Αλγόριθμος προσομοίωσης κβαντικών κυκλωμάτων με εναλλακτικές μορφές BDDs για την αναπαράσταση των καταστάσεων διανυσμάτων

Κωνσταντίνος Κουράτορας

Εξεταστική επιτροπή:
Επίκουρος Καθηγητής Βασίλης Σαμολαδάς (επιβλέπων)
Καθηγητής Απόστολος Δόλλας
Αναπληρωτής Καθηγητής Δημοσθένης Έλληνας

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών
Πολυτεχνείο Κρήτης

Χανιά, Αύγουστος 2008

Περίληψη

Η μόνη μέθοδος μελέτης κβαντικών αλγορίθμων είναι μέσω της προσομοίωσης κβαντικών υπολογιστών, καθώς δεν έχει αναπτυχθεί ακόμα υλικό στον τομέα αυτόν. Όμως, επειδή αυτού του είδους η προσομοίωση έχει μεγάλες απαιτήσεις σε χρόνο εκτέλεσης και αποθηκευτικό χώρο, είναι σημαντικό να αναπτυχθούν αποτελεσματικοί αλγόριθμοι για την αναπαράσταση κβαντικής πληροφορίας στους κλασικούς υπολογιστές. Στην παρούσα εργασία θα γίνει επέκταση του αλγορίθμου [1], που προσομοιώνει κβαντικά κυκλώματα κάνοντας χρήση των Multi-Terminal Binary Decision Diagrams για την αναπαράσταση των καταστάσεων με συμβολική μορφή και όχι πλέον ως πίνακες.

Αναζητώντας αποδοτικότερες δομές δεδομένων για την αναπαράσταση της πληροφορίας αυτής, θα χρησιμοποιηθούν εναλλακτικές δομές, βασισμένες στα Binary Decision Diagrams. Εφόσον ο ήδη υπάρχων αλγόριθμος μας έχει απαλλάξει από την ανάγκη χρήσης πινάκων, μπορούμε να επικεντρωθούμε στην εύρεση και μελέτη άλλων μορφών αναπαράστασης της κβαντικής πληροφορίας. Έτσι, με τη χρήση εναλλακτικών μορφών BDDs στοχεύουμε στη βελτίωση της απόδοσης του αλγορίθμου προσομοίωσης. Τα αποτελέσματα της εργασίας έχουν μεγάλη θεωρητική και πρακτική αξία, καθώς μια οποιαδήποτε βελτίωση της απόδοσης του αλγορίθμου, θα μας βοηθήσει αρκετά στη μελέτη των κβαντικών υπολογιστών.

Αφιερωμένο στους Κώστα, Μαρία, Νίκο, Γιάννη

Ευχαριστίες

Πριν από όλους, θα ήθελα να ευχαριστήσω τους γονείς μου, Κώστα και Μαρία, για την ψυχολογική και οικονομική υποστηρίξή τους σε όλη τη διάρκεια της ζωής μου, χωρίς την οποία δε θα ήταν εφικτό να υλοποιήσω τα όνειρά μου και να σπουδάσω στο αντικείμενο της επιστήμης των υπολογιστών.

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Βασίλη Σαμολαδά για την επίβλεψη και την καθοδήγησή του στην εργασία αυτή, για την προθυμία του να με βοηθήσει οποτεδήποτε είχα κάποιο πρόβλημα και γενικότερα για τη συνεργασία μας, η οποία υπήρξε άψογη. Επίσης ευχαριστώ προκαταβολικά τους κ. Απόστολο Δόλλα και κ. Δημοσθένη Έλληνα για την ανάγνωση του κειμένου και τις τυχόν παρατηρήσεις τους.

Τέλος, θα ήθελα να ευχαριστήσω τον Βλάση, τον Κώστα και τον Νίκο για τις δημιουργικές (και μη) ώρες που περάσαμε μαζί, για την πολύτιμη ανταλλαγή γνώσεων και απόψεων και για την υποστήριξή τους σε όλες τις δύσκολες στιγμές της φοιτητικής μου ζωής.

Αύγουστος 2008,

Κωνσταντίνος Κουράτορας

Περιεχόμενα

Κεφάλαιο

Ευρετήριο αλγορίθμων

Ευρετήριο σχημάτων

Σχήμα **Σελίδα**

Κεφάλαιο 1

Εισαγωγή

Οι κβαντικοί υπολογιστές αποτελούν εκ των πραγμάτων δύσκολο πεδίο έρευνας, καθώς η συγκεκριμένη επιστήμη βρίσκεται ακόμα σε πειραματικό στάδιο και δεν έχει αναπτυχθεί υλικό που να μας επιτρέπει την πρακτική εφαρμογή κβαντικών αλγορίθμων. Κατα συνέπεια, από τη στιγμή που η μελέτη των κβαντικών υπολογιστών μπορεί να βοηθηθεί σε μεγάλο βαθμό από τη δυνατότητα να εκτελεστούν κβαντικοί αλγόριθμοι, η μοναδική λύση είναι η προσομοίωση κάνοντας χρήση κλασικών υπολογιστών.

Η προσομοίωση τέτοιου είδους αλγορίθμων έχει μεγάλες απαιτήσεις σε επεξεργαστική ισχύ, σε χρόνο εκτέλεσης και σε αποθηκευτικό χώρο και είναι δύσκολο να υλοποιηθεί με τις τεχνικές που είναι γνωστές μέχρι σήμερα. Οι περισσότεροι αλγόριθμοι προσομοίωσης που αναφέρονται στη βιβλιογραφία αναπαριστούν τις κβαντικές καταστάσεις με τη λεγόμενη αναπαράσταση διανύσματος κατάστασης (state-vector). Για παράδειγμα κάνουν χρήση ενός μιγαδικού διανύσματος 2^n στοιχείων για την αποθήκευση της κατάστασης μιας μνήμης n-qubit. Με το υπάρχον υλικό, αυτή η προσέγγιση περιορίζεται περίπου σε $n \approx 30$.

Για να γίνει εφικτό να ξεπεραστεί το όριο αυτό, υπάρχουν δύο επιλογές. Είτε να εγκαταλειφθεί η αναπαράσταση του διανύσματος κατάστασης, είτε να χρησιμοποιηθούν κατάλληλες τεχνικές συμπίεσης για τη μείωση του

χώρου που απαιτείται για την αναπαράσταση του διανύσματος κατάστασης.

