μέθοδος RI είναι πολύ ταχύτερη της AI, στα δισδιάστατα και τρισδιάστατα προβλήματα που παρουσιάζουμε, φαίνεται ότι και οι δύο μέθοδοι έχουν πολύ μικρό χρόνο μεταγλώττισης. Σε μεγαλύτερες διαστάσεις όμως αρχίζουν να παρουσιάζονται σοβαρά προβλήματα κατά τη διαδικασία της μεταγλώττισης, τόσο στη μέθοδο AI όσο και στον Omega Calculator. Εκτελέσαμε ένα μεγάλο αριθμό από τετραδιάστατα προβλήματα και κατ' αρχήν παρατηρήσαμε ότι ο χρόνος μεταγλώττισης της μεθόδου AI αυξάνεται δραματικά και γίνεται απαγορευτικός. Μερικά παραδείγματα χρειάστηκαν αρκετές ώρες ή ακόμα και μέρες για να ολοκληρώσουν τη μεταγλώττισή τους. Ακόμα πιο σοβαρή όμως ήταν η παρατήρηση, ότι η μέθοδος AI δεν ήταν σε θέση να ολοκληρώσει τη μεταγλώττιση (ουσιαστικά την απαλοιφή FME) στις μισές από τις περιπτώσεις λόγω έλλειψης μνήμης. Υπενθυμίζουμε ότι η μέθοδος FME είναι διπλά εκθετική και σε χώρο, έτσι σε αρκετά τετραδιάστατα προβλήματα ακόμα και 1Gbyte εικονικής μνήμης δεν ήταν αρκετό για να καλύψει τις ανάγκες της FME σε μνήμη. Από την άλλη

και ο Omega Calculator παρουσίασε αρκετά προβλήματα σε τετραδιάστατους αλγορίθμους (σε μικρότερο βαθμό απ' ότι η μέθοδος AI) καθώς και εδώ σχεδόν στο 50% των περιπτώσεων η απαλοιφή FME διακόπηκε λόγω υπερχείλισης. Προφανώς, μετά από ένα πολύ μεγάλο αριθμό γραμμοπράξεων κάποιοι από τους συντελεστές του συστήματος ξεπέρασαν το μέγιστο ακέραιο της αρχιτεκτονικής. Σε όλες τις παραπάνω περιπτώσεις, η μέθοδος RI ολοκλήρωσε τη διαδικασία παραγωγής κώδικα μέσα σε λίγα δευτερόλεπτα στη χειρότερη περίπτωση. Επίσης αξίζει να σημειωθεί, ότι τα τετραδιάστατα προβλήματα είναι αρκετά συχνά στην πράξη, απαντώνται για παράδειγμα στη μελέτη της ταχύτητας ροής ρευστών από ένα τρισδιάστατο αντικείμενο ή στην εύρεση του δυναμικού σε ένα τρισδιαστατο χώρο.

Χρόνοι Εκτέλεσης

Προκειμένου να αξιολογήσουμε την αποδοτικότητα του κώδικα που παράγουν οι μέθοδοι AI και RI, αλλά και ο Omega Calculator, εκτελέσαμε τους σειριακούς μετασχηματισμένους κώδικες που γεννήθηκαν από τα παραπάνω παραδείγματα. Ορίζουμε ως TOF (Tiling Overhead Factor) το λόγο:

$$TOF = \frac{\text{Χρόνος Εκτέλεσης του Σειριακού Μετασχηματισμένου Κώδικα}}{\text{Χρόνος Εκτέλεσης του Αρχικού Κώδικα}}$$

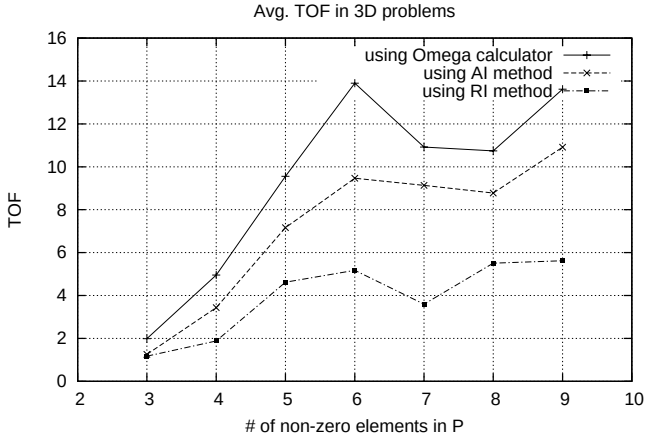
Το σώμα του βρόχου σε κάθε περίπτωση είναι μια απλή εντολή ανάθεσης σε ένα πίνακα και επομένως ο χρόνος που μετράμε είναι ουσιαστικά ο χρόνος αποτίμησης των ορίων του φωλιασμένου βρόχου. Επίσης, το μέγεθος του πίνακα είναι μικρό και τα μεγέθη των υπερκόμβων δεν είναι επιλεγμένα για να επιτυγχάνουν τοπικότητα στη λανθάνουσα μνήμη, έτσι ο σειριακός μετασχηματισμένος κώδικας δεν εμπεριέχει κάποια βελτιστοποίηση λόγω αποδοτικότερης χρήσης της λανθάνουσας μνήμης.

Ο παράγοντας TOF εκφράζει το επιπλέον κόστος που επιφέρει ο μετασχηματισμός υπερκόμβων στα νέα όρια του βρόχου. Αν ο TOF είναι πολύ μεγάλος, τότε θα είναι πολύ δύσκολη η επίτευξη επιτάχυνσης κατά την παραλληλοποίηση του βρόχου. Το ιδανικό είναι ο TOF να τείνει στη μονάδα, κάτι που σημαίνει ότι ο μετασχηματισμός υπερκόμβων επιβαρύνει ελάχιστα τον αρχικό κώδικα. Στον Πίνακα 5.4 παρουσιάζουμε τον παράγοντα TOF για τα δισδιάστατα και τρισδιάστατα παραδείγματα της προηγούμενης ενότητας, ενώ στο Σχήμα 5.1 φαίνεται ο TOF σαν συνάρτηση των μη-μηδενικών στοιχείων του πίνακα P.

Παρατηρούμε ότι η υπεροχή της μεθόδου RI ως προς τη μέθοδο AI αλλά και τον Omega

	AI-Omega	AI	RI		AI-Omega	AI	RI
TOF (2D) (P_2)				***Avg. TOF (2D)***			
Space1	6.27	4.55	1.61	P_1	2.85	1.03	1.31
Space2	6.12	4.62	1.63	P_2	6.62	4.78	1.69
Space3	7.45	5.16	1.82	P_3	8.23	6.41	3.75
TOF (3D) (P_7)				***Avg. TOF (3D)***			
Space4	15.50	9.86	4.65	P_4	1.99	1.26	1.17
Space5	16.09	10.05	5.14	P_5	4.96	3.44	1.88
Space6	16.20	10.10	5.29	P_6	9.55	7.16	4.62
Space7	12.67	9.04	4.80	P_7	13.90	9.47	5.17
Space8	12.72	8.92	4.65	P_8	11.24	9.14	3.60
Space9	11.80	8.95	4.84	P_9	10.74	8.78	5.51
Space10	12.29	9.38	6.84	P_{10}	13.62	11.07	5.62

Πίνακας 5.4: TOF για δισδιάστατα και τρισδιάστατα προβλήματα



Σχήμα 5.1: Μέσα TOF για τρισδιάστατα προβλήματα

Calculator είναι εμφανής σε όλες τις περιπτώσεις, εκτός από τον ορθογώνιο χώρο και τον ορθογώνιο μετασχηματισμό στις δύο διαστάσεις. Στα δισδιάστατα προβλήματα η μέθοδος RI έχει μέσο TOF 2.25, ενώ η μέθοδος AI και ο Omega έχουν 4.07 και 5.09 αντίστοιχα. Έχουμε δηλαδή 1.8 και 2.6 φορές γρηγορότερο κώδικα με τη μέθοδο RI από ότι με τη μέθοδο AI και τον Omega αντίστοιχα. Ανάλογη εικόνα έχουμε και στα τρισδιάστατα προβλήματα, όπου η μέθοδος RI παράγει 1.8 φορές γρηγορότερο κώδικα από την AI και 2.4 φορές γρηγορότερο από τον Omega. Όπως αναφέρθηκε και στην Ενότητα 3.4, τα αποτελέσματα αυτά είναι αναμενόμενα, καθώς η μέθοδος RI παράγει πιο απλές εκφράσεις για τη διάσχιση των σημείων στο εσωτερικό των υπερκόμβων. Το Σχήμα 5.1 δίνει επίσης ενδιαφέρουσες πληροφορίες για τη συμπεριφορά των δύο μεθόδων και του Omega, σε σχέση με το σχήμα του μετασχηματισμού υπερκόμβων.

Παρατηρούμε ότι για ορθογώνιους μετασχηματισμούς υπερκόμβων (τρία μη μηδενικά στοιχεία στη διαγώνιο του πίνακα P) ο παράγοντας TOF, παραμένει μικρός και στις τρεις περιπτώσεις, με τον κώδικα της μεθόδου RI να έχει TOF που πλησιάζει πιο κοντά στη μονάδα. Όσο το σχήμα του μετασχηματισμού υπερκόμβων γίνεται πιο περίπλοκο (προστίθενται μηδενικά στον πίνακα P), τόσο ο παράγοντας TOF μεγαλώνει. Η μέθοδος RI όμως δείχνει να διατηρεί τον παράγοντα TOF σε χαμηλά επίπεδα ακόμα και για πίνακες μετασχηματισμών P με λίγα ή καθόλου μηδενικά στοιχεία. Συμπεραίνουμε ότι η μέθοδος RI συμπεριφέρεται σχεδόν ιδανικά στους ορθογώνιους μετασχηματισμούς υπερκόμβων και διατηρεί την αποδοτικότητα του παραγόμενου κώδικα ακόμα και στους πλέον περίπλοκους μετασχηματισμούς.

5.1.2 Πραγματικές Εφαρμογές

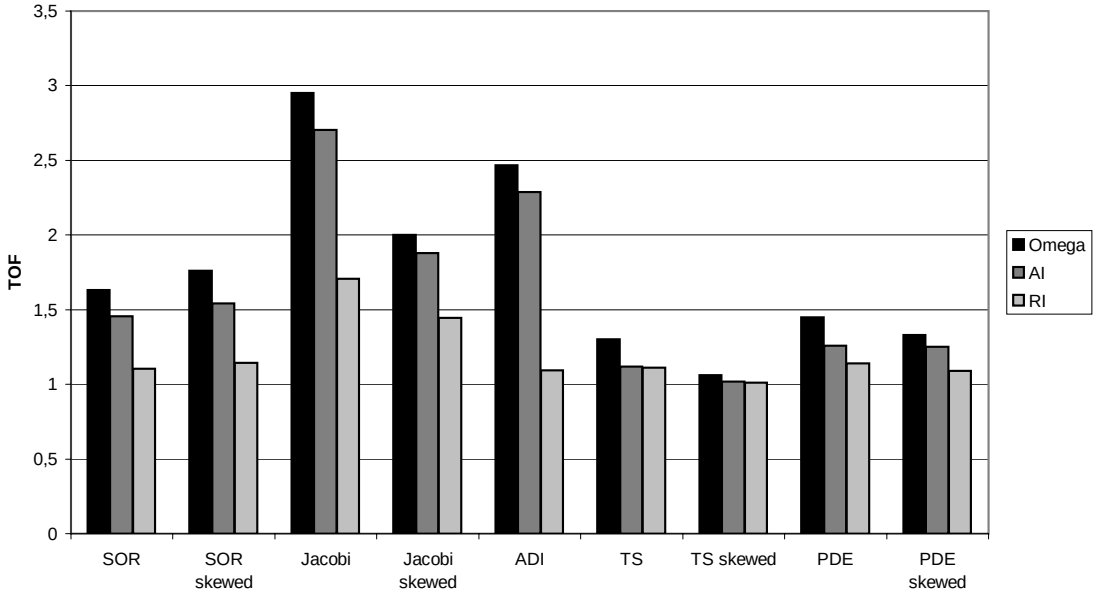
Η τελευταία σειρά μετρήσεων που αφορούσε το σειριακό μετασχηματισμένο κώδικα, έγινε σε πραγματικές εφαρμογές. Συγκεκριμένα χρησιμοποιήθηκαν οι κώδικες Jacobi, SOR, ADI, TS και PDE που παρουσιάζονται στο Παράρτημα Β. Οι μετασχηματισμοί υπερκόμβων H που χρησιμοποιήθηκαν προκύπτουν από τον κώνο υπερκόμβων του προβλήματος. Στις περιπτώσεις των προβλημάτων SOR, Jacobi και PDE, ο κώνος υπερκόμβων περιλαμβάνει τέσσερα διανύσματα. Οι συνδυασμοί των διανυσμάτων αυτών ανά τρία δίνουν τους πίνακες $H_1 - H_4$ σε κάθε περίπτωση. Όλοι οι κώδικες εκτός από τον ADI και τον TS, χρειάζεται να μετασχηματιστούν με αλλαγή κλίσης για να μπορεί να εφαρμοστεί ορθογώνιος μετασχηματισμός υπερκόμβων. Τα αποτελέσματα από την εφαρμογή των μεθόδων RI και AI στους παραπάνω κώδικες φαίνονται στον Πίνακα 5.5.

Παρατηρούμε ότι σε όλα τα παραδείγματα ο χρόνος μεταγλώττισης είναι πολύ μικρός και έτσι η επιλογή ανάμεσα στις μεθόδους, αναφορικά με αυτό το κριτήριο, δεν έχει ιδιαίτερη σημασία. Το γεγονός αυτό το αναμέναμε, καθώς οι χώροι είναι τρισδιάστατοι και ορθογώνιοι, ενώ στην πλειονότητά τους οι μετασχηματισμοί υπερκόμβων δεν είναι περίπλοκοι. Η απόδοση όμως της μεθόδου RI, σε σχέση με την ποιότητα του παραγόμενου κώδικα είναι αισθητά καλύτερη. Όπως φαίνεται και στα ραβδογράμματα του Σχήματος 5.2, σε όλες τις περιπτώσεις η μέθοδος RI έχει μικρότερο παράγοντα TOF από τη μέθοδο AI και τον Omega. Η διαφορά αυτή σε μερικά παραδείγματα είναι ιδιαίτερα σημαντική, π.χ. στον αλγόριθμο Jacobi η μέθοδος RI επιτυγχάνει σχεδόν 60% μικρότερο TOF και στη μέθοδο ADI σχεδόν 110% μικρότερο TOF σε σύγκριση με τη μέθοδο AI. Αυτό που επίσης παρουσιάζει ενδιαφέρον στον Πίνακα 5.5 είναι ότι, η αύξηση του αριθμού των γραμμοπράξεων κατά τη μεταγλώττιση, η οποία γενικά προέρχεται από περίπλοκα σχήματα στους μετασχηματισμούς υπερκόμβων και/ή στους χώ-

		Γραμμοπράξεις		Χρόνος Μεταγλώττισης (ms)			TOF		
		AI	**RI**	**AI-Omega**	**AI**	**RI**	**AI-Omega**	**AI**	**RI**
SOR	H_1	99	22	53.03	0.50	0.42	1.47	1.20	1.05
	H_2	107	22	50.27	0.53	0.42	1.50	1.21	1.01
	H_3	118	22	49.01	0.57	0.42	1.75	1.63	1.05
	H_4	165	40	90.04	0.77	0.5	1.80	1.78	1.30
SOR skewed	H_1	99	22	42.09	0.53	0.41	1.59	1.29	1.06
	H_2	107	22	40.60	0.53	0.42	1.60	1.29	1.06
	H_3	118	22	57.9	0.57	0.42	1.90	1.73	1.12
	H_4	165	40	91.97	0.77	0.51	1.95	1.86	1.34
Jacobi	H_1	645	28	346.99	5.3	0.46	2.08	1.91	1.57
	H_2	645	28	347.96	5.26	0.47	2.09	1.92	1.60
	H_3	800	28	362.5	8.86	0.47	2.06	1.90	1.56
	H_4	3207	46	1,353.55	194.88	0.53	5.58	5.09	2.10
Jacobi skewed	H_1	645	28	251.885	4.93	0.48	1.99	1.88	1.44
	H_2	645	28	248.27	4.98	0.47	1.98	1.87	1.46
	H_3	800	28	229.34	8.19	0.48	2.02	1.89	1.45
	H_4	691	28	238.82	5.95	0.47	2.01	1.88	1.43
ADI	H_1	180	28	47.42	0.85	0.46	1.46	1.47	1.07
TS	H_1	1014	22	49.96	3.32	0.55	1.3	1.12	1.11
TS (skewed)	H_1	286	22	26.72	1.27	0.54	1.06	1.02	1.01
PDE	H_1	868	22	40.8	2.84	0.55	1.31	1.09	1.07
	H_2	1014	22	49.91	3.31	0.55	1.29	1.08	1.07
	H_3	1342	22	49.16	4.35	0.55	1.58	1.34	1.12
	H_4	3481	368	90.8	11.64	1.71	1.64	1.51	1.19
PDE skewed	H_1	1086	22	44.66	3.51	0.54	1.36	1.08	1.07
	H_2	1260	22	40.63	4.1	0.55	1.42	1.35	1.09
	H_3	275	22	26.75	1.27	0.54	1.06	1.06	1.06
	H_4	2249	212	50.38	7.21	1.22	1.47	1.51	1.13

Πίνακας 5.5: Γραμμοπράξεις, χρόνος μεταγλώττισης και TOF για πραγματικές εφαρμογές

ρους επαναλήψεων, οδηγεί γενικά σε λιγότερο αποδοτικό κώδικα. Και αυτό το γεγονός είναι αναμενόμενο, καθώς οι νέες γραμμοπράξεις "γεννούν" ανισότητες που πρέπει να περιληφθούν κατά την αποτίμηση των νέων ορίων του μετασχηματισμένου βρόχου.