Η πρώτη επιλογή είναι ιδιαίτερα ενδιαφέρουσα αλλά και περίεργη και στην πραγματικότητα υπάρχουν τουλάχιστον δύο διαφορετικές προσεγγίσεις που ακολουθούν αυτή την πορεία. Δυστυχώς, αυτές ισχύουν μόνο σε περιορισμένες περιπτώσεις. Σύμφωνα με τις μέχρι σήμερα γνώσεις μας στο αντικείμενο, κάθε γενική τεχνική για κβαντική προσομοίωση βασίζεται στην αναπαράσταση διανύσματος κατάστασης. Η συμπίεση της κατάστασης αναπαράστασης για υπολογισμούς στους κλασσικούς υπολογιστές έχει μελετηθεί εκτενώς κατά τα τελευταία 20 χρόνια.

Μια μεγάλη καινοτομία ήταν η εισαγωγή στα Reduced Ordered Binary Decision Diagrams (ROBDDs) [10]. Τα ROBDDs μπορούν να συμπιέσουν μια μεγάλη δομή μειώνοντας της το μέγεθός της σε τέτοια επίπεδα ώστε να γίνει εφικτή η επεξεργασία και η μελέτη της. Έχει αποδειχτεί [19, 17, 18, 13] ότι μια επέκταση των ROBDDs, τα Multi-Terminal BDDs (MTBDDs) [11], μπορούν να χρησιμοποιηθούν αποδοτικά στη συμπίεση του διανύσματος κατάστασης ενός κβαντικού συστήματος. Η αναπαράσταση δε και των κβαντικών τελεστών ως MTBDDs, μας επιτρέπει να προσομοιώσουμε μεγάλα κβαντικά κυκλώματα.

Ακόμα και με τον τρόπο αυτόν όμως, στις περισσότερες περιπτώσεις το μέγεθος της συμπιεσμένης μορφής του διανύσματος κατάστασης αυξάνεται εκθετικά, όσο αυξάνεται και το μέγεθος του κυκλώματος που προσομοιώνεται. Κατά συνέπεια, μια μέθοδος κβαντικής προσομοίωσης που βασίζεται στα MTBDDs εξακολουθεί να είναι αρκετά απαιτητική σε πόρους συστήματος κλασικών υπολογιστών και είναι ζωτικής σημασίας η περαιτέρω βελτίωση της απόδοσης των αλγορίθμων που χρησιμοποιούμε.

Επίσης έχει προταθεί μια προσέγγιση [1] για προσομοίωση αλγορίθμων κβαντικών κυκλωμάτων που βασίζεται στα MTBDD και ξεχωρίζει στο γεγονός ότι χειρίζεται συμβολικές αναπαραστάσεις των κβαντικών τελεστών, σε

αντίθεση με τις προηγούμενες προσεγγίσεις που βασίζονταν σε πίνακες (είτε συμπιεσμένης μορφής είτε όχι). Η τεχνική αυτή παρουσιάζει σημαντικά πλεονεκτήματα στις επιδόσεις σε συγκριτικά με προηγούμενες δουλειές καθώς η πειραματική αξιολόγησή της δείχνει ότι υπερέχει των προηγούμενων περίπου δύο τάξεις μεγέθους και αποδεικνύεται ότι οι αλγόριθμοι είναι ταχύτεροι.

Στην παρούσα εργασία γίνεται επέκταση της τεχνικής των συμβολικών αναπαραστάσεων [1]. Εφόσον ο ήδη υπάρχων αλγόριθμος μας έχει απαλλάξει από την ανάγκη για χρήση πινάκων για την αναπαράσταση της κβαντικής πληροφορίας, μπορούμε να επικεντρωθούμε στην εύρεση και μελέτη άλλων μορφών αναπαράστασης της πληροφορίας αυτής. Έτσι, κάνοντας χρήση εναλλακτικών μορφών BDDs, στοχεύουμε στη βελτίωση της απόδοσης του αλγορίθμου προσομοίωσης. Οι εναλλακτικές μορφές που χρησιμοποιούνται είναι τα Zero-Suppressed BDDs και One-Suppressed BDDs, καθώς και συνδυασμός αυτών.

Τα αποτελέσματα της παρούσας εργασίας έχουν μεγάλη θεωρητική, αλλά και πρακτική αξία, καθώς μια οποιαδήποτε βελτίωση της απόδοσης του αλγορίθμου προσομοίωσης της κβαντικής πληροφορίας, θα μας βοηθήσει αρκετά στη μελέτη των κβαντικών υπολογιστών. Επίσης θα μας δοθεί η δυνατότητα της επεξεργασία μεγάλου όγκου δεδομένων που βρίσκεται σε κλασική κατάσταση σε συνδυασμό με κβαντική πληροφορία.

Στο Κεφάλαιο 2 γίνεται μια εισαγωγή στις τεχνολογίες που χρησιμοποιήθηκαν και σε ήδη υπάρχουσες συσχετιζόμενες εργασίες, τις οποίες θα πρέπει να έχει υπόψη ο αναγνώστης για να κατανοήσει τη διαδικασία που ακολουθείται αργότερα. Παρουσιάζονται τα κβαντικά κυκλώματα, η προσομοίωση κβαντικών αλγορίθμων, οι διάφορες μορφές Binary Decision Diagrams (BDDs) που χρησιμοποιούνται και η τεχνική των συμβολικών αναπαραστάσεων [1].

Στο Κεφάλαιο 3 αναλύεται ο αλγόριθμος που υλοποιήσαμε, τα βήματα μέχρι την τελική του μορφή και οι διαφορές του με τον προηγούμενο αλγόριθμο.

Στο Κεφάλαιο 4 παρουσιάζονται αναλυτικά αποτελέσματα που προέκυψαν μετά από εφαρμογή του αλγορίθμου και σύγκριση αυτών με αποτελέσματα προηγούμενων μεθόδων και προσεγγίσεων.

Στο Κεφάλαιο 5 καταγράφονται τα οφέλη από τα αποτελέσματα του αλγορίθμου και η συμβολή του στη βελτίωση της απόδοσης όταν προσομοιώνουμε κβαντικά κυκλώματα. Επίσης προτείνονται μελλοντικές επεκτάσεις για περαιτέρω βελτίωση του αλγορίθμου.

Κεφάλαιο 2

Συσχετιζόμενες εργασίες

2.1 Κβαντική υπολογιστική επιστήμη

Ο πρώτος που συνέλαβε την ιδέα των κβαντικών υπολογιστών ήταν ο David Deutsch το 1984. Σύμφωνα με ένα άρθρο που δημοσίευσε, ένας κβαντικός υπολογιστής θα μπορεί να επεξεργαστεί όλες τις εκδοχές ενός προβλήματος ταυτόχρονα. Αυτό ουσιαστικά συμβαίνει επειδή τα κβαντικά bits (qubits: quantum binary digit) μπορούν να βρεθούν σε υπερθέσεις του 0 και του 1, σε αντίθεση με τα κλασικά bits, όπου θα είναι ή 0 ή 1. Η εισαγωγή των δεδομένων, θα γίνεται με περιστρεφόμενα σωματίδια, όπου ανάλογα με τη φορά περιστροφής, θα αντιπροσωπεύουν το 0 ή το 1. Εάν έχουμε 250 περιστρεφόμενα σωματίδια, ένας κβαντικός υπολογιστής θα μπορεί να εκτελεί 10^{75} ταυτόχρονους υπολογισμούς σε ένα μόλις δευτερόλεπτο.

Το 1994 ο Peter Shor προσδιόρισε ένα χρήσιμο πρόγραμμα για έναν κβαντικό υπολογιστή, το οποίο αναφέρεται στα βήματα που θα ακολουθούσε ώστε να παραγοντοποιήσει ένα μεγάλο αριθμό. Το πρόβλημα αυτό και η εύρεση γρήγορου τρόπου παραγοντοποίσης είναι ένα από τα μεγαλύτερα που αντιμετωπίζουν αυτή τη στιγμή οι μαθηματικοί. Για την παραγοντοποίηση ενός αριθμού με 129 ψηφία, εργάστηκαν 600 υπολογιστές για μήνες. Ένας κβαντικός υπολογιστής εφοδιασμένος με το πρόγραμμα του Σορ θα ήταν σε θέση να παραγοντοποιήσει έναν αριθμό κατά 1.000.000 φορές

μεγαλύτερο, στο 1 εκατομμυριοστό του χρόνου. Μέχρι στιγμής έχουν ήδη δημιουργηθεί κι άλλα προγράμματα, τα οποία όμως δυστυχώς δεν είμαστε σε θέση να χρησιμοποιήσουμε, αφού μας λείπει ο κβαντικός υπολογιστής.

2.1.1 Κβαντικός υπολογιστής

Κβαντικός υπολογιστής ονομάζεται οποιαδήποτε υπολογιστική συσκευή που χρησιμοποιεί χαρακτηριστικές κβαντομηχανικές ιδιότητες, όπως η αρχή της υπέρθεσης και της διεμπλοκής καταστάσεων για να πραγματοποιεί επεξεργασία δεδομένων. Σε έναν κλασικό υπολογιστή, στοιχειώδης μονάδα πληροφορίας πληροφορίας είναι το bit, ενώ σε έναν κβαντικό υπολογιστή το qubit. Η βασική αρχή της κβαντικής υπολογιστικής επιστήμης είναι το γεγονός ότι οι κβαντομηχανικές ιδιότητες της ύλης μπορούν να χρησιμοποιηθούν για την αναπαράσταση και τη δόμηση δεδομένων, καθώς και το γεγονός ότι μπορούν να επινοηθούν και να κατασκευαστούν μηχανισμοί βασισμένοι στην κβαντομηχανική για την επεξεργασία αυτών των δεδομένων. Η κβαντική υπολογιστική επιστήμη βρίσκεται ακόμα σε πειραματικό στάδιο, ωστόσο τα αποτελέσματα των πειραμάτων που έχουν πραγματοποιηθεί σε αυτό το πεδίο (με μικρό αριθμό qubits) είναι ενθαρρυντικά.

2.1.2 Συμβολισμός Bra/Ket

Καθώς οι κβαντικές καταστάσεις αναπαριστώνται από διανύσματα, υπάρχει η ανάγκη για ένα συνοπτικό συμβολισμό τους. Ο Dirac [5] εισήγαγε ένα νέο που ονομάζεται Bra/Ket συμβολισμός, όπου το ket $|x\rangle$ χρησιμοποιείται για να περιγράψει τα διανύσματα της στήλης και το bra $\langle x|$ υποδηλώνει το συζυγή συμμετρικό του $|x\rangle$. Τα πιο συχνά χρησιμοποιούμενα διανύσματα είναι τα

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.1}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.2}$$

2.1.3 Qubit

Στην περίπτωση των κλασικών υπολογιστών, το bit, το οποίο αποτελεί τη στοιχειώδη μονάδα πληροφορίας, μπορεί λάβει μία από δύο δυνατές τιμές, το 0 ή το 1. Το κβαντικό bit ή, όπως συνηθίζεται να λέγεται, qubit, είναι αντίστοιχα η στοιχειώδης μονάδα της κβαντικής πληροφορίας και έχει επίσης μία κατάσταση στην οποία βρίσκεται όπως το κλασικό bit. Δύο πιθανές καταστάσεις για ένα qubit είναι οι $|0\rangle$ και $|1\rangle$, που αναλογούν στις καταστάσεις 0 και 1 του κλασικού bit. Η βασική του διαφορά ενός qubit από ένα κλασικό bit είναι ότι το qubit μπορεί να βρίσκεται σε ένα συνδυασμό των δύο καταστάσεων $|0\rangle$ και $|1\rangle$ ταυτόχρονα, το οποίο ονομάζεται υπέρθεση

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.3}$$

Χρησιμοποιώντας τις 2.1 και 2.2, μπορούμε να ξαναγράψουμε την 2.3

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.4}$$

Τα α και β ονομάζονται πλάτη και είναι μιγαδικοί αριθμοί τέτοιοι ώστε

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.5}$$

Στην περίπτωση των κλασικών υπολογιστών, μπορούμε ανά πάσα στιγμή να εξετάσουμε την κατάσταση ενός bit και να δούμε αν είναι 0 ή 1. Όμως, δε συμβαίνει το ίδιο και με τα qubits, καθώς δε μπορούμε να εξετάσουμε την κατάσταση του, δηλαδή τις τιμές των α και β. Αντ'αυτού όταν εξετάζουμε την κατάσταση ενός qubit, παίρνουμε μια πιθανότητα $|\alpha|^2$ να έχει την τιμή 0 ή μια πιθανότητα $|\beta|^2$ να έχει την τιμή 1. Απ'αυτό επίσης προκύπτει η ισχύς της 2.5 καθώς το άθροισμα των πιθανοτήτων πρέπει να είναι ίσο με τη μονάδα.

Η διαδικασία κατά την οποία εξετάζουμε την κατάσταση ενός qubit ονομάζεται μέτρηση και μετά από αυτή το qubit περιέρχεται σε μία από τις κατάστασεις $|0\rangle$ ή $|1\rangle$, ανάλογα τα αποτελέσματα της μέτρησης. Αν, για παράδειγμα, έχουμε ένα qubit στην κατάσταση

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.6}$$

μετρώντας την κατάσταση του, θα μας δώσει αποτέλεσμα 0 με πιθανότητα $\frac{1}{\sqrt{2}} = 0.5$ ή το αποτέλεσμα 1 με ίδια πιθανότητα. Έστω λοιπόν ότι το αποτέλεσμα της μέτρησης είναι 0, το qubit θα περιέλθει στην κατάσταση $|0\rangle$.