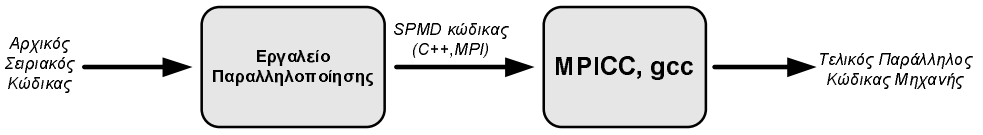
Σχήμα 5.2: TOF για πραγματικές εφαρμογές

5.2 Πειραματικά Αποτελέσματα του Τελικού Παράλληλου Κώδικα

Οι μέθοδοι παραγωγής σειριακού μετασχηματισμένου κώδικα και τελικού παράλληλου κώδικα, που παρουσιάστηκαν στα Κεφάλαια 3 και 4, εφαρμόστηκαν στην κατασκευή ενός εργαλείου αυτόματης παραγωγής παράλληλου κώδικα. Το εργαλείο αυτό παράγει C++ κώδικα με κλήσεις σε ρουτίνες της βιβλιοθήκης του MPI. Με τη βοήθεια του εργαλείου αυτού μπορούμε να γεννήσουμε αυτόματα SPMD κώδικα για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με οποιονδήποτε μετασχηματισμό που ορίζεται από κάποιο πίνακα H ή P. Αξίζει να τονιστεί ότι, από τη μελέτη της σχετικής βιβλιογραφίας, δεν έχει υποπέσει στην αντίληψή μας η ύπαρξη κάποιου αντίστοιχου εργαλείου που να παράγει παράλληλο κώδικα για γενικούς, μη-ορθογώνιους μετασχηματισμούς υπερκόμβων.

Σκοπός των πειραμάτων αυτής της ενότητας είναι να επιβεβαιωθούν προηγούμενες ερευνητικές εργασίες σχετικά με την επιλογή του σχήματος του μετασχηματισμού υπερκόμβων. Όπως αναφέρθηκε και στην Ενότητα 3.1, οι εργασίες [HS02] και [HCF03] προτείνουν την επιλογή του σχήματος του μετασχηματισμού υπερκόμβων από την επιφάνεια του κώνου υπερκόμβων του αλ-

γορίθμου. Ο κώνος υπερκόμβων είναι κάθετος στον κώνο των εξαρτήσεων (Σχήμα 3.1). Για το σκοπό αυτό θα εφαρμόσουμε ορθογώνιους και μη-ορθογώνιους μετασχηματισμούς υπερκόμβων σε μία σειρά πραγματικών εφαρμογών, που παρουσιάζονται στο Παράρτημα Β. Οι ενότητες που ακολουθούν παρουσιάζουν τα πειραματικά αποτελέσματα και τα σχετικά συμπεράσματα για τους κώδικες Jacobi, SOR, ADI, TS και PDE.



Σχήμα 5.3: Παραγωγή τελικού παράλληλου κώδικα μηχανής με τη χρήση του υλοποιημένου εργαλείου και των mpiCC και gcc

Ο παράλληλος κώδικας εκτελέστηκε σε μία συστοιχία 16 όμοιων υπολογιστών με επεξεργαστή Pentium III, συχνότητα λειτουργίας 500MHz και μνήμη RAM 256MB. Το λειτουργικό σύστημα των κόμβων είναι Linux με πυρήνα 2.4.20. Το δίκτυο διασύνδεσης του συστήματος είναι Fast Ethernet. Η μεταγλώττιση του παράλληλου κώδικα που παράγει το εργαλείο σε γλώσσα μηχανής, γίνεται με τον mpiCC που χρησιμοποιεί και τον gcc v.2.95.4. (Σχήμα 5.3). Χρησιμοποιήθηκε το δεύτερο επίπεδο βελτιστοποίησης στον gcc (-O2 option).

5.2.1 Jacobi

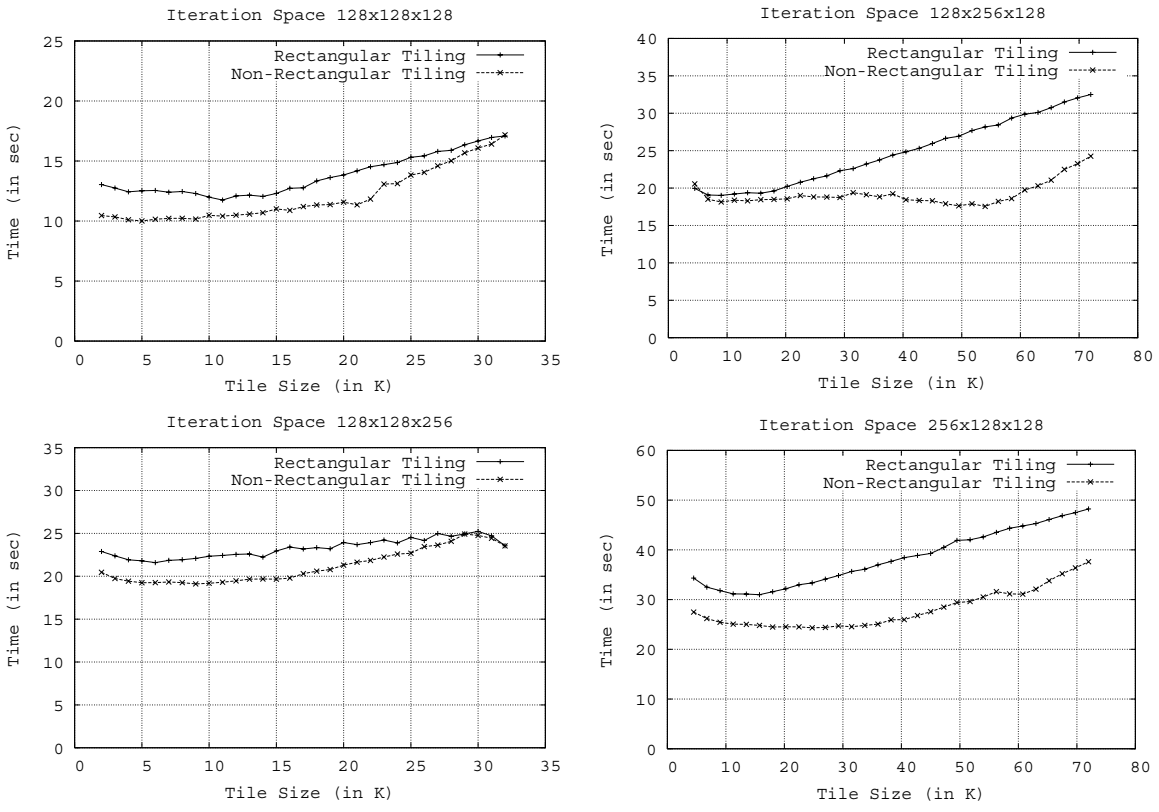
Ο πρώτος αλγόριθμος που παραλληλοποιήθηκε με τη βοήθεια των τεχνικών που παρουσιάστηκαν στην παρούσα εργασία, είναι η μέθοδος επίλυσης μερικών διαφορικών εξισώσεων του Jacobi. Πληροφορίες για τον αλγόριθμο, καθώς και η ανάλυση που αφορά τις εξαρτήσεις του, δίνονται στο Παράρτημα Β. Για να πραγματοποιήσουμε μία σαφή σύγκριση ανάμεσα σε ένα ορθογώνιο και ένα μη-ορθογώνιο μετασχηματισμό υπερκόμβων, διεξάγουμε τα πειράματα στο μετασχηματισμένο με αλλαγή κλίσης χώρο, όπου η εφαρμογή του ορθογώνιου μετασχηματισμού υπερκόμβων είναι έγκυρη (βλ. Παράρτημα Β). Σ' αυτό το χώρο, ο κώνος υπερκόμβων είναι ο $\mathcal{C} = \begin{bmatrix} -3 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$. Σύμφωνα με τις εργασίες [HS02] και [HCF03], το βέλτιστο σχήμα του μετασχηματισμού υπερκόμβων προκύπτει αν επιλεγούν διανύσματα-γραμμές για τον

πίνακα H από την επιφάνεια του κώνου αυτού. Δεδομένου ότι ο ορθογώνιος μετασχηματισμός υπερκόμβων εκφράζεται με έναν πίνακα της μορφής $H_r = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$ $(x, y, z \in Z^+)$, η σύγκριση με ένα μη-ορθογώνιο μετασχηματισμό που θα προέκυπτε για παράδειγμα επιλέγοντας τρεις γραμμές από τον πίνακα $\mathcal{C}$ θα ήταν δύσκολη, καθώς σ' αυτή την περίπτωση για κάθε μετασχηματισμό θα είχαμε διαφορετικό μέγεθος υπερκόμβου και διαφορετικό όγκο επικοινωνίας ανά υπερκόμβο. Επιλέγουμε σαν μη-ορθογώνιο μετασχηματισμό υπερκόμβων ένα μετασχηματισμό που διατηρεί τις δύο πρώτες γραμμές του H_r, ενώ σαν τρίτη γραμμή επιλέγουμε την τρίτη γραμμή του $\mathcal{C}$. Έτσι έχουμε $H_{nr} = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ -\frac{1}{z} & -\frac{1}{z} & \frac{1}{z} \end{bmatrix}$. Αν επιλέξουμε κοινούς παράγοντες x, y και z για τους δύο πίνακες H_r και H_{nr}, τότε τα μεγέθη των υπερκόμβων που προκύπτουν είναι ίσα, αφού $V_{comp}(H_r) = \frac{1}{|det(H_r)|} = xyz$ και $V_{comp}(H_{nr}) = \frac{1}{|det(H_{nr})|} = xyz$. Επιπρόσθετα, αν απεικονίσουμε υπερκόμβους κατά μήκος της τρίτης διάστασης στον ίδιο επεξεργαστή, τότε και ο όγκος επικοινωνίας ανά υπερκόμβο σε κάθε περίπτωση είναι ο ίδιος ($V_{comm}(H_r) = V_{comm}(H_{nr})$, βλ. Σχέση 2.6). Με αυτό τον τρόπο, η διαφορά στο συνολικό χρόνο εκτέλεσης που τυχόν θα προκύψει, θα οφείλεται αποκλειστικά στη διαφορά του άεργου χρόνου των επεξεργαστών που επιφέρουν τα διαφορετικά σχήματα των υπερκόμβων.

Για να αποκτήσουμε μια πιο θεωρητική οπτική των αναμενόμενων αποτελεσμάτων, θα πραγματοποιήσουμε μια απλοποιημένη θεωρητική ανάλυση. Αν υποθέσουμε ότι το λεξικογραφικά μεγαλύτερο σημείο του αρχικού χώρου είναι το $j_{max} = (T, I, J)$, τότε το σημείο αυτό μετά το μετασχηματισμό αλλαγής κλίσης θα μετασχηματιστεί στο $j'_{max} = (T, T + I, T + J)$. Το j'_{max} μετά το μετασχηματισμό υπερκόμβων θα ανήκει στον υπερκόμβο $\lfloor H j'_{max} \rfloor$. Χρησιμοποιώντας σαν γραμμικό διάνυσμα δρομολόγησης το $\Pi = [1, 1, 1]$, τότε το σημείο j'_{max} θα εκτελεστεί τη χρονική στιγμή $t_r = \Pi \lfloor H_r j'_{max} \rfloor = \frac{T}{x} + \frac{T+I}{y} + \frac{T+J}{z}$ με τον ορθογώνιο μετασχηματισμό υπερκόμβων και τη χρονική στιγμή $t_{nr} = \Pi \lfloor H_{nr} j'_{max} \rfloor = \frac{T}{x} + \frac{T+I}{y} + \frac{T+J}{z} - \frac{T}{z} - \frac{T+I}{z}$ με τον μη-ορθογώνιο μετασχηματισμό υπερκόμβων. Προφανώς $t_{nr} < t_r$, άρα αναμένουμε ο μη-ορθογώνιος μετασχηματισμός, να επιτύχει μικρότερο συνολικό χρόνο εκτέλεσης από τον ορθογώνιο.

Πραγματοποιήσαμε μετρήσεις για το συνολικό χρόνο εκτέλεσης σε τέσσερις διαφορετικούς χώρους επαναλήψεων. Σε κάθε περίπτωση επιλέξαμε κατάλληλα τις παραμέτρους x και y, ώστε ο αριθμός των απαιτούμενων διεργασιών να παραμένει σταθερός και μεταβάλαμε την παράμετρο z, για να μειώσουμε ή να αυξήσουμε το μέγεθος του υπερκόμβου. Τα αποτελέσματα των συνολικών χρόνων εκτέλεσης στους τέσσερις χώρους επαναλήψεων, για τον ορθογώνιο και το

μη-ορθογώνιο μετασχηματισμό φαίνονται στο Σχήμα 5.4.



Σχήμα 5.4: Jacobi: Χρόνοι εκτέλεσης για ορθογώνιο (rectangular) και μη-ορθογώνιο (non-rectangular) μετασχηματισμό σε τέσσερις χώρους επαναλήψεων

Παρατηρούμε ότι, όπως αναμενόταν, σε όλους τους χώρους και για όλα τα μεγέθη υπερκόμβων, ο μη-ορθογώνιος μετασχηματισμός επιτυγχάνει μικρότερους συνολικούς χρόνους εκτέλεσης από τον ορθογώνιο. Ο Πίνακας 5.6 συνοψίζει τα πειραματικά αποτελέσματα. Ο ελάχιστος χρόνος εκτέλεσης σε κάθε χώρο επιτυγχάνεται από το μη-ορθογώνιο μετασχηματισμό και μάλιστα με σημαντική διαφορά από τον ορθογώνιο. Ιδιαίτερα σημαντική είναι και η διαφορά στο μέσο χρόνο εκτέλεσης που φτάνει μέχρι το 36.7% στο χώρο $256 \times 128 \times 128$. Το γεγονός αυτό είναι ιδιαίτερα εντυπωσιακό, καθώς αυτή η μεγάλη μείωση στο συνολικό χρόνο εκτέλεσης προήλθε με μια μικρή αλλαγή στο σχήμα του μετασχηματισμού υπερκόμβων.

Χώρος Επαναλήψεων	t_r min	t_{nr} min	% diff.	t_r avg.	t_{nr} avg.	% diff.
$\mathbf{128 \times 128 \times 128}$	11.75	10.01	17.3	13.77	12.06	14.2
$\mathbf{128 \times 128 \times 256}$	21.59	19.11	13	23.26	21.14	10
$\mathbf{128 \times 256 \times 128}$	19.02	17.55	8.4	24.7	19.2	28.6
$\mathbf{256 \times 128 \times 128}$	30.98	24.32	27.4	38.3	28.02	36.7

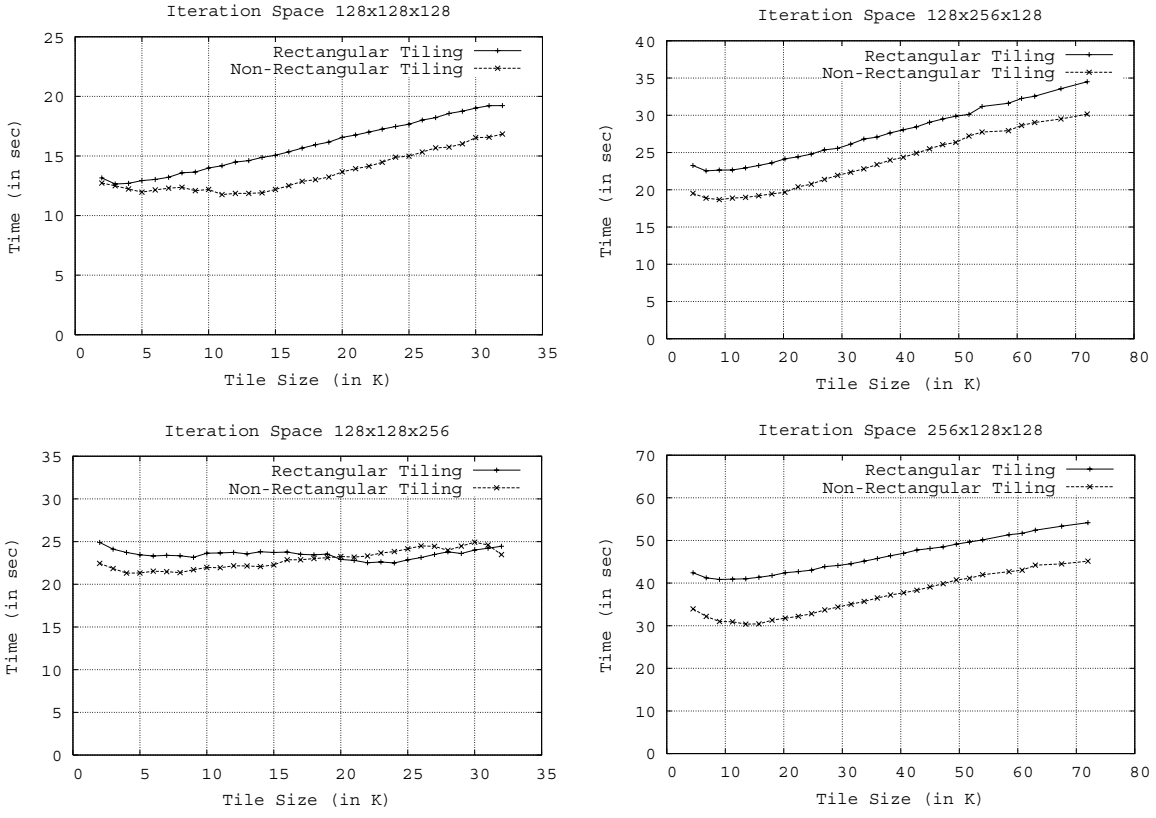
Πίνακας 5.6: Jacobi: Ελάχιστοι και μέσοι χρόνοι εκτέλεσης (sec) για τους τέσσερις χώρους επαναλήψεων

5.2.2 SOR

Η μέθοδος Gauss Successive Over-Relaxation (SOR) χρησιμοποιείται, όπως και η μέθοδος του Jacobi, για την επίλυση μερικών διαφορικών εξισώσεων. Και σ' αυτή την περίπτωση, επειδή ο αρχικός χώρος περιέχει εξαρτήσεις με αρνητικούς παράγοντες, απαιτείται εφαρμογή του μετασχηματισμού αλλαγής κλίσης, για να είναι δυνατή η εφαρμογή ορθογώνιου μετασχηματισμού υπερκόμβων. Έτσι, και εδώ οι σχετικές μετρήσεις και συγκρίσεις θα γίνουν στο μετασχηματισμένο χώρο. Στο χώρο αυτό ο κώνος υπερκόμβων είναι ο $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \\ -2 & 1 & 1 \end{bmatrix}$.