Όπως γίνεται κατανοητό, ένα qubit δεν είναι ισοδύναμο με ένα κλασικό bit, ακόμη και εάν οι πιθανότητες του να είναι 0 ή 1 είναι ίσες με τις αντίστοιχες του qubit να μετρηθεί στις καταστάσεις $|0\rangle$ και $|1\rangle$. Η διαφορά έγκειται στο γεγονός ότι η κβαντική υπέρθεση του qubit ορίζει, εκτός από τις πιθανότητες, και μια σχετική φάση μεταξύ των δύο καταστάσεων, επιτρέποντας την εμφάνιση φαινομένων συμβολής των δύο καταστάσεων.

Μια αναλογία με την κυματική θα μας βοηθήσει να κατανοήσουμε τι σημαίνει αυτό. Όταν δύο κύματα συναντώνται σε ένα σημείο, το αποτέλεσμα της πρόσθεσης τους δεν εξαρτάται μόνο από το πλάτος του κάθε κύματος, αλλά και από την φάση στην οποία βρίσκεται. Κύματα που βρίσκονται στην ίδια φάση θα αλληλοενισχυθούν, ενώ κύματα σε αντίθεση φάση θα αλληλοαναιρεθούν. Κατά τον ίδιο τρόπο, όταν δύο ή περισσότερα qubit αλληλεπιδρούν σε έναν κβαντικό υπολογισμό, έχουν σημασία οι σχετικές τους φάσεις. Οι πιθανότητες κάθε κατάστασης δεν δίνονται από τους συντελεστές, αλλά από τα τετράγωνα των μέτρων των συντελεστών που ορίζουν την συγκεκριμένη διαμόρφωση του qubit.

2.1.4 Κβαντική πύλη

Μέχρι στιγμής είδαμε πως μπορούμε να περιγράψουμε κάποια κβαντική κατάσταση, όμως δεν είδαμε με ποιον τρόπο είμαστε σε θέση να δημιουργήσουμε μια επιθυμητή κατάσταση. Αρχικά, θα πρέπει να δεχτούμε ως δεδομένο ότι ένας κβαντικός υπολογιστής μπορεί να αρχικοποιήσει τα qubits στην κατάσταση $|0\rangle$ ή $|1\rangle$. Θεωρώντας ότι η απλούστερη πράξη στην περίπτωση των κλασικών υπολογιστών είναι η NOT, η οποία αντιστρέφει την τιμή ενός bit, θα ορίσουμε αντίστοιχα μια παρόμοια πράξη και για τους κβαντικούς υπολογιστές. Άρα πρέπει να βρούμε μια πράξη που μετατρέπει την κατάσταση $|0\rangle$ σε $|1\rangle$ και το αντίστροφο. Έτσι, ορίζουμε ένα δισδιάστατο πίνακα X με μέγεθος 2×2

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.7}$$

Πολλαπλασιάζοντας τον X με $|0\rangle$ ή $|1\rangle$, έχουμε αντίστοιχα τα παρακάτω αποτελέσματα

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{2.8}$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.9}$$

Όπως παρατηρούμε από τις 2.8 και 2.9, ο πίνακας X όντως αναπαριστά την επιθυμητή πράξη και μετατρέπει την κατάσταση $|0\rangle$ σε $|1\rangle$ και αντίστροφα, οπότε αποτελεί την κβαντική πύλη NOT. Στην πραγματικότητα, η πράξη αυτή επιδρά γραμμικά, το οποίο σημαίνει ότι φέρνει την κατάσταση $\alpha|0\rangle + \beta|1\rangle$ στην κατάσταση $\alpha|1\rangle + \beta|0\rangle$

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{2.10}$$

Συμπεραίνουμε ότι, εφόσον η κατάσταση ενός qubit περιγράφεται από ένα διάνυσμα 1×2, οι τελεστές θα πρέπει να αναπαριστώνται από δισδιάστατους

πίνακες μεγέθους 2×2. Επίσης για να μπορεί ένας πίνακας να θεωρηθεί κβαντική πύλη θα πρέπει να τηρεί ένα και μοναδικό περιορισμό, ο οποίος λέει ότι πρέπει ο πίνακας να είναι ορθογώνιος, το οποίο σημαίνει

$$U^\dagger U = U U^\dagger = I \tag{2.11}$$

όπου $U^\dagger$ είναι ο συζυγής του U και I είναι ένας μοναδιαίος πίνακας 2×2. Εύκολα μπορούμε να δούμε ότι ο X είναι ορθογώνιος, καθώς

$$X^\dagger X = X X^\dagger = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.12}$$

Στην πραγματικότητα, ορίζοντας ένα τελεστή U που είναι unitary, σημαίνει ότι μπορούμε να κατασκευάσουμε έναν άλλο τελεστή $U^\dagger$, ο οποίος έχει την ακριβώς αντίστροφη λειτουργία. Δηλαδή έστω ότι εφαρμόζουμε μια κβαντική πύλη U σε μια κατάσταση $|\kappa\rangle$ και παίρνουμε μια κατάσταση $|\lambda\rangle$. Μπορούμε να επιστρέψουμε στην αρχική κατάσταση $|\kappa\rangle$, εφαρμόζοντας τον $U^\dagger$ στην $|\lambda\rangle$.

$$|\kappa\rangle \xrightarrow{\mathrm{U}} \underbrace{U|\kappa\rangle}_{|\lambda\rangle} \xrightarrow{\mathrm{U}^\dagger} U^\dagger U|\kappa\rangle = I|\kappa\rangle = |\kappa\rangle \tag{2.13}$$

Μια πολύ σημαντική και χρήσιμη κβαντική πύλη είναι η πύλη Hadamard

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.14}$$

η οποία είναι ορθογώνια καθώς $\mathrm{H} = \mathrm{H}^\dagger$ και $\mathrm{H}\mathrm{H}^\dagger = \mathrm{I}$. Η αξία της πύλης Hadamard, έγκειται στο γεγονός ότι, εφαρμόζοντάς την στις κατάστασεις $|0\rangle$ και $|1\rangle$, δημιουργεί μια κατάσταση ίσης υπέρθεσης

$$\mathrm{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.15}$$

$$\mathrm{H}|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{2.16}$$

το οποίο σημαίνει ότι η πιθανότητα να περιέλθει το qubit στην κατάσταση $|0\rangle$ είναι ίση με την πιθανότητα να περιέλθει στην κατάσταση $|1\rangle$, δηλαδή υπάρχει ίδια πιθανότητα να μετρήσουμε 0 και να μετρήσουμε 1.

Μια άλλη κβαντική πύλη είναι η πύλη

$$\mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.17}$$

η οποία όταν εφαρμοστεί πάνω σε μια κβαντική κατάσταση, έχει σαν αποτέλεσμα την αρχική αυτή κατάσταση.

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\mathrm{I}} \alpha|0\rangle + \beta|1\rangle \tag{2.18}$$

2.1.5 Ελεγχόμενες πύλες

Κάποιες φορές μπορεί να έχουμε ένα σύστημα που αποτελείται από δύο qubits και να θέλουμε να εφαρμόσουμε μια πύλη στο ένα ανάλογα την τιμή που έχει το άλλο. Για παράδειγμα, αν θέλουμε να εφαρμόσουμε την πύλη NOT στο δεύτερο, όταν το πρώτο είναι στην κατάσταση $|1\rangle$. Αυτή η πύλη λέγεται ελεγχόμενη πύλη NOT (CNOT) και περιγράφεται από τον πίνακα

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.19}$$

Εύκολα μπορούμε να διαπιστώσουμε ότι η πύλη CNOT κάνει τις παρακάτω μετατροπές

$$|00\rangle \xrightarrow{\mathrm{CNOT}} |00\rangle \tag{2.20}$$

$$|01\rangle \xrightarrow{\mathrm{CNOT}} |01\rangle \tag{2.21}$$

$$|10\rangle \xrightarrow{\mathrm{CNOT}} |11\rangle \tag{2.22}$$

$$|11\rangle \xrightarrow{\mathrm{CNOT}} |10\rangle \tag{2.23}$$

δηλαδή το πρώτο qubit παραμένει το ίδιο, ενώ το δεύτερο αλλάζει στο αντίστροφό του αν το πρώτο είναι στην κατάσταση $|1\rangle$. Στη γενική του μορφή αυτό εκφράζεται ως

$$|A, B\rangle \xrightarrow{\mathrm{CNOT}} |A, B \oplus A\rangle \tag{2.24}$$

όπου A, B $\epsilon \{0, 1\}$ και το $\oplus$ είναι η πράξη XOR μεταξύ των A και B. Το πρώτο qubit που καθορίζει την τιμή του δεύτερου λέγεται qubit ελέγχου και το δεύτερο qubit του οποίου η τιμή καθορίζεται από το πρώτο λέγεται qubit στόχος.

Γενικότερα, όταν έχουμε μια πύλη U, κατασκευάζουμε μια ελεγχόμενη πύλη U, έτσι ώστε να αλλάζει η τιμή του δεύτερου qubit σύμφωνα με την U, μόνο όταν το πρώτο qubit είναι $|1\rangle$

$$CU = \begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix} \tag{2.25}$$

όπου I είναι ο 2×2 μοναδιαίος πίνακας, U είναι ο 2×2 πίνακας που αναπαριστά την πύλη U και ο 0 είναι ο 2×2 πίνακας με όλα τα στοιχεία του ίσα με μηδέν (0). Καθώς ο U είναι ορθογώνιος, και ο CU είναι επίσης ορθογώνιος.

Μπορούμε να ορίσουμε πύλες με περισσότερα από ένα qubits ελέγχου. Για παράδειγμα η πύλη *Toffoli*, είναι μια ελεγχόμενη NOT πύλη με δύο qubits ελέγχου. Η αναπαράστασή της σε πίνακα είναι

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.26}$$

και όταν εφαρμοστεί σε ένα κβαντικό καταχωρητή τριών qubits, αντιστρέφει το target qubit μόνο όταν και τα δύο άλλα qubits ελέγχου είναι ταυτόχρονα στην κατάσταση $|1\rangle$

$$|A, B, C\rangle \xrightarrow{\text{Toffoli}} |A, B, C \oplus AB\rangle \tag{2.27}$$

Όπως γίνεται αντιληπτό, η πύλη Toffoli μπορεί να χρησιμοποιηθεί προσομοιώντας την πύλη NAND, καθώς αν η τιμή του C είναι 1 στην 2.27, τότε

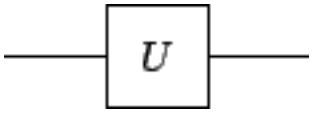
το αποτέλεσμα στο τρίτο qubit είναι η πράξη NAND του A και B

$$|A, B, 1\rangle \xrightarrow{\text{Toffoli}} |A, B, 1 \oplus AB\rangle = |A, B, \neg AB\rangle \tag{2.28}$$

Η διαφορά της πύλης Toffoli από την κλασική NAND είναι ότι η Toffoli, ούσα κβαντική πύλη, είναι αναστρέψιμη. Τη στιγμή που έχουμε υλοποιήσει σε κβαντικό υπολογιστή, την πύλη NAND, μπορούμε να προσομοιώσουμε οποιαδήποτε άλλη πύλη, το οποίο σημαίνει ότι μπορεί να εκτελέσει τις ίδιες λειτουργίες με έναν κλασικό υπολογιστή.

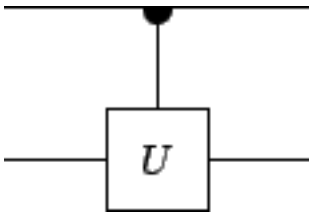
2.1.6 Κβαντικό κύκλωμα

Σε αναλογία με τους κλασικούς υπολογιστές, υπάρχουν σύμβολα που υποδηλώνουν τις κβαντικές πύλες μέσα σε κυκλώματα. Μια πύλη ενός qubit σχεδιάζεται με ένα κουτί που έχει το όνομα της πύλης. Ένα παράδειγμα μιας πύλης U φαίνεται στο σχήμα 2.1.



Σχήμα 2.1: Συμβολισμός πύλης U

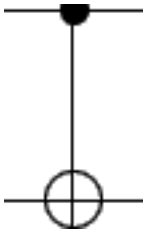
Ένα παράδειγμα συμβολισμού μιας ελεγχόμενης πύλης U φαίνεται στο σχήμα 2.2.