Ο μη-ορθογώνιος πίνακας $H_{nr} = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ -\frac{1}{z} & 0 & \frac{1}{z} \end{bmatrix}$ είναι παράλληλος στην επιφάνεια του κώνου υπερκόμβων. Ακολουθώντας παρόμοια ανάλυση με την περίπτωση της μεθόδου Jacobi, προκύπτει ότι το σημείο $j_{max} = (T, I, J)$, που στο μετασχηματισμένο χώρο γίνεται $j'_{max} = (T, T+I, T+2J)$, θα εκτελεστεί τη χρονική στιγμή $t_r = \frac{T}{x} + \frac{T+I}{y} + \frac{2T+J}{z}$ με τον ορθογώνιο μετασχηματισμό και τη χρονική στιγμή $t_{nr} = \frac{T}{x} + \frac{T+I}{y} + \frac{2T+J}{z} - \frac{T}{z} = t_r - \frac{T}{z}$ με τον μη-ορθογώνιο. Προφανώς και εδώ ισχύει $t_{nr} < t_r$ και επομένως αναμένουμε ο μη-ορθογώνιος μετασχηματισμός να έχει μικρότερο συνολικό χρόνο εκτέλεσης από τον ορθογώνιο.

Στο Σχήμα 5.5 φαίνονται οι χρόνοι εκτέλεσης του αλγορίθμου SOR σε τέσσερις χώρους για διάφορα μεγέθη υπερκόμβων. Και σ' αυτή την περίπτωση, όπως αναμενόταν, ο μη-ορθογώνιος μετασχηματισμός επιφέρει επιτάχυνση στην εκτέλεση του παραλληλοποιημένου κώδικα. Στον Πίνακα 5.7 παρατηρούμε συνοπτικά τη βελτίωση στην επίδοση της παράλληλης εκτέλεσης λόγω του σχήματος του μετασχηματισμού υπερκόμβων. Και εδώ ιδιαίτερα εντυπωσιακό είναι το γεγονός, ότι αλλάζοντας ένα μόνο παράγοντα στον πίνακα μετασχηματισμού H επιτυγχάνουμε κατά μέσο όρο 25% βελτίωση στο χώρο $256 \times 128 \times 128$.



Σχήμα 5.5: SOR: Χρόνοι εκτέλεσης για ορθογώνιο (rectangular) και μη-ορθογώνιο (non-rectangular) μετασχηματισμό σε τέσσερις χώρους επαναλήψεων

5.2.3 ADI

Η τρίτη εφαρμογή που παραλληλοποιήθηκε, προέρχεται και αυτή από το πεδίο της αριθμητικής ανάλυσης και συγκεκριμένα από την επαναληπτική επίλυση μερικών διαφορικών εξισώσεων. Όπως φαίνεται και στο Παράρτημα Β, τα διανύσματα εξάρτησης στον αρχικό χώρο περιλαμβάνουν μη-αρνητικές συντεταγμένες και έτσι ο ορθογώνιος μετασχηματισμός υπερκόμβων είναι έγκυρος στον αρχικό χώρο. Ο κώνος υπερκόμβων στο χώρο αυτό είναι $\mathcal{C} = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

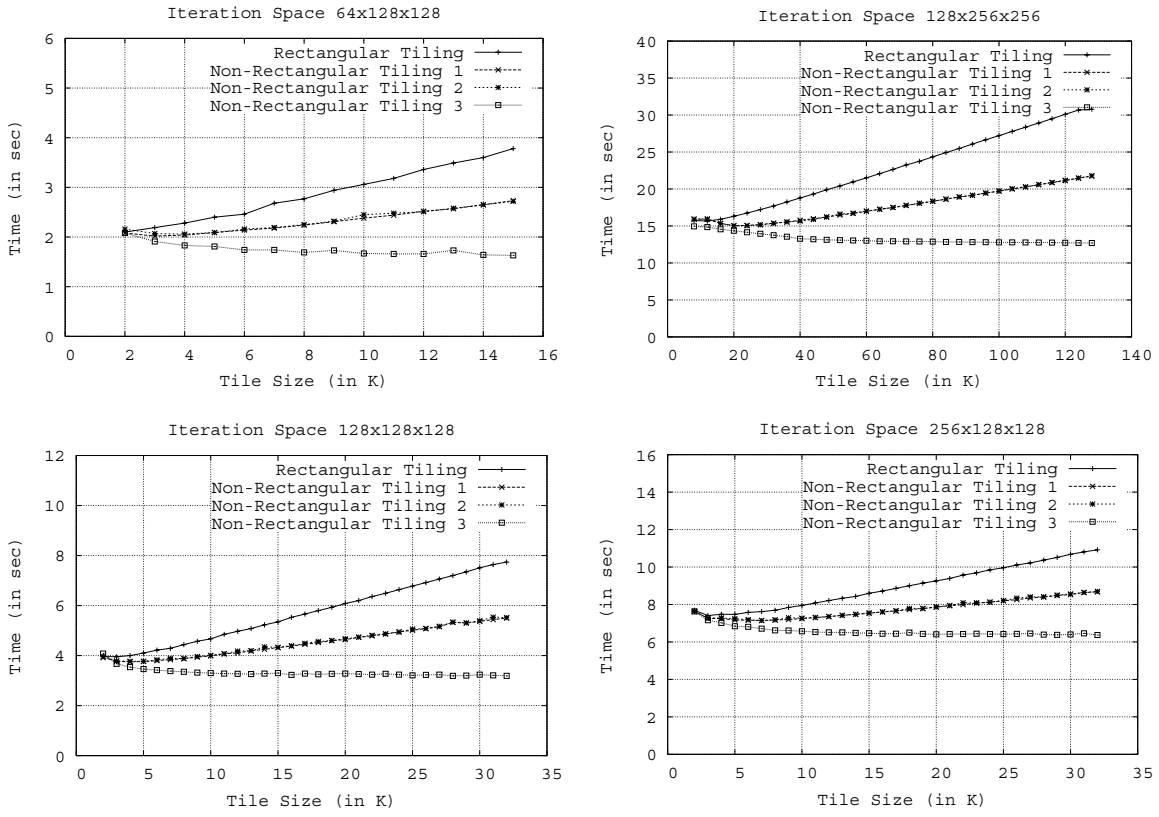
Εφαρμόσαμε τρεις διαφορετικούς μη-ορθογώνιους μετασχηματισμούς υπερκόμβων στο χώ-

Χώρος Επαναλήψεων	t_r min	t_{nr} min	% diff.	t_r avg.	t_{nr} avg.	% diff.
$\mathbf{128 \times 128 \times 128}$	12.65	11.76	7.5	15.77	13.56	16.3
$\mathbf{128 \times 128 \times 256}$	22.5	21.31	5.6	23.51	22.89	2.7
$\mathbf{128 \times 256 \times 128}$	22.54	18.68	20.6	24.63	21.21	16.1
$\mathbf{256 \times 128 \times 128}$	40.85	30.37	34.5	41.63	33.15	25.5

Πίνακας 5.7: SOR: Ελάχιστοι και μέσοι χρόνοι εκτέλεσης (sec) για τους τέσσερις χώρους επαναλήψεων

ρο αυτό. Οι μετασχηματισμοί περιγράφονται από τους πίνακες $H_{nr1} = \begin{bmatrix} \frac{1}{x} & -\frac{1}{x} & 0 \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$, $H_{nr2} = \begin{bmatrix} \frac{1}{x} & 0 & -\frac{1}{x} \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$ και $H_{nr3} = \begin{bmatrix} \frac{1}{x} & -\frac{1}{x} & -\frac{1}{x} \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$. Ο τρίτος μετασχηματισμός είναι παράλληλος στην επιφάνεια του κώνου υπερκόμβων και επομένως θεωρείται βέλτιστος. Ακολουθώντας ανάλυση όμοια με προηγουμένως, υπολογίζουμε τους θεωρητικούς χρόνους του ορθογώνιου και των μη-ορθογώνιων μετασχηματισμών. Έχουμε $j_{max} = (T, I, J)$, και έτσι $t_r = \frac{T}{x} + \frac{I}{y} + \frac{J}{z}$, $t_{nr1} = t_r - \frac{I}{x}$, $t_{nr2} = t_r - \frac{J}{x}$ και $t_{nr3} = t_r - \frac{I}{x} - \frac{J}{x}$. Δηλαδή ισχύει $t_{nr3} < t_{nr1}, t_{nr2} < t_r$. Αναμένουμε δηλαδή ο μετασχηματισμός H_{nr3} να επιφέρει μικρότερο χρόνο εκτέλεσης από τους H_{nr1} και H_{nr2} οι οποίοι με τη σειρά τους θα έχουν μικρότερο χρόνο εκτέλεσης από τον ορθογώνιο μετασχηματισμό H_r. Επίσης, αν $I = J$ αναμένουμε ότι οι H_{nr1} και H_{nr2} θα έχουν τον ίδιο χρόνο εκτέλεσης.

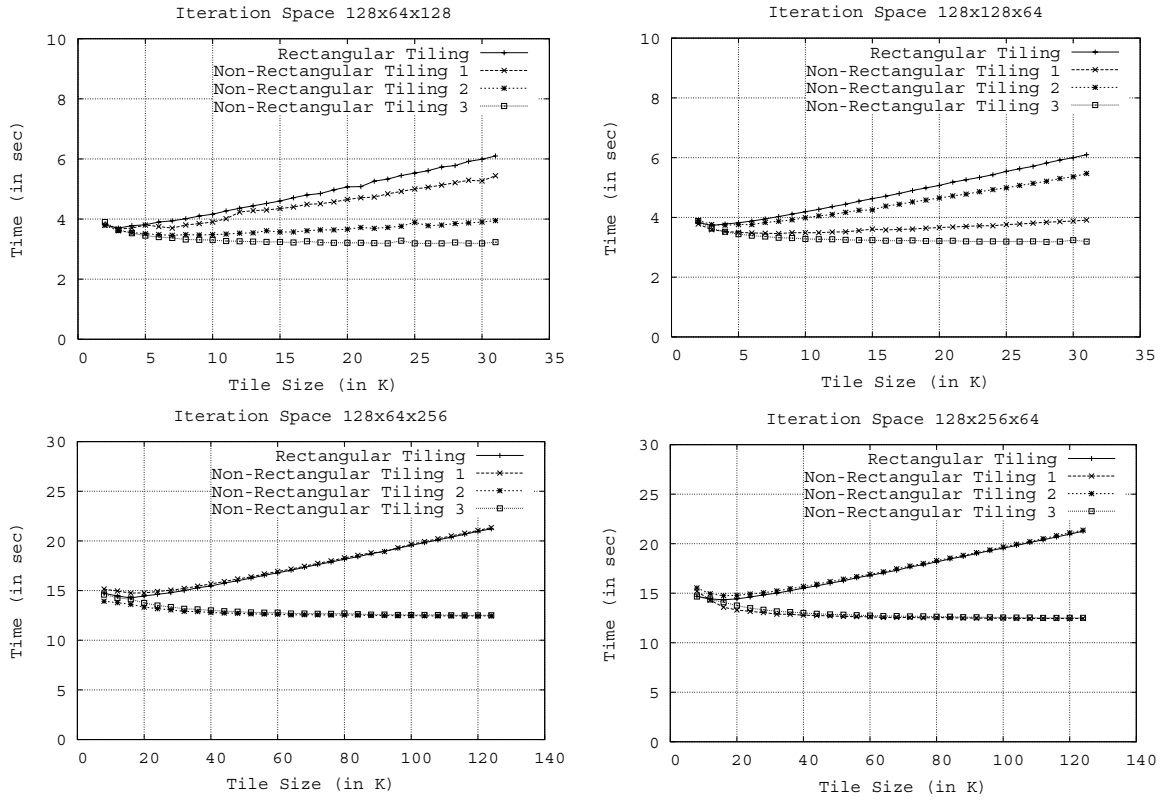
Σ' αυτή την περίπτωση απεικονίζουμε τους υπερκόμβους της πρώτης διάστασης στον ίδιο επεξεργαστή. Με αυτό τον τρόπο επιτυγχάνουμε ίδιο μέγεθος υπερκόμβου και ίδιο όγκο επικοινωνίας ανά υπερκόμβο και για τους τέσσερις μετασχηματισμούς. Κατά την παραλληλοποίηση του κώδικα ADI, διατηρήσαμε τις παραμέτρους y και z ίσες και σταθερές και μεταβάλλαμε την παράμετρο x για να αυξομειώσουμε το μέγεθος του υπερκόμβου. Έτσι, οι τυχόν διαφορές στο χρόνο εκτέλεσης θα οφείλονται αποκλειστικά στη διαφοροποίηση του άεργου χρόνου των επεξεργαστών. Στο Σχήμα 5.6 παρουσιάζουμε τους χρόνους εκτέλεσης των τεσσάρων μετασχηματισμών σε τέσσερις χώρους επαναλήψεων για τους οποίους ισχύει $I = J$ και για διάφορα μεγέθη υπερκόμβων. Και στην περίπτωση της μεθόδου ADI τα αποτελέσματα είναι όπως τα αναμέναμε. Ο βέλτιστος μη-ορθογώνιος μετασχηματισμός H_{nr3} επιτυγχάνει τους μικρότερους χρόνους εκτέλεσης, ενώ ακολουθούν οι δύο άλλοι μη-ορθογώνιοι μετασχηματισμοί H_{nr1} και H_{nr2} οι οποίοι μάλιστα έχουν και σχεδόν ίδιους χρόνους εκτέλεσης, γεγονός που επιβεβαιώνει



Σχήμα 5.6: ADI: Χρόνοι εκτέλεσης για έναν ορθογώνιο (rectangular) και τρεις μη-ορθογώνιους (non-rectangular 1-3) μετασχηματισμούς σε τέσσερις χώρους επαναλήψεων ($I = J$)

το θεωρητικά αναμενόμενο αφού $I = J$.

Ιδιαίτερο ενδιαφέρον παρουσιάζει στην περίπτωση του ADI, η μελέτη της επίδρασης του σχήματος του χώρου επαναλήψεων στον τελικό χρόνο εκτέλεσης. Περισσότερες λεπτομέρειες για τον παράγοντα αυτό δίνονται στις εργασίες [HS98] και [HS02]. Με βάση το απλοποιημένο μοντέλο θεωρητικής ανάλυσης που χρησιμοποιούμε εδώ, αναμένουμε ότι αν $I > J$, τότε $t_{nr1} < t_{nr2}$ και μάλιστα αν $I >> J$ τότε $t_{nr1} \rightarrow t_{nr3}$ και $t_{nr2} \rightarrow t_r$. Αντιστρόφως, για $J > I$, τότε $t_{nr1} > t_{nr2}$ και για $J >> I$ $t_{nr2} \rightarrow t_{nr3}$ και $t_{nr1} \rightarrow t_r$. Για να εξετάσουμε την επίδραση του μεγέθους και του σχήματος του χώρου επαναλήψεων στον τελικό χρόνο εκτέλεσης, πάντα σε συνάρτηση με το σχήμα του μετασχηματισμού υπερκόμβων, πραγματοποιήσαμε μια σειρά πειραμάτων σε τέσσερις χώρους για τους οποίους ισχύει $I \neq J$. Τα αποτελέσματα φαίνονται στο Σχήμα 5.7. Παρατηρούμε ότι τα πειραματικά αποτελέσματα επαληθεύουν



Σχήμα 5.7: ADI: Χρόνοι εκτέλεσης για έναν ορθογώνιο (rectangular) και τρεις μη-ορθογώνιους (non-rectangular 1-3) μετασχηματισμούς σε τέσσερις χώρους επαναλήψεων ($I \neq J$)

με μεγάλη ακρίβεια τα θεωρητικά αναμενόμενα. Στο χώρο $128 \times 64 \times 128$ πράγματι έχουμε $t_{nr3} < t_{nr1} < t_{nr2} < t_r$, ενώ στο χώρο $128 \times 128 \times 64$ οι χρόνοι t_{nr1} και t_{nr2} αντιστρέφονται, οπότε ισχύει $t_{nr3} < t_{nr2} < t_{nr1} < t_r$. Στο χώρο $128 \times 64 \times 256$, όπου η διαφορά των I και J είναι πολύ μεγάλη, παρατηρούμε ότι $t_{nr2} \rightarrow t_{nr3}$ και $t_{nr1} \rightarrow t_r$ και αντίστοιχα στο χώρο $128 \times 256 \times 64$ έχουμε $t_{nr1} \rightarrow t_{nr3}$ και $t_{nr2} \rightarrow t_r$. Ο Πίνακας 5.8 συνοψίζει τα αποτελέσματα και για τους οκτώ χώρους επαναλήψεων. Σημειώνεται ότι η % διαφορά που παρουσιάζεται, αφορά τη διαφορά ανάμεσα στο βέλτιστο μη-ορθογώνιο μετασχηματισμό H_{nr3} και στον ορθογώνιο H_r. Παρατηρούμε ότι εδώ οι διαφορές είναι ακόμα μεγαλύτερες σε όλους τους χώρους και φτάνουν ακόμα και το 72% στο χώρο $128 \times 256 \times 256$.