Σχήμα 2.2: Συμβολισμός ελεγχόμενης πύλης U

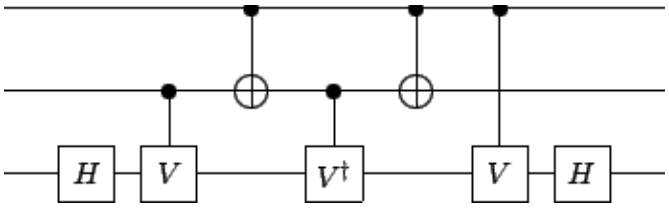
Στο σχήμα 2.2, η μαύρη τελεία αναπαριστά το qubit ελέγχου, ενώ τα άλλα είναι τα qubits στόχοι. Η ελεγχόμενη πύλη NOT (CNOT), λόγω της

μεγάλης σημασίας της, έχει ένα ιδαίτερο συμβολισμό που φαίνεται στο σχήμα 2.3



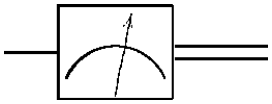
Σχήμα 2.3: Συμβολισμός ελεγχόμενης πύλης NOT (CNOT)

Με αυτόν τον τρόπο, μπορούμε να χτίσουμε πιο πολύπλοκα κβαντικά κυκλώματα. Η αναπαράσταση ενός πιο πολύπλοκου κυκλώματος φαίνεται στο 2.4



Σχήμα 2.4: Παράδειγμα κβαντικού κυκλώματος

Τέλος, χρειάζεται και ένας συμβολικός τρόπος αναπαράστασης της μέτρησης των qubits μέσα σε ένα κύκλωμα. Η λειτουργία αυτή μετατρέπει ένα απλό qubit σε ένα κλασικό bit με κάποια πιθανότητα. Το κλασικό bit διαχωρίζεται από το qubit με τη σχεδίαση διπλής γραμμής, όπως φαίνεται στο σχήμα 2.5



Σχήμα 2.5: Αναπαράσταση μέτρησης qubit

2.2 Προσομοίωση κβαντικών αλγορίθμων

Μέχρι στιγμής, η υπάρχουσα τεχνολογία μας επιτρέπει να δημιουργήσουμε κβαντικούς υπολογιστές με δυνατότητα διαχείρισης πολύ περιορισμένου αριθμού qubits. Επίσης οι συκεκριμένοι κβαντικοί υπολογιστές χαρακτηρίζονται από ιδιαίτερη αστάθεια και ως εκ τούτου, δε μπορούμε να δοκιμάσουμε πάνω σ'αυτούς, κβαντικούς αλγορίθμους. Δεδομένης, λοιπόν, της απουσίας κβαντικών υπολογιστών, η μόνη μέθοδος για να ερευνήσουμε την απόδοση κβαντικών αλγορίθμων είναι η προσομοίωση τους σε κλασικούς υπολογιστές.

Πλήθος προγραμμάτων έχουν αναπτυχθεί [7], που αποσκοπούν να προσομοιώσουν, με όσο το δυνατόν καλύτερα αποτελέσματα, κβαντικούς αλγορίθμους. Τα περισσότερα από αυτά χρησιμοποιούν αποκλειστικά την αναπαράσταση διανύσματος κατάστασης και ως εκ τούτου οι απαιτήσεις σε μνήμη μεγαλώνουν εκθετικά. Ένα παράδειγμα είναι το Fraunhofer Quantum Computing Portal [6], στο οποίο έχει κάποιος τη δυνατότητα να εισάγει κβαντικά κυκλώματα μέχρι 31 qubits, τα οποία διαχειρίζονται σε ένα cluster 32 κόμβων.

Ένα από τα πιο γνωστά προγράμματα για προσομοίωση είναι το Matlab και, το αντίστοιχο ελεύθερο λογισμικό ανοιχτού κώδικα, Octave, τα οποία έχουν σχεδιαστεί για να διαχειρίζονται διανύσματα και πίνακες και να εκτελούν πράξεις μεταξύ τους. Όμως, μετά από κάποιο, σχετικά μικρό, πλήθος qubits (≈ 14), τα προγράμματα αυτά εξαντλούν όλη την διαθέσιμη μνήμη και δε μπορούν να εκτελέσουν παραπάνω πράξεις.

Υπάρχει επίσης ένας προσομοιωτής κβαντικών κυκλωμάτων, ο οποίος, με διαφορετική αναπαράσταση των διανυσμάτων καταστάσεων και των πράξεων, έχει καλύτερα αποτελέσματα. Ο προσομοιωτής αυτός λέγεται QuIDDPro [17, 19]. Ο συγκεκριμένος αλγόριθμος κάνει χρήση ενός τύπου των Algebraic

Decision Diagrams [2], τα Quantum Information Decision Diagrams και έτσι επιτυγχάνει γρηγορότερη κβαντική προσομοίωση με λιγότερη κατανάλωση μνήμης. Τα αποτελέσματα της έρευνας του συγκεκριμένου αλγορίθμου, αποδεικνύουν ότι έχει καλύτερα αποτελέσματα από αυτά του Matlab.

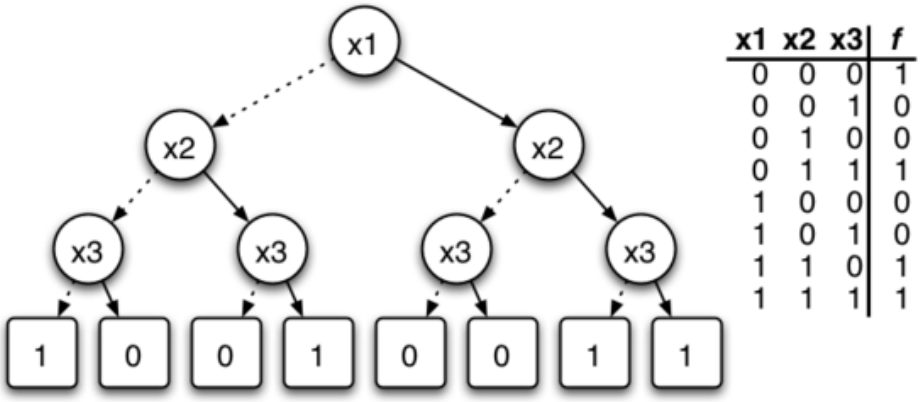
Όλες οι παραπάνω τεχνικές είναι υποδεέστερες και έχουν χειρότερα αποτελέσματα από τη μέθοδο των συμβολικών αναπαραστάσεων με χρήση Multi Terminal Binary Decision Diagrams, η οποία περιγράφεται λίγο παρακάτω. Για το λόγο αυτό, η παρούσα εργασία βασίζεται σε ένα τέτοιο αλγόριθμο επεκτείνοντας τον με χρήση εναλλακτικών μορφών Binary Decision Diagrams.