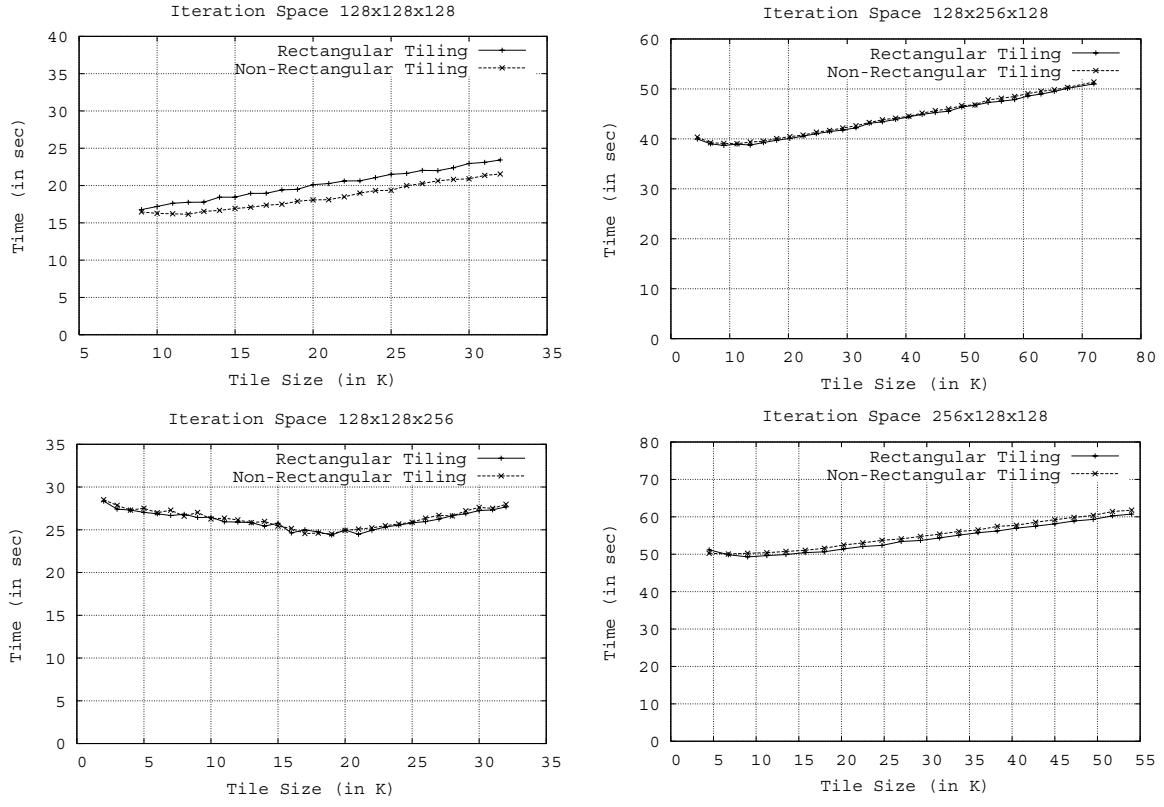
Χώρος Επαναλήψεων	t_r min	t_{nr1} min	t_{nr2} min	t_{nr3} min	% diff.	t_r avg.	t_{nr1} avg.	t_{nr2} avg.	t_{nr3} avg.	% diff.
$\mathbf{64 \times 128 \times 128}$	2.1	2.02	2.06	1.63	28.8	2.88	2.31	2.33	1.75	64.3
$\mathbf{128 \times 128 \times 128}$	3.95	3.75	3.77	3.19	23.8	5.69	4.51	4.53	3.32	71.5
$\mathbf{128 \times 256 \times 256}$	15.67	15.03	15.03	12.7	23.4	22.8	17.82	17.82	13.25	72.1
$\mathbf{256 \times 128 \times 128}$	7.41	7.15	7.14	6.37	16.3	8.95	7.76	7.78	6.57	36.2
$\mathbf{128 \times 64 \times 128}$	3.7	3.67	3.46	3.19	16	4.78	4.44	3.65	3.29	31
$\mathbf{128 \times 64 \times 256}$	14.27	14.75	12.44	12.51	14.1	17.35	12.52	12.76	12.95	34
$\mathbf{128 \times 128 \times 64}$	3.73	3.46	3.74	3.18	17.3	4.80	3.64	4.46	3.29	45.9
$\mathbf{128 \times 256 \times 64}$	14.34	12.44	14.75	12.51	14.6	17.4	12.8	17.5	12.95	34.1

Πίνακας 5.8: ADI: Ελάχιστοι και μέσοι χρόνοι εκτέλεσης (sec) για τους οκτώ χώρους επαναλήψεων

5.2.4 TS

Ο αλγόριθμος TS χρησιμοποιείται στην επεξεργασία εικόνας. Στον πίνακα εξαρτήσεων του αρχικού φωλιασμένου βρόχου υπάρχουν αρνητικοί παράγοντες και για το λόγο αυτό ο χώρος πρέπει να μετασχηματιστεί με αλλαγή κλίσης για να μπορούμε να εφαρμόσουμε ορθογώνιο μετασχηματισμό υπερκόμβων. Ο κώνος υπερκόμβων του αρχικού χώρου είναι ο $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ ενώ στον μετασχηματισμένο χώρο ο κώνος υπερκόμβων είναι ο $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Παρατηρούμε πως στο μετασχηματισμένο χώρο ο ορθογώνιος μετασχηματισμός υπερκόμβων είναι βέλτιστος. Επομένως, στο παράδειγμα αυτό δεν υπάρχει κάποιος μη-ορθογώνιος μετασχηματισμός που να οδηγεί σε μικρότερο χρόνο εκτέλεσης από τον ορθογώνιο. Αυτό που έχει όμως ενδιαφέρον να εξετάσει κανείς στη συγκεκριμένη περίπτωση, είναι οι χρόνοι εκτέλεσης του μη-ορθογώνιου μετασχηματισμού υπερκόμβων στον αρχικό χώρο και του ορθογώνιου μετασχηματισμού υπερκόμβων, στον χώρο που έχει μετασχηματιστεί με αλλαγή κλίσης. Έτσι, εφαρμόσαμε τον μη-ορθογώνιο μετασχηματισμό $H_{nr} = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ \frac{1}{y} & \frac{1}{y} & 0 \\ \frac{1}{z} & \frac{1}{z} & \frac{1}{z} \end{bmatrix}$ στον αρχικό χώρο και τον ορθογώνιο μετασχηματισμό, κατά τα γνωστά, $H_r = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$ στο μετασχηματισμένο χώρο. Αναμένουμε θεωρητικά οι δύο περιπτώσεις να έχουν ίδιο συνολικό χρόνο εκτέλεσης, ενώ τυχόν διαφορές θα οφείλονται στην υλοποίηση του μετασχηματισμού υπερκόμβων στην

πρώτη περίπτωση και στην υλοποίηση του μετασχηματισμού αλλαγής κλίσης στη δεύτερη. Τα αποτελέσματα για τέσσερις χώρους και διαφορετικά μεγέθη υπερκόμβων φαίνονται στο Σχήμα 5.8.



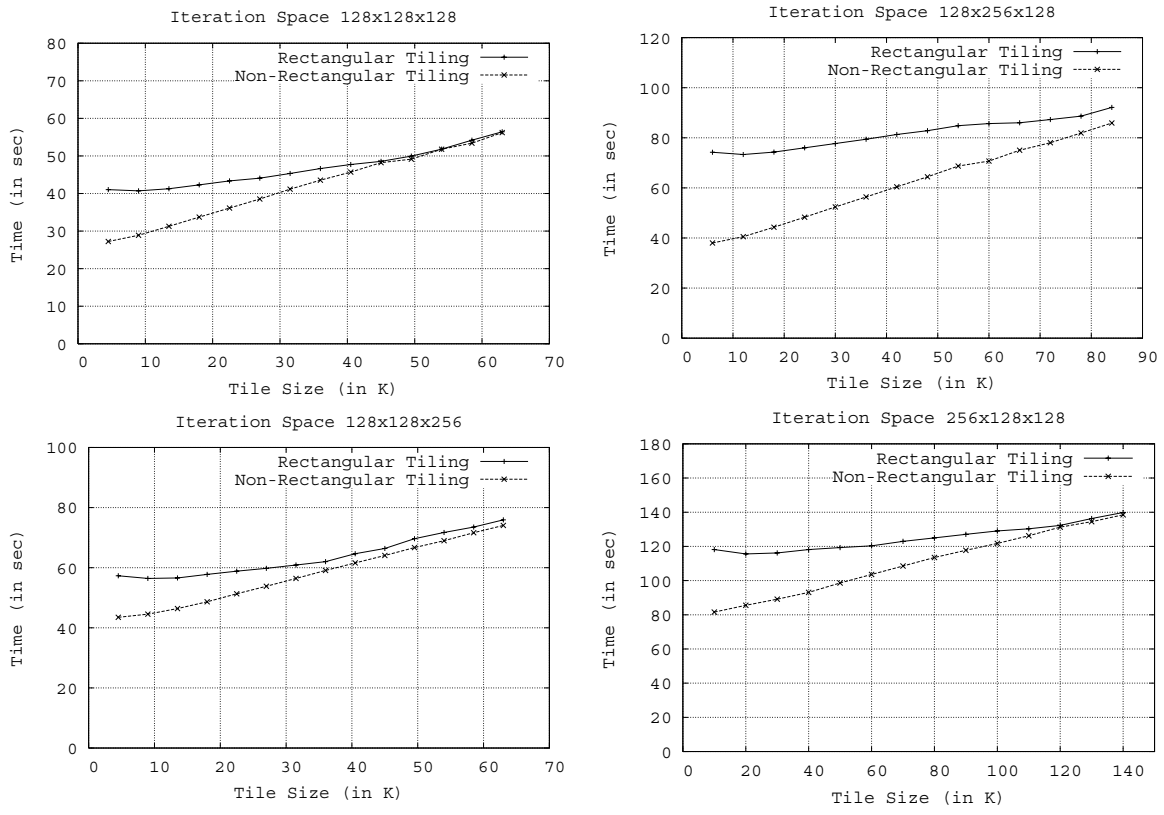
Σχήμα 5.8: TS: Χρόνοι εκτέλεσης για ορθογώνιο (rectangular) και μη-ορθογώνιο (non-rectangular) μετασχηματισμό σε τέσσερις χώρους επαναλήψεων

Παρατηρούμε ότι, όπως αναμενόταν, οι χρόνοι εκτέλεσης συμπίπτουν σε πολύ μεγάλο βαθμό στους τρεις από τους τέσσερις χώρους. Ακόμα και στο χώρο $128 \times 128 \times 128$ που υπάρχει διαφορά στους χρόνους εκτέλεσης αυτή δεν ξεπερνά το 6-7%. Εκτός από την επιβεβαίωση των θεωρητικά αναμενόμενων χρόνων εκτέλεσης, εξάγουμε επίσης το συμπέρασμα ότι η απευθείας εφαρμογή του μη-ορθογώνιου μετασχηματισμού υπερκόμβων στον αρχικό χώρο, είναι το ίδιο αποδοτική ή και αποδοτικότερη από τη διαδοχική εφαρμογή αλλαγής κλίσης και ορθογώνιου μετασχηματισμού υπερκόμβων.

5.2.5 PDE

Ο κώδικας PDE χρησιμοποιείται κι αυτός κατά την επίλυση μερικών διαφορικών εξισώσεων. Έχει εννιά διανύσματα εξάρτησης και καθώς μερικά από αυτά περιέχουν αρνητικές συντεταγμένες, ο αρχικός χώρος του προβλήματος μετασχηματίζεται με αλλαγή κλίσης. Ο κώνος υπερκόμβων στον τελικό χώρο είναι ο $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Παρατηρούμε πως ο ορθογώνιος μετασχηματισμός ομαδοποίησης είναι παράλληλος σε τρία διανύσματα-γραμμή του κώνου και επομένως είναι θεωρητικά βέλτιστος. Η δεύτερη γραμμή του κώνου ομαδοποίησης όμως εισάγει και μη-ορθογώνιους μετασχηματισμούς σαν υποψήφιους βέλτιστους. Σ' αυτή την τελευταία σειρά πειραμάτων εφαρμόσαμε στο πρόβλημα ορθογώνιο μετασχηματισμό ομαδοποίησης που όπως έχει ειπωθεί περιγράφεται από τον πίνακα $H_r = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$ και τον μη-ορθογώνιο μετασχηματισμό που προκύπτει από την αντικατάσταση της τελευταίας γραμμής του H_r από την τρίτη γραμμή του κώνου ομαδοποίησης. Έτσι, $H_{nr} = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ 0 & \frac{1}{y} & 0 \\ \frac{1}{z} & -\frac{1}{z} & \frac{1}{z} \end{bmatrix}$. Αν $j_{max} = (T, I, J)$ είναι το λεξικογραφικά μεγαλύτερο σημείο στον αρχικό χώρο τότε με βάση το μετασχηματισμό αλλαγής κλίσης το σημείο αυτό θα μετασχηματιστεί στο $j'_{max} = (T, 2T + I, 2T + I + J)$. Κατά τα γνωστά, το σημείο αυτό θα εκτελεστεί τη χρονική στιγμή $t_r = \frac{T}{x} + \frac{2T+I}{y} + \frac{2T+I+J}{z}$ με τον ορθογώνιο μετασχηματισμό και τη χρονική στιγμή $t_{nr} = \frac{T}{x} + \frac{2T+I}{y} + \frac{T}{z} - \frac{2T+I}{z} + \frac{2T+I+J}{z}$. Προκύπτει, ότι παρά το γεγονός ότι οι δύο μετασχηματισμοί είναι θεωρητικά βέλτιστοι, ο μη-ορθογώνιος αναμένεται να είναι καλύτερος αφού $t_{nr} < t_r$. Στο Σχήμα 5.9 παρουσιάζουμε τους χρόνους εκτέλεσης για τέσσερις χώρους επαναλήψεων και διάφορα μεγέθη υπερκόμβων.

Παρατηρούμε και σ' αυτή την περίπτωση ότι ο μη-ορθογώνιος μετασχηματισμός οδηγεί σε σημαντικά μικρότερους χρόνους εκτέλεσης από τον ορθογώνιο. Ο Πίνακας 5.9 συνοψίζει τις διαφορές αυτές. Οι διαφορές στο συγκεκριμένο παράδειγμα είναι πολύ μεγάλες στις ελάχιστες τιμές του χρόνου εκτέλεσης, οι οποίες και επιτυγχάνονται για μικρά μεγέθη υπερκόμβων. Με την αύξηση του μεγέθους των υπερκόμβων οι δύο μετασχηματισμοί φαίνεται να συγκλίνουν σε μεγάλους χρόνους εκτέλεσης. Χαρακτηριστικό είναι πάντως, ότι στο χώρο $128 \times 256 \times 128$ η % διαφορά στον ελάχιστο χρόνο εκτέλεσης φτάνει το 92,4%.



Σχήμα 5.9: PDE: Χρόνοι εκτέλεσης για ορθογώνιο (rectangular) και μη-ορθογώνιο (non-rectangular) μετασχηματισμό σε τέσσερις χώρους επαναλήψεων

5.3 Συμπεράσματα

Από τα αποτελέσματα των πειραμάτων που παρουσιάστηκαν στο κεφάλαιο αυτό, μπορούν να εξαχθούν τα εξής συμπεράσματα: Όσον αφορά το σειριακό μετασχηματισμένο κώδικα, η μέθοδος που προτάθηκε στο Κεφάλαιο 3, απλοποιεί σημαντικά τη διαδικασία παραγωγής του σειριακού μετασχηματισμένου κώδικα, κάτι που έχει άμεσο αποτέλεσμα στο χρόνο μεταγλώττισης. Αν και ο χρόνος μεταγλώττισης δεν είναι πρωταρχικής σημασίας, δεδομένου ότι ένα πρόγραμμα μεταγλωττίζεται μία φορά ενώ εκτελείται πολλές, η μείωση του χρόνου μεταγλώττισης μπορεί να είναι ιδιαίτερα σημαντική σε κατηγορίες αλγορίθμων (π.χ. τεσσάρων διαστάσεων βρόχοι), όπου η μέθοδος AI απαιτεί δεκάδες ώρες για να ολοκληρωθεί. Η απλοποίηση της διαδικασίας μεταγλώττισης επιτρέπει και την ολοκλήρωση της μεταγλώττισης σε αλγορίθμους που δεν θα ήταν δυνατόν να μεταγλωττιστούν λόγω έλλειψης μνήμης ή λόγω υπερχείλισης.

Χώρος Επαναλήψεων	t_r **min**	t_{nr} **min**	**% diff.**	t_r **avg.**	t_{nr} **avg.**	**% diff.**
$\mathbf{128 \times 128 \times 128}$	40.7	27.22	49.5	46.67	41.78	11.7
$\mathbf{128 \times 128 \times 256}$	56.42	43.5	29.7	63.67	57.91	9.9
$\mathbf{128 \times 256 \times 128}$	73.31	37.99	92.4	81.7	61.78	32.23
$\mathbf{256 \times 128 \times 128}$	115.59	81.56	41.7	125.04	110.25	13.4

Πίνακας 5.9: PDE: Ελάχιστοι και μέσοι χρόνοι εκτέλεσης (sec) για τους τέσσερις χώρους επαναλήψεων

Η πλέον σημαντική όμως συνεισφορά της μεθόδου που προτάθηκε για την παραγωγή του σειριακού μετασχηματισμένου κώδικα, είναι στην αποδοτικότητα του παραγόμενου κώδικα. Από τα πειραματικά αποτελέσματα της Ενότητας 5.1 προκύπτει, ότι η βελτίωση στο χρόνο εκτέλεσης του σειριακού μετασχηματισμένου κώδικα που επιφέρει η προτεινόμενη μέθοδος είναι ιδιαίτερα σημαντική. Προφανώς το γεγονός αυτό είναι ζωτικής σημασίας, καθώς η επίτευξη επιδόσεων κατά την παραλληλοποίηση ενός αλγορίθμου είναι ο αντικειμενικός στόχος, και θα πρέπει να αναζητάται σε όλα τα στάδια της παραλληλοποίησης. Ακόμα και οι πιο έξυπνες τεχνικές θεωρητικής παραλληλοποίησης θα έχουν μειωμένη απόδοση, αν η μέθοδος υλοποίησής τους παράγει κακής ποιότητας τελικό κώδικα.

Όσον αφορά τα πειραματικά αποτελέσματα σε σχέση με τον τελικό παράλληλο κώδικα, μπορούν να εξαχθούν τα εξής συμπεράσματα: Το σχήμα του μετασχηματισμού υπερκόμβων έχει πολύ μεγάλη επίδραση στο συνολικό χρόνο εκτέλεσης ενός μετασχηματισμένου χώρου επαναλήψεων. Σε μεγάλο αριθμό αλγορίθμων, χώρων επανάληψης και μεγεθών υπερκόμβων, τα πειραματικά αποτελέσματα έδειξαν ότι η επιλογή του σχήματος του μετασχηματισμού υπερκόμβων από τον κώνο των υπερκόμβων, είναι δυνατόν να μειώσει το χρόνο εκτέλεσης ακόμα και σε ποσοστά που φτάνουν το 90%. Η μείωση αυτή προέρχεται από τη μείωση του άεργου χρόνου των επεξεργαστών όπως θεωρητικά έχει αποδειχθεί στις εργασίες [HS02] και [HCF03]. Τα αποτελέσματα αυτά αναδεικνύουν την αξία ενός μεταγλωττιστή που θα είναι σε θέση να παράγει κώδικα για γενικούς, ορθογώνιους και μη-ορθογώνιους, μετασχηματισμούς υπερκόμβων.

Κεφάλαιο 6

Συμπεράσματα και Προτάσεις

Ο κύριος σκοπός της παρούσας διατριβής ήταν η ανάπτυξη μίας αποδοτικής μεθόδου για την αυτόματη παραγωγή παράλληλου SPMD κώδικα, για φωλιασμένους βρόχους που έχουν μετασχηματιστεί με το μετασχηματισμό υπερκόμβων. Η τελική αρχιτεκτονική του παραγόμενου κώδικα είναι μία αρχιτεκτονική κατανεμημένης μνήμης, η οποία, όπως αναφέρθηκε, αποτελεί την πιο δύσκολη τελική αρχιτεκτονική κατά την αυτόματη παραγωγή παράλληλου κώδικα. Ο σκοπός αυτός επετεύχθη δίνοντας επιπρόσθετα ιδιαίτερη έμφαση και στην αποδοτικότητα του παραγόμενου κώδικα. Τα συμπεράσματα που προκύπτουν από την έρευνα που οδήγησε στην εργασία αυτή συνοπτικά είναι τα εξής:

- Κάθε μετασχηματισμός σε ένα φωλιασμένο βρόχο επιφέρει στον τελικό κώδικα μία επιβάρυνση που οφείλεται ακριβώς στην υλοποίηση του μετασχηματισμού. Στην περίπτωση του μετασχηματισμού υπερκόμβων, όπου ένας αρχικός n-διάστατος βρόχος μετασχηματίζεται σε ένα 2n-διάστατο, η επιβάρυνση αυτή μπορεί να είναι ιδιαίτερα σημαντική, ειδικά όταν εφαρμόζεται ένας μη-ορθογώνιος μετασχηματισμός. Αν στην παραπάνω παρατήρηση προστεθεί και το γεγονός ότι η παραλληλοποίηση ενός σειριακού προγράμματος επιφέρει επιπλέον επιβάρυνση, τότε συμπεραίνει εύκολα κανείς ότι η παραγωγή παράλληλου κώδικα για γενικούς μετασχηματισμούς υπερκόμβων, είναι μία διαδικασία που πρέπει να γίνεται με μεγάλη προσοχή, ώστε η αποδοτικότητα του παραγόμενου κώδικα να είναι αυτή που αναμένεται κατά την παράλληλη εκτέλεσή του σε μία αρχιτεκτονική υψηλών επιδόσεων.

- Η μέθοδος που παρουσιάστηκε στην παρούσα εργασία δίνει έμφαση στην αποδοτικότητα του παράλληλου κώδικα με δύο τρόπους:

  1. Κατ' αρχήν, κατά την παραγωγή του σειριακού μετασχηματισμένου κώδικα εφαρμόζεται ένας κατάλληλος μετασχηματισμός, ο οποίος απλοποιεί σημαντικά τις εκφράσεις αποτίμησης των ορίων του μετασχηματισμένου βρόχου, μεταφέροντας την πολυπλοκότητα σε πράξεις μεταξύ ακεραίων εκτός των ορίων του βρόχου. Δεδομένου ότι αυτές οι πράξεις εκτελούνται πολύ αποδοτικά στους μοντέρνους επεξεργαστές και βελτιστοποιούνται πολύ πιο εύκολα από τους μεταγλωττιστές σε σχέση με τις πολύπλοκες συναρτήσεις αποτίμησης στα όρια των βρόχων, επιτυγχάνεται σημαντική μείωση στην πολυπλοκότητα του παραγόμενου κώδικα.
  2. Κατά τη φάση της παραλληλοποίησης του σειριακού μετασχηματισμένου κώδικα και με τη βοήθεια του ίδιου μετασχηματισμού, δίνεται η δυνατότητα περαιτέρω απλοποίησης της διαδικασίας, καθώς όλα τα σύνολα δεδομένων που διασχίζονται είναι ορθογώνιοι χώροι, γεγονός που επίσης απλοποιεί τις εκφράσεις αποτίμησης των ορίων των βρόχων που τα διασχίζουν. Σαν γενικό συμπέρασμα μπορεί να ειπωθεί ότι η παραλληλοποίηση των μη-ορθογώνια μετασχηματισμένων βρόχων υλοποιείται εισφέροντας σχεδόν την ίδια επιβάρυνση με αυτή που επιφέρεται κατά την υλοποίηση ορθογώνιων μετασχηματισμών.

- Πέρα από την προσπάθεια διατήρησης της αποδοτικότητας του παραγόμενου κώδικα σε όσο το δυνατόν υψηλότερα επίπεδα, η μέθοδος που παρουσιάστηκε μειώνει δραστικά και το χρόνο μεταγλώττισης. Παρά το γεγονός ότι ο χρόνος μεταγλώττισης είναι δευτερεύουσας σημασίας, σε κάποιες περιπτώσεις ο χρόνος αυτός μπορεί να είναι τόσο μεγάλος που να καθιστά τη μεταγλώττιση κάποιων μετασχηματισμένων βρόχων απαγορευτική. Αυτό οφείλεται στην ιδιαίτερα μεγάλη πολυπλοκότητα της μεθόδου Fourier-Motzkin που χρησιμοποιείται για την απαλοιφή συστημάτων ανισοτήτων κατά τη μεταγλώττιση. Εξ αιτίας αυτής της πολυπλοκότητας είναι επίσης πιθανόν σε κάποιες περιπτώσεις να μην είναι δυνατή η ολοκλήρωση της μεθόδου, λόγω έλλειψης μνήμης ή υπερχείλισης μετά από πολύ μεγάλο αριθμό γραμμοπράξεων. Η προτεινόμενη μέθοδος ουσιαστικά εξαλείφει τα παραπάνω προβλήματα, τροφοδοτώντας τη μέθοδο Fourier-Motzkin με πολύ μικρότερα συστήματα ανισοτήτων τα οποία απαλείφονται στη χειρότερη περίπτωση σε μερικά λεπτά της ώρας, εκεί που η ήδη προτεινόμενη μέθοδος [AI91] θα απαιτούσε ώρες ή και μέρες, ή δεν θα ήταν σε θέση να ολοκληρώσει τη μεταγλώττιση.

- Η προτεινόμενη μέθοδος παραγωγής παράλληλου κώδικα υλοποιήθηκε σε ένα εργαλείο αυ-

τόματης παραλληλοποίησης, προκειμένου να ελεγχθεί στην πράξη η θεωρία περί επιλογής ενός βέλτιστου σχήματος στον μετασχηματισμό υπερκόμβων. Με τη βοήθεια του εργαλείου αυτού ήμασταν για πρώτη φορά σε θέση να εφαρμόσουμε μη-ορθογώνια σχήματα μετασχηματισμών και να συγκρίνουμε το συνολικό χρόνο εκτέλεσής τους, με αυτόν που προκύπτει από την εφαρμογή των ορθογώνιων μετασχηματισμών που χρησιμοποιούνται έως τώρα. Από την παραπάνω πειραματική διαδικασία προέκυψαν πολύ ενδιαφέροντα συμπεράσματα σε σχέση με την επίδραση του σχήματος του μετασχηματισμού υπερκόμβων στο συνολικό χρόνο εκτέλεσης ενός φωλιασμένου βρόχου. Συγκεκριμένα, τα πειράματα που διεξήχθησαν επιβεβαίωσαν τη θεωρία με βάση την οποία, το σχήμα του μετασχηματισμού ομαδοποίησης θα πρέπει να προκύπτει από τον κώνο υπερκόμβων του αλγορίθμου. Μάλιστα, ιδιαίτερα εντυπωσιακό είναι το γεγονός ότι σε κάποιες περιπτώσεις, μεταβάλλοντας μόνο ένα ή δύο στοιχεία στον πίνακα μετασχηματισμού υπερκόμβων, ο συνολικός χρόνος εκτέλεσης του παράλληλου προγράμματος μπορεί να μειωθεί σε ποσοστά που ξεπερνούν το 30% και μπορεί να φτάνουν ακόμα και το 70%.

Η παρούσα εργασία αποτελεί μία ολοκληρωμένη προσπάθεια αυτόματης παραγωγής παράλληλου κώδικα για γενικούς μετασχηματισμούς υπερκόμβων. Σαν προέκταση της μεθόδου που παρουσιάστηκε, προτείνεται να διερευνηθούν τα εξής:

- Η προτεινόμενη μέθοδος, όπως αναφέρθηκε και στην Ενότητα 4.2, προϋποθέτει την ύπαρξη απεριόριστου αριθμού διεργασιών, οι οποίες στο στάδιο της εκτέλεσης από το MPI ανατίθενται σε πεπερασμένο αριθμό υπολογιστικών κόμβων. Το γεγονός αυτό μπορεί να οδηγήσει στη δημιουργία άεργων διεργασιών, ή διεργασιών στις οποίες δεν έχει κατανεμηθεί ισομερώς το υπολογιστικό φορτίο. Μία λύση του προβλήματος αυτού είναι να ληφθεί υπ' όψη ο ακριβής αριθμός των επεξεργαστών κατά το στάδιο της μεταγλώττισης και να γίνουν οι κατάλληλες τροποποιήσεις στον παραγόμενο κώδικα, ώστε να δημιουργούνται διεργασίες ίσες με τον αριθμό των επεξεργαστών και με ομοιόμορφα κατανεμημένο το υπολογιστικό φορτίο.

- Τα τελευταία χρόνια χρησιμοποιούνται ολοένα και περισσότερο διεπίπεδες αρχιτεκτονικές παράλληλης επεξεργασίας, στο πρώτο επίπεδο των οποίων βρίσκονται κόμβοι με συστήματα μοιραζόμενης μνήμης, τα οποία διασυνδέονται στο υψηλότερο επίπεδο και διαμορφώνουν ένα σύστημα κατανεμημένης μνήμης. Η διαδικασία παραγωγής παράλληλου κώδικα θα μπορούσε να λάβει υπ' όψη τα ιδιαίτερα χαρακτηριστικά της αρχιτεκτονικής αυτής, προκειμένου να παράγει κώδικα που θα εκμεταλλεύεται περισσότερο τα χαρακτηριστικά αυτά. Συγκεκριμένα, είναι δυνατόν να γίνει υλοποίηση ενός υβριδικού προγραμματι-

στικού μοντέλου, το οποίο θα επιτρέπει τη δημιουργία νημάτων (threads) στο εσωτερικό κάθε υπολογιστικού κόμβου μοιραζόμενης μνήμης (π.χ. με χρήση Pthreads, ή της βιβλιοθήκης OpenMP) τα οποία θα επικοινωνούν με τα νήματα στους άλλους υπολογιστικούς κόμβους με τη βοήθεια του προγραμματιστικού μοντέλου για συστήματα κατανεμημένης μνήμης (π.χ. με τη βιβλιοθήκη MPI όπως στην παρούσα εργασία).

Τα συμπεράσματα της εργασίας αυτής μπορούν να εφαρμοστούν για την αποδοτικότερη εκτέλεση μίας μεγάλης κατηγορίας αλγορίθμων, οι οποίοι περιγράφονται με τη βοήθεια DOACROSS βρόχων. Οι βρόχοι αυτοί μπορεί να προκύψουν στην επεξεργασία εικόνας ή κατά την επίλυση μερικών διαφορικών εξισώσεων που περιγράφουν προβλήματα αρχικών/συνοριακών τιμών. Κλασσικά προβλήματα που μοντελοποιούνται σαν προβλήματα αρχικών/συνοριακών τιμών, είναι η ροή ρευστών (CFD), η διάχυση θερμότητας, η διάδοση δυναμικού κ.ά. Συνεπώς, τα αποτελέσματα της εργασίας αυτής μπορούν να χρησιμοποιηθούν για τη βελτιστοποίηση ενός μεγάλου αριθμού εφαρμογών από ένα ευρύ φάσμα επιστημονικών περιοχών.

Οι παραπάνω εφαρμογές, εξ αιτίας ακριβώς της ύπαρξης n γραμμικά ανεξάρτητων διανυσμάτων εξάρτησης, είναι δύσκολα παραλληλοποιήσιμες, καθώς απαιτούν πολύ συχνή επικοινωνία για την αποστολή και λήψη δεδομένων. Ο μετασχηματισμός υπερκόμβων χρησιμοποιείται σ' αυτές τις περιπτώσεις ως ο μόνος μετασχηματισμός που μπορεί να επιβάλει μείωση της συχνότητας επικοινωνίας. Είναι δε πειραματικά επιβεβαιωμένο, ότι οι εφαρμογές αυτές δεν επιτυγχάνουν επιτάχυνση αν δεν εφαρμοστεί ο μετασχηματισμός υπερκόμβων πριν από την παραλληλοποίησή τους. Η μέθοδος που παρουσιάστηκε στην παρούσα διατριβή, δίνει τη δυνατότητα ελαχιστοποίησης του άεργου χρόνου των επεξεργαστών και συνακόλουθα συνεισφέρει σημαντικά στη μείωση του συνολικού χρόνου εκτέλεσης, μέσω της αξιοποίησης γενικών μη-ορθογώνιων μετασχηματισμών υπερκόμβων.

Παράρτημα Α

Η Μέθοδος Απαλοιφής Fourier-Motzkin

Η μέθοδος απαλοιφής συστημάτων ανισοτήτων Fourier-Motzkin (Fourier-Motzkin Elimination-FME), είναι μία μέθοδος που χρησιμοποιείται κατά κόρον στην παραγωγή κώδικα από τους μεταγλωττιστές, που πραγματοποιούν βελτιστοποιήσεις [Ban93] [WFW+94] [Wol95] [AKA01]. Η FME εφαρμόζεται για να μετασχηματίσει ένα σύστημα ανισοτήτων που περιγράφει ένα πολύεδρο σε μορφή τέτοια, ώστε οι νέες ανισότητες να μπορούν να χρησιμοποιηθούν, για να προσδιορίσουν τα όρια των μεταβλητών ελέγχου ενός φωλιασμένου βρόχου που θα διασχίσει το πολύεδρο. Αν $A\vec{x} \leq \vec{\alpha}$, $\vec{x} \in Z^n$ είναι ένα σύστημα p ανισοτήτων που περιγράφει ένα πολύεδρο στις n διαστάσεις, τότε μπορούμε να διασχίσουμε αυτό το πολύεδρο με ένα n-διάστατο φωλιασμένο βρόχο, αν για κάθε ανισότητα στην οποία περιέχεται η μεταβλητή x_k, *δεν* περιέχονται οι μεταβλητές $x_{k+1}, \ldots, x_n$. Η μέθοδος FME μπορεί να μετασχηματίσει ένα σύστημα που δεν έχει την παραπάνω ιδιότητα, σε ένα νέο το οποίο την έχει. Χαρακτηριστική περίπτωση εφαρμογής της FME, είναι ο προσδιορισμός των νέων ορίων ενός φωλιασμένου βρόχου, που έχει μετασχηματιστεί από κάποιο μετασχηματισμό T (βλ. Ενότητα 2.6).