2.3 Binary Decision Diagrams

Το binary decision diagram (BDD) είναι στη γενική του μορφή ένας κατευθυνόμενος άκυκλος γράφος $G = (V \bigcup T, E, next, var, val, s)$ που χρησιμοποιείται για την αναπαράσταση μια δυαδικής συνάρτησης $f(x_0, \ldots, x_{n-1})$ σε ένα χώρο D, όπου x_i είναι οι δυαδικές μεταβλητές. Σε κάθε κορυφή t αντιστοιχεί μια τιμή $val(t) \in D$, ενώ σε κάθε ενδιάμεσο κόμβο u δίνεται η ονομασία από μια δυαδική μεταβλητή $var(u)$ και έχει ακριβώς δύο γείτονες, το γείτονα-0 και το γείτονα-1, οι οποίοι δηλώνονται με $next(u, 0)$ και $next(u, 1)$ αντίστοιχα. Σαν αρχικός κόμβος του γράφου ορίζεται μία κορυφή s. Σε κάθε ορισμό τιμών στις x_i, αντιστοιχεί και ένα μοναδικό μονοπάτι κατά μήκος του γράφου, ξεκινώντας από τον s και καταλήγωντας στον τερματικό κόμβο t, παράγοντας την τιμή $val(t)$.

Έστω ότι έχουμε τη συνάρτηση f(x1, x2, x3), της οποίας ο πίνακας αληθείας φαίνεται στο σχήμα 2.6. Στο δέντρο στα αριστερά, η τιμή της συνάρτησης μπορεί να βρεθεί για ένα συγκεκριμένο σύνολο μεταβλητών ακο-

λουθώντας το μονοπάτι στο γράφο από τον αρχικό κόμβο μέχρι τον τερματικό. Στο σχήμα 2.6, οι διακεκομμένες γραμμές αναπαριστούν τις ακμές σε γείτονα-0, ενώ οι ολόκληρες γραμμές αναπαριστούν ακμές σε γείτονα-1. Έτσι, για να βρούμε την τιμή της συνάρτησης (x1=0, x2=1, x3=1), ξεκινάμε από το x1, διασχίζοντας το γράφο προς τα κάτω ακολουθώντας τη διακεκομμένη γραμμή προς το x2 (εφόσον το x1 έχει τιμή 0) και στη συνέχεια ακολουθούμε τις ολόκληρες γραμμές (εφόσον τα x2 και x3 έχουν τιμή 1). Αυτό μας οδηγεί στον τερματικό κόμβο 1, ο οποίος είναι η τιμή της f (x1=0, x2=1, x3=1).

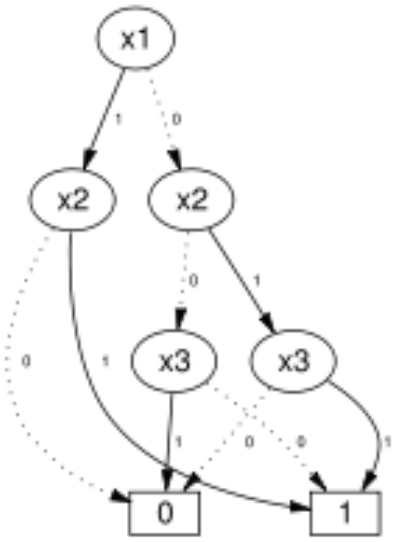


x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Σχήμα 2.6: Binary Decision Tree

Ο Bryant [10] ανέπτυξε αποδοτικούς αλγορίθμους για την περίπτωση όπου $D = \{0, 1\}$ (για παράδειγμα, BDDs για δυαδικές συναρτήσεις), εισάγοντας δύο συνθήκες στη δομή του BDD, τη διάταξη και τη μείωση. Με τον όρο διάταξη εννοούμε ότι οι τιμές των μεταβλητών κατά μήκος ενός μονοπατιού του γράφου αυξάνονται, ενώ με τον όρο μείωση εννοούμε ότι (α) οι γείτονες κάθε κόμβου είναι ξεχωριστοί και (β) δεν υπάρχουν ισοδύναμοι κόμβοι (να έχουν δηλαδή ίδια τιμή). Έτσι, το δέντρο δυαδικής αναπαράστασης (binary decision tree) του σχήματος 2.6, μπορεί να μετατραπεί σε διάγραμμα δυαδική αναπαράστασης (binary decision diagram),

σύμφωνα με τις δύο αυτές συνθήκες και έτσι το αποτέλεσμα της πράξης αυτής φαίνεται στο BDD του σχήματος 2.7.



Σχήμα 2.7: Binary Decision Diagram

Γενικότερα όταν γίνεται αναφορά στον όρο BDD, σχεδόν πάντα αυτό αντιστοιχεί στο Reduced Ordered Binary Decision Diagram (ROBDD), το οποίο είναι το BDD που ικανοποιεί τις συνθήκες της διάταξης και της μείωσης.

Μια παρόμοια με τα ROBDD εργασία πάνω στα BDD, είναι τα Multi Terminal BDDs (MTBDDs), τα οποία παρουσιάστηκαν από τους Fujita, McGeer και Yanget [11]. Ένα MTBDD είναι ένα ROBDD όπου το D είναι αριθμητικό σύνολο, συνήθως $\mathbb{R}$ ή $\mathbb{C}$.

Σε μια σειρά από έρευνες [19, 17, 18], μελετήθηκε το ζήτημα της δυναμικής των QuIDDs, μια μικρή παραλλαγή των MTBDDs, για κβαντική προσομοίωση. Τα QuIDDs των n δυαδικών μεταβλητών αναπαριστούν διανύσματα κατάστασης των n qubits και τα QuIDDs των $2n$ δυαδικών με-

ταβλητών αναπαριστούν n-qubit κβαντικούς τελεστές. Τα πειράματά τους έδειξαν ότι τα QuIDDs προσφέρουν σημαντική συμπίεση των κβαντικών καταστάσων και επιτρέπουν αποδοτική εφαρμογή των κβαντικών τελεστών, σε σύγκριση με διάφορες άλλες τεχνικές που χρησιμοποιούν αναπαράσταση διανύσματος κατάστασης. Αρχικά η μέθοδος εφαρμόστηκε πάνω στον αλγόριθμο του Grover, ενώ αργότερα ο Κουφογιαννάκης [13] εφάρμοσε την τεχνική αυτή και στον αλγόριθμο του Shor, επιτυγχάνοντας αρκετά μεγάλη συμπίεση. Πρόσφατα, άλλες παραλλαγές των MTBDD έχουν κάνει την εμφάνιση τους κατά την εύρεση νέων μεθόδων για προσομοίωση κβαντικών υπολογιστών. [8, 14].