Α.1 Υλοποίηση της Μεθόδου

Έστω $A\vec{x} \leq \vec{\alpha}$, $\vec{x} \in Z^n$, ένα σύστημα ανισοτήτων που περιγράφει ένα n-διάστατο πολύεδρο, το οποίο θέλουμε να διασχίσουμε με ένα φωλιασμένο βρόχο. Αν το σύστημα περιέχει p ανισότητες, τότε προφανώς ο πίνακας A είναι $p \times n$. Η διαδικασία απαλοιφής ξεκινά με τον προσδιορισμό των ορίων της μεταβλητής x_n και συνεχίζει με τη μεταβλητή x_{n-1} κ.ο.κ. μέχρι να προσδιοριστούν τα όρια και της x_1. Η μέθοδος FME υλοποιείται ως εξής:

- **Βήμα 1**: Αν x_k είναι η μεταβλητή της οποίας θέλουμε να προσδιορίσουμε τα όρια, τότε μετασχηματίζουμε την k-στή στήλη του πίνακα A (που περιέχει τους πολλαπλασιαστικούς παράγοντες της μεταβλητής x_k), ώστε να περιέχει μόνο παράγοντες $+1, -1$ ή 0. Στο τέλος αυτού του βήματος το σύστημα των ανισοτήτων μπορεί να διαχωριστεί σε τρία συστήματα:

$$A_+\vec{x} \leq \vec{\alpha}_+ \quad A_-\vec{x} \leq \vec{\alpha}_- \quad A_0\vec{x} \leq \vec{\alpha}_0$$

  όπου οι πίνακες A_+, A_- και A_0 περιέχουν $+1, -1$ και 0 στην k-στή στήλη αντίστοιχα.

- **Βήμα 2**: Τα δύο πρώτα υποσυστήματα του προηγούμενου βήματος γράφονται ως εξής:

$$A_+\vec{x} \leq \vec{\alpha}_+ \Rightarrow x_k \leq \vec{\alpha}_+ - (\overline{A}_+\overline{\vec{x}})$$

$$A_-\vec{x} \leq \vec{\alpha}_- \Rightarrow (\overline{A}_-\overline{\vec{x}}) - \vec{\alpha}_- \leq x_k$$

  όπου ο πίνακας $\overline{A}_+$ προκύπτει από τον πίνακα A_+ με αφαίρεση της στήλης k και το διάνυσμα $\overline{\vec{x}}$ προκύπτει από το $\vec{x}$ με αφαίρεση της μεταβλητής x_k. Ανάλογα ορίζονται και οι πίνακες $\overline{A}_-$ και $\overline{A}_0$.

- **Βήμα 3**: Τα όρια του x_k είναι:

$$max((\overline{A}_-\overline{\vec{x}}) - \vec{\alpha}_-) \leq x_k \leq min(\vec{\alpha}_+ - (\overline{A}_+\overline{\vec{x}})) \tag{Α.1}$$

  Επειδή τα όρια που προκύπτουν από τη Σχέση Α.1 μπορεί να είναι μη-ακέραια, ενώ τα όρια των μεταβλητών ελέγχου ενός φωλιασμένου βρόχου πρέπει να είναι ακέραια, χρησιμοποιούμε τον αμέσως μικρότερο ακέραιο στα κάτω όρια και τον αμέσως μεγαλύτερο ακέραιο στα άνω όρια.

- **Βήμα 4**: Από τη Σχέση Α.1 δημιουργούμε το εξής σύστημα ανισοτήτων:

$$(\overline{A}_{-}\overline{\vec{x}}) - \vec{\alpha}_{-} \leq \vec{\alpha}_{+} - (\overline{A}_{+}\overline{\vec{x}}) \tag{Α.2}$$

- **Βήμα 5**: Συνδυάζουμε *κάθε* ανισότητα από το αριστερό μέλος της Σχέσης Α.2 με *κάθε* ανισότητα στο δεξί μέλος της. Με αυτό τον τρόπο προκύπτει ένα νέο σύστημα ανισοτήτων, το οποίο συνδυαζόμενο με το $\overline{A}_0\overline{\vec{x}} \leq \vec{\alpha}_0$ παράγει το σύστημα:

$$\overline{A\vec{x}} \leq \overline{\vec{\alpha}}$$

  Το σύστημα αυτό θα χρησιμοποιηθεί για τον προσδιορισμό των ορίων της μεταβλητής x_{k-1}.

Τα βήματα (1)-(5) επαναλαμβάνονται για $k = n, \ldots, 1$.

Παράδειγμα Α.1 Έστω ένα πολύεδρο στις τρεις διαστάσεις που περιγράφεται από το σύστημα $A\vec{x} \leq \vec{\alpha},\ \vec{x} \in Z^3$, όπου:

$$A = \begin{bmatrix} 0 & -1 & -3 \\ 0 & 0 & -1 \\ -1 & 0 & 6 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \\ 1 & 0 & -6 \end{bmatrix} \text{ και } \vec{\alpha} = \begin{pmatrix} -10 \\ -1 \\ -1 \\ 15 \\ 3 \\ 50 \end{pmatrix}$$

Θα εφαρμόσουμε τη μέθοδο FME στο σύστημα αυτό, προκειμένου να διασχίσουμε το πολύεδρο με ένα τρισδιάστατο φωλιασμένο βρόχο.

$k = 3$:

Σύμφωνα με το Βήμα 1, ο παραπάνω πίνακας, προσαυξημένος με το διάνυσμα $\vec{\alpha}$, μετασχηματίζεται στον πίνακα:

$$\left(\begin{array}{ccc|c} -\frac{1}{6} & 0 & 1 & -\frac{1}{6} \\ 0 & \frac{1}{3} & 1 & 5 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & -1 & -1 \\ 0 & -\frac{1}{3} & -1 & -\frac{10}{3} \\ \frac{1}{6} & 0 & -1 & \frac{50}{6} \end{array}\right)$$

Οι τρεις πρώτες γραμμές του πίνακα αυτού αποτελούν το σύστημα $A_+\vec{x} \leq \vec{\alpha}_+$, ενώ οι τρεις τελευταίες το $A_-\vec{x} \leq \vec{\alpha}_-$. Σ' αυτή την περίπτωση το σύστημα $A_0\vec{x} \leq \vec{\alpha}_0$ είναι κενό. Από τα παραπάνω συστήματα προσδιορίζονται τα όρια της μεταβλητής x_3.

Προσθέτοντας κάθε μία από τις τρεις πρώτες γραμμές του πίνακα, σε κάθε μία από τις τρεις κάτω γραμμές, προκύπτει το σύστημα (αγνοούμε την τρίτη στήλη που περιέχει μόνο μηδενικα):

$$\left(\begin{array}{cc|c} -\frac{1}{6} & 0 & -\frac{7}{6} \\ -\frac{1}{6} & -\frac{1}{3} & -\frac{21}{6} \\ 0 & 0 & \frac{49}{6} \\ 0 & \frac{1}{3} & 4 \\ 0 & 0 & \frac{5}{3} \\ \frac{1}{6} & \frac{1}{3} & \frac{40}{3} \\ 0 & 0 & 2 \\ 0 & -\frac{1}{3} & -\frac{1}{3} \\ \frac{1}{6} & 0 & \frac{68}{6} \end{array}\right)$$

$k = 2$:

Το παραπάνω σύστημα μετασχηματίζεται εύκολα στη μορφή:

$$\left(\begin{array}{cc|c} 0 & 1 & 12 \\ \frac{1}{2} & 1 & 40 \\ -\frac{1}{2} & -1 & -\frac{21}{2} \\ 0 & -1 & -1 \\ -1 & 0 & -7 \\ 0 & 0 & 49 \\ 0 & 0 & 5 \\ 0 & 0 & 2 \\ 1 & 0 & 68 \end{array}\right)$$

Από το σύστημα αυτό προκύπτουν τα άνω και κάτω όρια της μεταβλητής x_2. Ανάλογα για $k = 1$ προκύπτει τελικά:

$$\left(\begin{array}{r|r} 1 & 78 \\ 1 & 68 \\ -1 & 3 \\ -1 & -7 \\ 0 & 5 \\ 0 & 49 \\ 0 & 2 \\ 0 & 59 \\ 0 & 11 \end{array}\right)$$

απ' όπου υπολογίζονται τα όρια της μεταβλητής x_1. Τελικά, το πολύεδρο θα διασχίζεται από ένα βρόχο της μορφής:

```
FOR (x1=max(-3,7); x1 ≤min(78,68); x1++)
   FOR (x2=max(1,⌈(21−x1)/2⌉); x2 ≤min(12,⌊(80−x1)/2⌋); x2++)
      FOR (x3=max(1,⌈(10−x2)/3⌉,⌈(x1−50)/6⌉); x3 ≤min(3,⌊(15−x2)/3⌋,⌊(x1−1)/6⌋); x3++)
          ...
      ENDFOR
   ENDFOR
ENDFOR
```

Α.2 Απλοποιήσεις

Η μέθοδος FME, όπως προκύπτει και από την ανάλυση της παραπάνω ενότητας, παράγει μεγάλο αριθμό ανισοτήτων που χρησιμοποιούνται για την αποτίμηση των ορίων του φωλιασμένου βρόχου. Κάποιες από τις ανισότητες αυτές όμως δεν είναι απαραίτητες για τη σωστή διάσχιση του πολύεδρου. Χαρακτηριστικά στο Παράδειγμα Α.1, η έκφραση $max(-3, 7)$ αποτιμάται πάντα στην τιμή 7, κάτι που σημαίνει ότι η ανισότητα $-x_1 \leq 3$ που προκύπτει από την εφαρμογή της FME, δεν είναι απαραίτητη για τη σωστή αποτίμηση των ορίων. Επειδή οι πλεονάζουσες ανισότητες προσθέτουν εκφράσεις στην αποτίμηση των ορίων, οι οποίες επιβαρύνουν την εκτέλεση του φωλιασμένου βρόχου, είναι γενικά σκόπιμο οι ανισότητες αυτές να αφαιρούνται.

Εξετάζουμε δύο μεθόδους απλοποίησης των συστημάτων που προκύπτουν κατά την απαλοιφή FME. Η πρώτη ονομάζεται ad hoc απλοποίηση, έχει χαμηλή πολυπλοκότητα αλλά δεν

επιτυγχάνει να εντοπίσει όλες τις πλεονάζουσες ανισότητες. Η δεύτερη ονομάζεται ακριβής (exact) απλοποίηση, εντοπίζει όλες τις πλεονάζουσες ανισότητες, αλλά έχει εξαιρετικά μεγάλη πολυπλοκότητα, που καθιστά σε αρκετές περιπτώσεις την εφαρμογή της απαγορευτική. Οι δύο αυτές μέθοδοι μπορούν να εφαρμοστούν διαδοχικά, πρώτα η ad hoc και στη συνέχεια η ακριβής, ώστε να μειωθεί όσο το δυνατόν το κόστος της απλοποίησης. Ο αναγνώστης μπορεί να βρει λεπτομέρειες για τις δύο μεθόδους στην εργασία [BW95]. Στη συνέχεια θα δώσουμε τη γενική ιδέα κάθε μεθόδου.

Η μέθοδος ad hoc ελέγχει διαδοχικά τα όρια των μεταβλητών $x_1, \ldots, x_n$ και ανιχνεύει αυτές τις ανισότητες, που για τα συγκεκριμένα όρια δεν επαληθεύονται ποτέ. Ας υποθέσουμε ότι για τις μεταβλητές x_1 και x_2 ενός φωλιασμένου βρόχου ισχύει $1 \leq x_1 \leq 8$ και $max(5x_1+3, 3x_1+3) \leq x_2 \leq min(5x_1+8, 8x_1-4)$. Για $1 \leq x_1 \leq 8$ ισχύει πάντα $5x_1+3 > 3x_1+3$ επομένως η ανισότητα $3x_1+3 \leq x_2$ είναι πλεονάζουσα. Αντίθετα, οι ανισότητες $x_2 \leq 5x_1+8$ και $x_2 \leq 8x_1-4$, είναι και οι δύο απαραίτητες για την αποτίμηση του άνω ορίου της μεταβλητής x_2. Γενικά η μέθοδος υπολογίζει για κάθε μεταβλητή x_k τα σύνολα $[l_k^{min}, u_k^{max}]$. Αν για κάποια ανισότητα προκύψει ότι για το κάτω όριο της μεταβλητής x_k, έστω l', ισχύει $l' > u_k^{max}$, τότε η ανισότητα αυτή είναι πλεονάζουσα και αφαιρείται. Αντίστοιχα πλεονάζουσα είναι μια ανισότητα, αν προκύψει ότι για το άνω όριο u' της μεταβλητής x_k, ισχύει $u' < l_k^{min}$.

Η ad hoc απλοποίηση παρέχει μία γρήγορη και αποδοτική μέθοδο για την αφαίρεση πλεοναζουσών ανισοτήτων. Το σύστημα μπορεί να απλοποιηθεί περαιτέρω με την εφαρμογή της ακριβούς απλοποίησης. Η ακριβής απλοποίηση στηρίζεται στην παρατήρηση, ότι μία ανισότητα είναι πλεονάζουσα αν, όταν αντιστραφεί, το σύστημα που θα προκύψει από την εφαρμογή της FME, δεν περιέχει κανένα πραγματικό σημείο. Είναι προφανές ότι η ακριβής απλοποίηση είναι εξαιρετικά πολύπλοκη και χρονοβόρα, καθώς απαιτεί την εφαρμογή της μεθόδου FME για κάθε μία ανισότητα που υπάρχει ή δημιουργείται κατά την απαλοιφή FME. Συνήθως το κέρδος στην αποδοτικότητα του παραγόμενου κώδικα είναι πολύ μικρό με την εφαρμογή της ακριβούς απλοποίησης, ενώ αντίθετα ο χρόνος μεταγλώττισης μπορεί να αυξηθεί δραματικά.

Α.3 Πολυπλοκότητα

Από την παρουσίαση της μεθόδου FME είναι μάλλον εμφανές ότι η μέθοδος παρουσιάζει ιδιαίτερα υψηλή πολυπλοκότητα. Συγκεκριμένα, στο Βήμα 5 έχουμε συνδυασμό ανισοτήτων για την παραγωγή νέων και η διαδικασία αυτή επαναλαμβάνεται n φορές. Αν θέλουμε να διασχίσουμε ένα n-διάστατο πολύεδρο που περιγράφεται από p ανισότητες, τότε η πολυπλοκότητα

της απαλοιφής FME έχει ως εξής: Στη χειρότερη περίπτωση, στο Βήμα 4 κατά τον προσδιορισμό των ορίων της μεταβλητής x_n, θα διαχωρίσουμε $p/2$ ανισότητες που περιγράφουν το κάτω όριο της x_n και $p/2$ ανισότητες που περιγράφουν το άνω όριό της. Άρα οι συνδυασμοί των ανισοτήτων αυτών θα δημιουργήσουν $(p/2)^2$ νέες ανισότητες που θα χρησιμοποιηθούν για τον προσδιορισμό των ορίων της μεταβλητής x_{n-1}. Αν και σ' αυτή την περίπτωση οι μισές ανισότητες αφορούν το άνω και οι άλλες μισές το κάτω όριο της x_{n-1}, τότε ο συνδυασμός τους θα δώσει $(p^2/8)^2$ νέες ανισότητες. Έτσι, δεδομένου ότι στη χειρότερη περίπτωση και οι p ανισότητες θα περιέχουν n μεταβλητές, η πολυπλοκότητα της μεθόδου FME θα είναι:

$$C_{FME} = O\left(\frac{p^{2^{n-1}}}{2^{2^n-2}}\right) \approx O\left((\frac{p}{2})^{2^{n-1}}\right)$$

Άρα η πολυπλοκότητα της μεθόδου FME είναι διπλά εκθετική στη διάσταση του πολύεδρου, ή αντίστοιχα στο βάθος του φωλιασμένου βρόχου.

Παράρτημα Β

Κώδικες των Προγραμμάτων

Β.1 Jacobi

Η μέθοδος Jacobi είναι μια παραδοσιακή επαναληπτική μέθοδος στην αριθμητική ανάλυση, που χρησιμοποιείται για την αριθμητική επίλυση προβλημάτων αρχικών/συνοριακών τιμών. Τέτοια προβλήματα θα μπορούσαν να είναι αυτά της διάχυσης θερμότητας ή δυναμικού σε μία δισδιάστατη επιφάνεια. Έτσι, σύμφωνα με τον αλγόριθμο του Jacobi, αν V_{ij}^n είναι το δυναμικό στη θέση με συντεταγμένες ij τη χρονική στιγμή n, τότε θα ισχύει:

$$V_{ij}^{n+1} = \frac{1}{4}(V_{i+1,j}^n + V_{i,j+1}^n + V_{i-1,j}^n + V_{i,j-1}^n + \Delta x^2 q_{ij})$$

Είναι δυνατόν να προσεγγίσουμε επαναληπτικά τις τιμές του δυναμικού σε ένα σημείο με συντεταγμένες (i, j), επαναλαμβάνοντας τον παραπάνω υπολογισμό για μεγάλο αριθμό βημάτων. Ο φωλιασμένος βρόχος που υλοποιεί τον αλγόριθμο Jacobi δίνεται στη συνέχεια. Η μεταβλητή ελέγχου t εκφράζει τα βήματα επανάληψης του αλγορίθμου, ενώ οι i και j τις χωρικές συντεταγμένες. Ο πίνακας A περιέχει τις τιμές του δυναμικού. Θεωρείται ότι ήδη έχουν πραγματοποιηθεί αρχικοποιήσεις που υπαγορεύονται από τις αρχικές/οριακές συνθήκες του προβλήματος.

```
FOR (t=1; t≤T; t++)
   FOR (i=1; i≤I; i++)
      FOR (j=1; j≤J; j++)
         A[t,i,j]=0.25(A[t-1,i-1,j]+A[t-1,i,j-1]+A[t-1,i+1,j]+A[t-1,i,j+1]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του παραπάνω φωλιασμένου βρόχου είναι: $D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$ και ο κώνος υπερκόμβων που προκύπτει από τον πίνακα D είναι $\mathcal{C} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. Επειδή τα διανύσματα εξάρτησης του παραπάνω βρόχου περιέχουν και αρνητικές συντεταγμένες, ο παραπάνω βρόχος δεν μπορεί να μετασχηματιστεί με ορθογώνιο μετασχηματισμό υπερκόμβων (ισχύει $HD \not\geq 0$). Για το λόγο αυτό εφαρμόζουμε στον παραπάνω βρόχο μετασχηματισμό αλλαγής κλίσης που περιγράφεται από τον πίνακα $T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$.

Ο μετασχηματισμένος κώδικας θα είναι:

```
FOR (t'=1; t'≤T; t'++)
   FOR (i'=t'+1; i'≤t'+I; i'++)
      FOR (j'=t'+1; j'≤t'+J; j'++)
         t=t'; i=-t'+i'; j=-t'+j';
         A[t,i,j]=0.25(A[t-1,i-1,j]+A[t-1,i,j-1]+A[t-1,i+1,j]+A[t-1,i,j+1]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο νέος πίνακας εξαρτήσεων θα είναι $D' = TD = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{bmatrix}$. Παρατηρούμε ότι όλες οι συντεταγμένες του νέου πίνακα εξαρτήσεων είναι θετικές. Ο κώνος υπερκόμβων στο μετασχη-

ματισμένο φωλιασμένο βρόχο είναι: $\mathcal{C} = \begin{bmatrix} -3 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$.

B.2 Gauss Successive Over-Relaxation–SOR

Η μέθοδος SOR χρησιμοποιείται σαν βελτίωση της μεθόδου Jacobi για την επαναληπτική επίλυση προβλημάτων αρχικών/συνοριακών συνθηκών. Κάθε επανάληψη της μεθόδου πραγματοποιεί μια πράξη της μορφης:

$$V_{ij}^{n+1} = \frac{w}{4}(V_{i+1,j}^{n} + V_{i,j+1}^{n} + V_{i-1,j}^{n+1} + V_{i,j-1}^{n+1} + \Delta x^2 q_{ij}) + (1-w)V_{ij}^{n}$$

Παρατηρούμε δηλαδή, ότι η μέθοδος SOR χρησιμοποιεί σε κάθε βήμα υπολογισμού τις τιμές που έχουν μόλις υπολογιστεί σ' αυτό το βήμα. Για το λόγο αυτό, η μέθοδος SOR συγκλίνει γρηγορότερα από τη μέθοδο Jacobi. Ο φωλιασμένος βρόχος που υλοποιεί τον αλγόριθμο SOR είναι:

```
FOR (t=1; t≤T; t++)
   FOR (i=1; i≤I; i++)
      FOR (j=1; j≤J; j++)
         A[t,i,j]=w/4(A[t,i-1,j]+A[t,i,j-1]+A[t-1,i+1,j]+A[t-1,i,j+1])+
                  (1-w)A[t-1,i,j];
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του παραπάνω φωλιασμένου βρόχου είναι $D = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}$ και ο αντίστοιχος κώνος υπερκόμβων $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. Και σ' αυτή την περίπτωση απαιτείται μετασχηματισμός αλλαγής κλίσης, για να είναι έγκυρη η εφαρμογή ορθογώνιου μετασχηματισμού υπερκόμβων. Ο μετασχηματισμός αλλαγής κλίσης περιγράφεται από τον πίνακα

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}.$$

Ο φωλιασμένος βρόχος μετά το μετασχηματισμό θα γίνει:

```
FOR (t'=1; t'≤T; t'++)
   FOR (i'=t'+1; i'≤t'+I; i'++)
      FOR (j'=2t'+1; j'≤2t'+J; j'++)
         t=t'; i=-t'+i';j=-2t'+j';
         A[t,i,j]=w/4(A[t,i-1,j]+A[t,i,j-1]+A[t-1,i+1,j]+A[t-1,i,j+1])+
                  (1-w)A[t-1,i,j];
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του νέου φωλιασμένου βρόχου θα είναι $D' = TD = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 2 & 1 & 1 \end{bmatrix}$

και ο αντίστοιχος κώνος υπερκόμβων είναι $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \\ -2 & 1 & 1 \end{bmatrix}$.

B.3 Alternating Direction Implicit Integration–ADI

Η μέθοδος ADI χρησιμοποιείται και αυτή για την επίλυση μερικών διαφορικών εξισώσεων. Ουσιαστικά αποτελείται από δύο βήματα, στο πρώτο διεξάγονται πράξεις κατά μήκος της διάστασης i και στο δεύτερο διεξάγονται πράξεις κατά μήκος της διάστασης j. Αν ενοποιήσουμε τα δύο βήματα αυτά, προκύπτει ο εξής φωλιασμένος βρόχος:

```
FOR (t=1; t≤T; t++)
   FOR (i=1; i≤I; i++)
      FOR (j=1; j≤J; j++)
         X[t,i,j]=X[t-1,i,j]+X[t-1,i,j-1]*A[i,j]/B[t-1,i,j-1]-
                  X[t-1,i-1,j]*A[i,j]/B[t-1,i-1,j];
         B[t,i,j]=B[t-1,i,j]-A[i,j]*A[i,j]/B[t-1,i,j-1]-
                  A[i,j]*A[i,j]/B[t-1,i-1,j];
      ENDFOR
```

```
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του αλγορίθμου αυτού είναι ο $D = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Οι συντεταγμένες όλων των διανυσμάτων εξάρτησης είναι μη αρνητικές, επομένως δεν απαιτείται αλλαγή κλίσης στο συγκεκριμένο παράδειγμα για την εφαρμογή ορθογώνιου μετασχηματισμού υπερκόμβων. Ο κώνος υπερκόμβων στο παράδειγμα αυτό είναι: $\mathcal{C} = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

B.4 Texture Smoothing–TS

Ο αλγόριθμος TS είναι μια βασική μέθοδος στην επεξεργασία εικόνας, που περιλαμβάνει την εύρεση της μέσης φωτεινότητας κάθε σημείου μιας εικόνας, σαν το μέσο όρο της φωτεινότητας των γειτονικών του σημείων. Έτσι, ο φωλιασμένος βρόχος του αλγόριθμου TS είναι:

```
FOR (t=1; t≤T; t++)
   FOR (i=1; i≤I; i++)
      FOR (j=1; j≤J; j++)
         A[t,i,j]=1/9(A[t-1,i,j]+A[t,i-1,j-1]+A[t,i-1,j]+A[t,i-1,j+1]+A[t,i,j-1]+
                  A[t-1,i,j+1]+A[t-1,i+1,j-1]+A[t-1,i+1,j]+A[t-1,i+1,j+1]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του παραπάνω φωλιασμένου βρόχου είναι
$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 & -1 \end{bmatrix}$, ενώ ο κώνος υπερκόμβων είναι $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$.
Ο παραπάνω φωλιασμένος βρόχος μετασχηματίζεται με τη βοήθεια του πίνακα $T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}$.
Ο μετασχηματισμένος κώδικας θα είναι:

```
FOR (t'=1; t'≤T; t'++)
   FOR (i'=t'+1; i'≤t'+I; i'++)
```

```
      FOR (j'=2t'+i'+1; j'≤2t'+i'+J; j'++)
         t=t'; i=i'-t'; j=j'-i'-t';
         A[t,i,j]=1/9(A[t-1,i,j]+A[t,i-1,j-1]+A[t,i-1,j]+A[t,i-1,j+1]+A[t,i,j-1]+
                  A[t-1,i,j+1]+A[t-1,i+1,j-1]+A[t-1,i+1,j]+A[t-1,i+1,j+1]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του μετασχηματισμένου κώδικα θα είναι $D' = TD = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 1 & 1 & 2 & 1 & 0 \end{bmatrix}$ και ο αντίστοιχος κώνος υπερκόμβων $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

B.5 9-point Star Partial Differential Equation Stencil–PDE

Ο τελευταίος κώδικας που μας απασχολεί, προέρχεται και αυτός από την επίλυση μερικών διαφορικών εξισώσεων. Σ' αυτή την επαναληπτική μέθοδο, κάθε σημείο επαναπροσδιορίζεται με βάση τον εαυτό του και τα γειτονικά του σημεία, χρησιμοποιώντας όμως δύο γειτονικά σημεία σε κάθε κατεύθυνση. Έτσι, κάθε επανάληψη της μεθόδου πραγματοποιεί μια πράξη της μορφής:

$$V_{ij}^{n+1} = \frac{1}{9}(V_{i,j}^{n} + V_{i+1,j}^{n} + V_{i+2,j}^{n} + V_{i,j+1}^{n} + V_{i,j+2}^{n} + V_{i-1,j}^{n+1} + V_{i-2,j}^{n+1} + V_{i,j-1}^{n+1} + V_{i,j-2}^{n+1} + \Delta x^2 q_{ij})$$

Ο φωλιασμένος βρόχος του αλγορίθμου PDE είναι:

```
FOR (t=1; t≤T; t++)
   FOR (i=1; i≤I; i++)
      FOR (j=1; j≤J; j++)
         A[t,i,j]=f(A[t-1,i,j]+A[t,i-1,j]+A[t,i-2,j]+A[t-1,i+1,j]+A[t-1,i+2,j]+
                  A[t,i,j-1]+A[t,i,j-2]+A[t-1,i,j+1]+A[t-1,i,j+2]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων του παραπάνω φωλιασμένου βρόχου είναι

$D = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 2 & -1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & -1 & -2 \end{bmatrix}$ και ο κώνος υπερκόμβων είναι $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

Επειδή υπάρχουν αρνητικά στοιχεία στα διανύσματα εξάρτησης, χρησιμοποιούμε τον πίνακα $T = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}$ για να μετασχηματίσουμε τον παραπάνω φωλιασμένο βρόχο.

Ο μετασχηματισμένος κώδικας θα είναι:

```
FOR (t'=1; t'≤T; t'++)
   FOR (i'=2t'+1; i'≤2t'+I; i'++)
      FOR (j'=2t'+i'+1; j'≤2t'+i'+J; j'++)
         t=t'; i=i'-2t'; j=j'-i';
         A[t,i,j]=f(A[t-1,i,j]+A[t,i-1,j]+A[t,i-2,j]+A[t-1,i+1,j]+A[t-1,i+2,j]+
                  A[t,i,j-1]+A[t,i,j-2]+A[t-1,i,j+1]+A[t-1,i,j+2]);
      ENDFOR
   ENDFOR
ENDFOR
```

Ο πίνακας εξαρτήσεων στον μετασχηματισμένο κώδικα θα είναι

$D' = TD = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 2 & -1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & -1 & -2 \end{bmatrix}$ και ο κώνος υπερκόμβων θα είναι $\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Παράρτημα Γ

Μετασχηματισμοί Υπερκόμβων που Χρησιμοποιήθηκαν στα Πειράματα

Γ.1 Δισδιάστατα Προβλήματα

$$P_1 = \begin{bmatrix} 10 & 0 \\ 0 & 15 \end{bmatrix} \; P_2 = \begin{bmatrix} 10 & 0 \\ 4 & 15 \end{bmatrix} \; P_3 = \begin{bmatrix} 10 & 3 \\ 4 & 15 \end{bmatrix}$$

Γ.2 Τρισδιάστατα Προβλήματα

$$P_4 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \; P_5 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 5 & 4 \end{bmatrix} \; P_6 = \begin{bmatrix} 5 & 0 & 2 \\ 0 & 3 & 0 \\ 0 & 5 & 4 \end{bmatrix} \; P_7 = \begin{bmatrix} 5 & 0 & 2 \\ 0 & 3 & 0 \\ 4 & 5 & 4 \end{bmatrix} \; P_8 = \begin{bmatrix} 5 & 0 & 2 \\ 4 & 3 & 0 \\ 4 & 6 & 4 \end{bmatrix}$$

$$P_9 = \begin{bmatrix} 5 & 0 & 2 \\ 4 & 3 & 1 \\ 4 & 6 & 4 \end{bmatrix} \; P_{10} = \begin{bmatrix} 5 & 6 & 2 \\ 9 & 3 & 4 \\ 5 & 6 & 4 \end{bmatrix}$$

Παράρτημα Δ

Πίνακας Συμβόλων

Σύμβολο	**Επεξήγηση**	Σελ.
Z	σύνολο ακεραίων	15
R	σύνολο ρητών	16
n	αριθμός των βρόχων σε ένα φώλιασμα, ή βάθος του φωλιασμένου βρόχου	16
J	αρχικός χώρος επαναλήψεων ή χώρος δεικτών	17
l_k	κάτω όριο διάστασης k στον J	16
u_k	άνω όριο διάστασης k στον J	16
D	πίνακας εξαρτήσεων	21
$\vec{d}_k$	διάνυσμα εξάρτησης (η k-στή στήλη του πίνακα D)	21
q	αριθμός των εξαρτήσεων	20
H	πίνακας μετασχηματισμού υπερκόμβων	27
P	αντίστροφος πίνακας μετασχηματισμού υπερκόμβων ($P = H^{-1}$)	27
TIS	Χώρος Επαναλήψεων Υπερκόμβου (Tile Iteration Space)	28
J^S	Χώρος Υπερκόμβων (Tile Space)	28
TOS	Χώρος Αρχών Υπερκόμβων (Tile Origin Space)	28
D^S	Πίνακας Εξαρτήσεων Υπερκόμβων	28
f_w	συνάρτηση εγγραφής σε ένα πίνακα	44
DS	Χώρος Δεδομένων	45
l_k^S	κάτω όριο διάστασης k στον J^S	53
u_k^S	άνω όριο διάστασης k στον J^S	53

Σύμβολο	Επεξήγηση	Σελ.
l'_k	κάτω όριο διάστασης k στον $TTIS$	60
u'_k	άνω όριο διάστασης k στον $TTIS$	60
$TTIS$	Μετασχηματισμένος Χώρος Επαναλήψεων Υπερκόμβου (Transformed Tile Iteration Space)	60
H'	πίνακας μετασχηματισμού από τον TIS στον $TTIS$	61
P'	πίνακας μετασχηματισμού από τον $TTIS$ στον TIS ($P' = H'^{-1}$)	61
V	διαγώνιος πίνακας για τον οποίο ισχύει $H' = VH$	61
c_k	βήματα στον $TTIS$ ($= \widetilde{h'}_{kk}$)	63
$\widetilde{H'}$	Ερμητιανή Κανονική Μορφή του H'	63
a_{kl}	αρχικές τιμές στον $TTIS$ ($= \widetilde{h'}_{kl}$)	63
$p\vec{id}$	$n-1$-διάστατο διάνυσμα αναγνωριστικό των επεξεργαστών	75
m	διεύθυνση απεικόνισης	77
j^S_m	συντεταγμένη στη διεύθυνση απεικόνισης του Χώρου Υπερκόμβων	77
t^S	συντεταγμένη στη διεύθυνση απεικόνισης του Χώρου Υπερκόμβων (ισοδύναμη με $j^S_m - l^S_m$)	77
LDS	Τοπικός Χώρος Δεδομένων (Local Data Space)	79
map	συνάρτηση απεικόνισης από τον LDS στον $TTIS$	80
map^{-1}	συνάρτηση απεικόνισης από τον $TTIS$ στον LDS	81
loc	συνάρτηση απεικόνισης από τον LDS στον J	83
loc^{-1}	συνάρτηση απεικόνισης από τον J στον LDS	84
D'	πίνακας εξαρτήσεων στον $TTIS$ ($D' = H'D$)	87
$C\vec{C}$	διάνυσμα επικοινωνίας	88
D^m	προκύπτει από τον D^S με αφαίρεση της m-οστής γραμμής	90
$\vec{d^m}$	διάνυσμα-στήλη του πίνακα D^m	90
$d^S(d^m)$	όλες οι εξαρτήσεις μεταξύ υπερκόμβων d^S που δημιουργούν την εξάρτηση επεξεργαστών d^m	91
$d^m(\vec{d^S})$	η εξάρτηση επεξεργαστών d^m που αντιστοιχεί στην εξάρτηση υπερκόμβων d^S	91

Βιβλιογραφία

[ACN+00] R. Andonov, P. Calland, S. Niar, S. Rajopadhye, and N. Yanev. First Steps Towards Optimal Oblique Tile Sizing. In *Proc. of the 8th International Workshop on Compilers for Parallel Computers*, pages 351–366, Aussois, France, Jan. 2000.

[AI91] C. Ancourt and F. Irigoin. Scanning Polyhedra with DO Loops. In *Proc. of the Third ACM SIGPLAN Symposium on Principles & Practice of Programming Languages*, pages 39–50, Williamsburg, Virginia, US, Apr. 1991.

[AKA01] R. Allen, K. Kennedy, and J. R. Allen. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach.* Morgan Kaufmann Publishers, San Francisco, California, US, 2001.

[AKN95] A. Agarwal, D. Kranz, and V. Natarajan. Automatic Partitioning of Parallel Loops and Data Arrays for Distributed Shared-Memory Multiprocessors. *IEEE Trans. on Parallel and Distributed Systems*, 6(9):943–962, 1995.

[AKPT99] T. Andronikos, N. Koziris, G. Papakonstantinou, and P. Tsanakas. Optimal Scheduling for UET/UET-UCT Generalized N-Dimensional Grid Task Graphs. *Journal of Parallel and Distributed Computing*, 57(2):140–165, May 1999.

[AL93] S. P. Amarasinghe and M. S. Lam. Communication Optimization and Code Generation for Distributed Memory Machines. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'93)*, Albuquerque, New Mexico, US, Jun. 1993.

[AMC97] V. Adve and J. Mellor-Crummey. Advanced Code Generation for High Performance Fortran. In *Languages, Compilation Techniques and Run Time Systems for Scalable Parallel Systems*, chapter 18, Lecture Notes in Computer Science Series. Springer-Verlag, 1997.

[ASTK02] M. Athanasaki, A. Sotiropoulos, G. Tsoukalas, and N. Koziris. Pipelined Scheduling of Tiled Nested Loops onto Clusters of SMPs using Memory Mapped Network Interfaces. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing (SC2002)*, Baltimore, Maryland, Nov 2002.

[Ban88] U. Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, 1988.

[Ban93] U. Banerjee. *Loop Transformations for Restructuring Compilers.* Kluwer Academic Publishers, 1993.

[BDRR94] P. Boulet, A. Darte, T. Risset, and Y. Robert. (Pen)-ultimate Tiling? *INTEGRATION, The VLSI Jounal*, 17:33–51, 1994.

[BDRV99] P. Boulet, J. Dongarra, Y. Robert, and F. Vivien. Static Tiling for Heterogeneous Computing Platforms. *Parallel Computing*, 25(5):547–568, 1999.

[Ber66] A. Bernstein. Analysis of Programs for Parallel Programming. *IEEE Trans. on Computers*, 15(5):757–763, Oct 1966.

[BW95] A. Bik and H. Wijshoff. Implementation of Fourier-Motzkin Elimination. In *Proc. of the First Annual Conference of the ASCI*, pages 377–386, The Netherlands, 1995.

[CDK+01] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon. *Parallel Programming in OpenMP.* Morgan Kaufmann Publishers, San Francisco, California, US, 2001.

[CDR97] P.-Y. Calland, J. Dongarra, and Y. Robert. Tiling with Limited Resources. Technical Report CS-97-350, University of Tennessee, Feb. 1997.

[CDRV98] P.-Y. Calland, A. Darte, Y. Robert, and F. Vivien. On the Removal of Anti and Output Dependences. *International Journal of Parallel Programming*, 26(2):285–312, 1998.

[CL95] M. Cierniak and W. Li. Unifying Data and Control Transformations for Distributed Shared Memory Machines. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'95)*, pages 205–217, La Jolla, California, US, Jun. 1995.

[CM99] J. Chame and S. Moon. A Tile Selection Algorithm for Data Locality and Cache Interference. In *Proc. of the International Conference on Supercomputing (ICS'99)*, pages 492–499, Rhodes, Greece, 1999.

[CMT94] S. Carr, K. McKinley, and C.-W. Tseng. Compiler optimizations for improving data locality. In *Proc. of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'94)*, pages 252–262, San Jose, California, US, Oct. 1994.

[CMZ92] B. Chapman, P. Mehrotra, and H. Zima. Programming in Vienna Fortran. In *Proc. of the Third Workshop on Compilers for Parallel Computers*, pages 121–160, Jul. 1992.

[CR95] P.-Y Calland and T. Risset. Precise Tiling for Uniform Loop Nests. In *Proc. of the International Conference on Application Specific Array Processors (ASAP'95)*, pages 330–337, Strasbourg, France, Jul. 1995.

[DDR97] F. Desprez, J. Dongarra, and Y. Robert. Determining the Idle Time of a Tiling: New Results. *Journal of Information Science and Engineering*, 14:167–190, Mar. 1997.

[DGAK03] N. Drosinos, G. Goumas, M. Athanasaki, and N. Koziris. Delivering High Performance to Parallel Applications Using Advanced Scheduling. In *Proc. of the Parallel Computing 2003 (ParCo 2003)*, Dresden, Germany, Sep. 2003.

[D'H92] E. D'Hollander. Partitioning and Labeling of Loops by Unimodular Transformations. *IEEE Trans. on Parallel and Distributed Systems*, 3(4):465–476, Jul. 1992.

[DK03] N. Drosinos and N. Koziris. Advanced Hybrid MPI/OpenMP Parallelization Paradigms for Nested Loop Algorithms onto Clusters of SMPs. In *Proceedings of the 10th EuroPVM/MPI Conference 2003 (EuroPVM/MPI 2003)*, pages 203–213, Venice, Italy, Sep. 2003.

[FHK+91] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C. Tseng, and M. Wu. Fortran-D Language Specification. Technical Report TR-91-170, Dept. of Computer Science, Rice University, Dec. 1991.

[FLV95] A. Fernandez, J. Llaberia, and M. Valero. Loop Transformations Using Nonunimodular Matrices. *IEEE Trans. on Parallel and Distributed Systems*, 6(8):832–840, Aug. 1995.

[GAK02a] G. Goumas, M. Athanasaki, and N. Koziris. Automatic Code Generation for Executing Tiled Nested Loops Onto Parallel Architectures. In *Proc. of the ACM Symposium on Applied Computing (SAC'02)*, pages 876–881, Madrid, Spain, Mar. 2002.

[GAK02b] G. Goumas, M. Athanasaki, and N. Koziris. Code Generation Methods for Tiling Transformations. *JISE: Journal of Information Science and Engineering*, 18:667–691, 2002.

[GAK03] G. Goumas, M. Athanasaki, and N. Koziris. An Efficient Code Generation Technique for Tiled Iteration Spaces. *IEEE Trans. on Parallel and Distributed Systems*, 14(10):1021–1034, Oct. 2003.

[GB92] M. Gupta and P. Banerjee. Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):179–193, Mar. 1992.

[GDAK02a] G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. Compiling Tiled Iteration Spaces for Clusters. In *Proc. of the IEEE International Conference on Cluster Computing*, pages 360–369, Chicago, Illinois, US, Sep. 2002.

[GDAK02b] G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. Data parallel code generation for arbitrarily tiled loop nests. In *Proc. of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pages 610–616, Las Vegas, Nevada, US, Jun. 2002.

[GMS+95] M. Gupta, S. Midkiff, E. Schonberg, V. Seshadri, D. Shields, K. Wang, W. Ching, and T. Ngo. An HPF Compiler for the IBM SP2. In *Proc. of Supercomputing '95*, San Diego, California, US, Dec. 1995.

[GSK01] G. Goumas, A. Sotiropoulos, and N. Koziris. Minimizing Completion Time for Loop Tiling with Computation and Communication Overlapping. In *Proc. of the IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS'01)*, San Francisco, California, US, Apr. 2001.

[GSS96] M. Gupta, E. Schonberg, and H. Srinivasan. A Unified Framework for Optimizing Communication in Data-Parallel Programs. *IEEE Transactions on Parallel and Distributed Systems*, 7(7):689–704, 1996.

[HCF97] K Hogstedt, L. Carter, and J. Ferrante. Determining the Idle Time of a Tiling. In *Proc. of the Principles of Programming Languages (POPL'97)*, pages 319–323, Paris, France, Jan. 1997.

[HCF99] K. Hogstedt, L. Carter, and J. Ferrante. Selecting Tile Shape for Minimal Execution Time. In *Proc. of the ACM Symposium on Parallel Algorithms and Architectures*, pages 201–211, Saint Malo, France, Jun. 1999.

[HCF03] K Hogstedt, L. Carter, and J. Ferrante. On the Parallel Execution Time of Tiled Loops. *IEEE Trans. on Parallel and Distributed Systems*, 14(3):307–321, Mar. 2003.

[HP95] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, San Francisco, California, US, 1995.

[HP96] M. R. Haghighat and C. D. Polychronopoulos. Symbolic Analysis for Parallelizing Compilers. *ACM Transactions on Programming Languages and Systems*, 18(4):477–518, Jul. 1996.

[HS98] E. Hodzic and W. Shang. On Supernode Transformation with Minimized Total Running Time. *IEEE Trans. on Parallel and Distributed Systems*, 9(5):417–428, May 1998.

[HS02] E. Hodzic and W. Shang. On Time Optimal Supernode Shape. *IEEE Trans. on Parallel and Distributed Systems*, 13(12):1220–1233, Dec 2002.

[IT88] F. Irigoin and R. Triolet. Supernode Partitioning. In *Proc. of the 15th Ann. ACM SIGACT-SIGPLAN Symp. Principles of Programming Languages (POPL'85)*, pages 319–329, San Diego, California, US, Jan. 1988.

[Jim99] M. Jimenez. *Multilevel Tiling for Non-Rectangular Iteration Spaces*. PhD thesis, Universitat Politecnica de Catalunia, 1999.

[KBC+99] M. Kandemir, P. Banerjee, A. Choudary, J. Ramanujam, and N. Shenoy. A Global Communication Optimization Technique Based on Data-flow Analysis and Linear Algebra. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 21(6):1251–1297, Nov. 1999.

[KCRB99] M. Kandemir, A. Choudhary, J. Ramanujam, and P. Banerjee. On reducing false sharing while improving locality on shared memory multiprocessors. In *Proc. of the International Conference on Parallel Architectures and Compilation Techniques (PACT'99)*, pages 203–211, Newport Beach, California, US, Oct. 1999.

[KCS+98] M. Kandemir, A. Choudhary, N. Shenoy, P. Banerjee, and J. Ramanujam. A Hyperplane Based Approach for Optimizing Spatial Locality in Loop Nests. In *Proc. of the International Conference on Supercomputing (ICS'98)*, pages 69–76, Melbourne, Australia, Jul. 1998.

[KM92] K. Kennedy and K. McKinley. Optimizing for Parallelism and Data Locality. In *Proc. of the International Conference on Supercomputing (ICS'92)*, pages 323–334, Washington, D. C., US, Jul. 1992.

[KM93] K. Kennedy and K. McKinley. Maximizing Loop Parallelism and Improving Data Locality via Loop Fusion and Distribution. In *Proc. of the 1993 Workshop on Languages and Compilers for Parallel Computing*, pages 301–320, Portland, Oregon, US, Aug. 1993.

[KMP+95] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott. The Omega Library Interface Guide. Technical Report CS-TR-3445, CS Dept., Univ. of Maryland, College Park, Mar 1995.

[KPCM99] I. Kodukula, K. Pingali, R. Cox, and D. Maydan. An Experimental Evaluation of Tiling and Shackling for Memory Hierarchy Management. In *Proc. of the International Conference on Supercomputing (ICS'99)*, pages 482–491, Rhodes, Greece, 1999.

[KRC97] M. Kandemir, J. Ramanujam, and A. Choudhary. A Compiler Algorithm for Optimizing Locality in Loop Nests. In *Proc. of the International Conference on Supercomputing (ICS'97)*, pages 269–276, Vienna, Austria, Jul. 1997.

[KRC99] M. Kandemir, J. Ramanujam, and A. Choudary. Improving Cache Locality by a Combination of Loop and Data Transformations. *IEEE Trans. on Parallel and Distributed Systems*, 48(2):159–167, Feb. 1999.

[KSG03] N. Koziris, A. Sotiropoulos, and G. Goumas. A Pipelined Schedule to Minimize Completion Time for Loop Tiling with Computation and Communication Overlapping. *Journal of Parallel and Distributed Computing*, 63(11):1138–1151, Nov. 2003.

[Li93] W. Li. *Compiling for NUMA Parallel Machines.* PhD thesis, Cornell Univ. Ithaca, New York, 1993.

[Li94] W. Li. Compiler Optimizations for Cache Locality and Coherence. Technical Report TR504, Cornell Univ. Ithaca, New York, 1994.

[LLL01] A. Lim, S. Liao, and M. Lam. Blocking and Array Contraction Across Arbitrarily Nested Loops Using Affine Partitioning. *ACM SIGPLAN Notices*, 36(7):103–112, 2001.

[LP92] W. Li and K. Pingali. Compiler Optimizations for Cache Locality and Coherence. Technical Report TR 92-1278, Cornell Univ. Ithaca, New York, 1992.

[LP94] W. Li and K. Pingali. A Singular Loop Transformation Framework Based on Non-singular Matrices. *International Journal of Parallel Programming*, 22(2):183–205, 1994.

[LRW91] M. Lam, E. Rothberg, and M. Wolf. The Cache Performance and Optimizations of Blocked algorithms. In *Proc. of the Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 63–74, Santa Clara, California, US, Apr. 1991.

[MCT96] K. McKinley, S. Carr, and C.-W. Tseng. Improving data locality with loop transformations. *ACM Transactions on Programming Languages and Systems*, 18(4):424–453, Jul. 1996.

[ML92] E. P. Markatos and T. J. LeBlanc. Using processor affinity in loop scheduling on shared-memory multiprocessors. *IEEE Trans. on Parallel and Distributed Systems*, 5(4):379–400, Apr. 1992.

[MPI94] Message Passing Interface Forum MPIF. MPI: A message-passing interface standard. Technical Report UT-CS-94-230, 1994.

[MPI96] Message Passing Interface Forum MPIF. MPI-2: Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville, 1996.

[OSKO95] H. Ohta, Y. Saito, M. Kainaga, and H. Ono. Proc. of the 9th international conference on supercomputing (ics'95. In *Optimal Tile Size Adjustment in Compiling General DOACROSS Loop Nests*, pages 270–279, 1995.

[Pug92] W. Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. *Communications of the ACM*, 35(8):102–114, Aug. 1992.

[Ram92] J. Ramanujam. Non-Unimodular Loop Transformations of Nested Loops. In *Proc. of the Supercomputing 92*, pages 214–223, Minneapolis, Minnesota, US, Nov. 1992.

[Ram95] J. Ramanujam. Beyond Unimodular Transformations. *Journal of Supercomputing*, 9(4):365–389, Oct. 1995.

[RS91] J. Ramanujam and P. Sadayappan. Compile-Time Techniques for Data Distribution in Distributed Memory Machines. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):472–482, 1991.

[RS92] J. Ramanujam and P. Sadayappan. Tiling Multidimensional Iteration Spaces for Multicomputers. *Journal of Parallel and Distributed Computing*, 16:108–120, 1992.

[SF92] W. Shang and J.A.B. Fortes. Independent Partitioning of Algorithms with Uniform Dependencies. *IEEE Trans. on Computers*, 41(2):190–206, Feb. 1992.

[SLR+95] E. Su, A. Lain, S. Ramaswamy, D. J. Palermo, E. W. Hodges, and P. Banerjee. Advanced Compilation Techniques in the PARADIGM Compiler for Distributed Memory Multicomputers. In *Proc. of the International Conference on Supercomputing (ICS'95)*, Madrid, Spain, Jul. 1995.

[SM97] S. Singhai and K. McKinley. A Parameterized Loop Fusion Algorithm for Improving Parallelism and Cache Locality. *The Computer Journal*, 40(6):340–355, 1997.

[STK01] A. Sotiropoulos, G. Tsoukalas, and N. Koziris. A Pipelined Execution of Tiled Nested Loops onto a Cluster of PCs using PCI-SCI NICs. In *Proc. of the 2001 SCI-Europe Conference*, Dublin, Ireland, Oct. 2001.

[STK02] A. Sotiropoulos, G. Tsoukalas, and N. Koziris. Enhancing the Performance of Tiled Loop Execution onto Clusters using Memory Mapped Network Interfaces and Pipelined Schedules. In *Proc. of the 2002 Workshop on Communication Architecture for Clusters (CAC'02), Int'l Parallel and Distributed Processing Symposium (IPDPS'02)*, Fort Lauderdale, Florida, US, Apr. 2002.

[TLH94] J. Torrellas, M. Lam, and J. Hennessy. False Sharing and Spatial Locality in Multiprocessor Caches. *IEEE Transactions on Computers*, 43(6):651–663, 1994.

[TN93] T.H. Tzen and L.M. Ni. Trapezoid Self-Scheduling: A Practical Scheduling Scheme for Parallel Compilers. *IEEE Trans. on Parallel and Distributed Systems*, 4(1):87–98, Jan. 1993.

[TX00] P. Tang and J. Xue. Generating Efficient Tiled Code for Distributed Memory Machines. *Parallel Computing*, 26(11):1369–1410, 2000.

[TZ94] P. Tang and J. Zigman. Reducing Data Communication Overhead for DOACROSS Loop Nests. In *Proc. of the International Conference on Supercomputing (ICS'94)*, pages 44–53, Manchester, UK, Jul. 1994.

[WFW+94] Robert P. Wilson, Robert S. French, Christopher S. Wilson, Saman P. Amarasinghe, Jennifer-Ann M. Anderson, Steven W. K. Tjiang, Shih-Wei Liao, Chau-Wen Tseng, Mary W. Hall, Monica S. Lam, and John L. Hennessy. SUIF: An Infrastructure for Research on Parallelizing and Optimizing Compilers. *SIGPLAN Notices*, 29(12):31–37, 1994.

[WL91a] M. Wolf and M. Lam. A Data Locality Optimizing Algorithm. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'91)*, Toronto, Ontario, Canada, Jun. 1991.

[WL91b] M. Wolf and M. Lam. A Loop Transformation Theory and an Algorithm to Maximize Parallelism. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):452–471, Oct. 1991.

[Wol89] M. Wolfe. More Iteration Space Tiling. In *Proc. of the Supercomputing 89*, pages 655–664, Reno, Nevada, US, Nov. 1989.

[Wol95] M.J. Wolfe. *High-Performance Compilers for Parallel Computing.* Addison-Wesley, Reading, Massachusetts, USA, 1995.

[Xue94] J. Xue. Automatic Non-unimodular Loop Transformations for Massive Parallelism. *Parallel Computing*, 20(5):711–728, 1994.

[Xue97a] J. Xue. Communication-Minimal Tiling of Uniform Dependence Loops. *Journal of Parallel and Distributed Computing*, 42(1):42–59, 1997.

[Xue97b] J. Xue. On Tiling as a Loop Transformation. *Parallel Processing Letters*, 7(4):409–424, 1997.

[XW02] J. Xue and W.Cai. Time-minimal Tiling when Rise is Larger than Zero. *Parallel Computing*, 28(6):915–939, 2002.