2.4 Αλγόριθμος συμβολικών αναπαραστάσεων

Ο αλγόριθμος συμβολικών αναπαραστάσεων βασίζεται στα Multi Terminal Binary Decision Diagrams για προσομοίωση κβαντικών κυκλωμάτων. Το χαρακτηριστικό του συγκεκριμένου αλγορίθμου και ταυτόχρονα η ειδοποιός διαφορά του από άλλους αλγορίθμους είναι η δυνατότητα διαχείρισης των κβαντικών τελεστών, όχι πλέον σαν πίνακες είτε σε κάποια συμπιεσμένη μορφή είτε όχι, αλλά με συμβολικές αναπαραστάσεις.

Η μέθοδος αυτή φαίνεται να παρουσιάζει πολύ σημαντικά πλεονεκτήματα σχετικά με προηγούμενες προσεγγίσεις ως προς την απόδοση και την επίδοση. Επιπροσθέτως, τα πειραματικά αποτελέσματα δείχνουν ότι η συγκεκριμένη τεχνική επιδεικνύει σημαντική βελτίωση έναντι παλαιότερων, ακόμα και δύο τάξεις μεγέθους, και παρουσιάζεται σαφώς πιο γρήγορη κατά την εκτέλεση της.

Στο σημείο αυτό, θα παρουσιαστούν κάποιοι ορισμοί που θα μας επιτρέψουν την αναπαράσταση των κβαντικών τελεστών ως αναδρομικές εκφράσεις πινάκων. Οι εκφράσεις αυτές επακολούθως χρησιμοποιούνται για

την εξεύρεση και απόδειξη της ορθότητας του αλγορίθμου.

2.4.1 Powermatrices και powervectors

Ένα powermatrix A (ή PM για συντομία) είναι ένας μιγαδικός τετραγωνικός πίνακας διαστάσεων $2^k \times 2^k$, όπου το $k \geqslant 0$ είναι η *τάξη* του A και χαρακτηρίζεται από το $ord(A)$. Τα PM τάξης 0 είναι *μονοδιάστατα*. Δύο (ή περισσότερα) PM της ίδιας τάξης ονομάζονται *όμοια*. Το I_k υποδηλώνει το μοναδιαίο PM τάξης k.

Ο τελεστής $\square$ ορίζεται ως

$$A \square B = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & B \end{bmatrix}$$

όπου $\mathbf{0}$ είναι το μηδενικό PM όμοιο με το A, B.

Ένα powervector a (ή PV για συντομία) τάξης k είναι ένα διάνυσμα μεγέθους 2^k, όπου το $k \geqslant 0$ είναι η *τάξη* του PV a και χαρακτηρίζεται από το $ord(a)$. Όπως και στην περίπτωση των PM, τα PV τάξης 0 είναι *μονοδιάστατα* και τα PV ίδιας τάξης είναι *όμοια*.

Έστω ότι έχουμε δύο όμοια PV a και b, το $a \square b$ είναι ένα PV τάξης $ord(a) + 1$ και αποτελεί την αλληλουχία των a και b. Να σημειωθεί ότι ο τελεστής $\square$ εφαρμόζεται και σε PV και σε τετραγωνικά PM.

Τα MTBDDs συνήθως χρησιμοποιούνται ως συναρτήσεις δυαδικών μεταβλητών. Όμως, ο συμβολισμός αυτός είναι δυσχρηστος σε περιπτώσεις γραμμικών αλγεβρικών τελεστών και εξισώσεων, το οποίο εισάγει λάθη χειρισμού των δεικτών. Αντιθέτως, το $\square$ επιτρέπει τη διαχείριση MTBDDs με αλγεβρικό τρόπο αποφεύγοντας, σχεδόν εξ'ολοκλήρου, το συμβολισμό με δείκτες, το οποίο οδηγεί σε περισσότερο ξεκάθαρους τύπους.

Παρακάτω παρουσιάζονται κάποιες ιδιότητες των $\square$. Με κεφαλαία δηλώνονται τα PM, με πεζά λατινικά τα PV και με ελληνικά γράμματα τα

μονοδιάστατα.

$$\lambda(A \,\square\, B) = (\lambda A) \,\square (\lambda B)$$

$$(A_0 \,\square\, A_1) \otimes U = (A_0 \otimes U) \,\square (A_1 \otimes U)$$

$$(A \,\square\, B)^t = A^t \,\square\, B^t$$

$$(A \,\square\, B)(C \,\square\, D) = (AC) \,\square (BD)$$

$$\lambda(a \,\square\, b) = (\lambda a) \,\square (\lambda b)$$

$$(a_0 \,\square\, a_1) \otimes u = (a_0 \otimes u) \,\square (a_1 \otimes u)$$

$$(a \,\square\, b) + (c \,\square\, d) = (a + c) \,\square (b + d)$$

$$(A \,\square\, B) + (C \,\square\, D) = (A + C) \,\square (B + D)$$

$$(A \,\square\, B)(a \,\square\, b) = (Aa) \,\square (Bb)$$

2.4.2 Υλοποίηση αλγορίθμου

Στην υλοποίηση του αλγορίθμου, τα MTBDDs αναπαριστούν μιγαδικά PVs. Τα PMs δεν υλοποιήθηκαν ως MTBDDs, παρόλο που θα μπορούσε να γίνει κάτι τέτοιο σύμφωνα με το [17]. Εκτός τους MTBDD αλγορίθμους που περιγράφονται στο [11], υλοποιήθηκαν επίσης συναρτήσεις για μέτρηση και εφαρμογή κβαντικών τελεστών.

Η υλοποίηση κάνει χρήση των παρακάτω συναρτήσεων:

TNode(z**):** Επιστρέφει ένα μοναδικό τερματικό κόμβο u με $\text{val}(u) = z$.

Node(a**,**b**,**i**):** Αν $a = b$ επιστρέφει a, διαφορετικά επιστρέφει ένα μοναδικό εσωτερικό (μη τερματικό) κόμβο u με $\text{var}(u) = i$, $\text{next}(u, 0) = a$ και $\text{next}(u, 1) = b$.

Decompose(a**,**i**):** Αν $i < \text{var}(a)$, επιστρέφει το ζεύγος κόμβων (a, a), διαφορετικά επιστρέφει το ζεύγος κόμβων $(\text{next}(a, 0), \text{next}(a, 1))$.

Lookup(k), Cache(k,v): Υλοποιεί μια αντιστοιχία κλειδιών k σε τιμές v βασισμένη σε πίνακες κατακερματισμού.

Ο κώδικας φαίνεται στον αλγόριθμο 1, σαν ψευδοκώδικας σε python. Η υλοποίηση χωρίζεται σε δύο αναδρομικές συναρτήσεις. Η πρώτη συνάρτηση, **Apply(F, ψ)** είναι η βασική συνάρτηση που καλείται. Δέχεται μόνο δύο MTBDDs σαν ορίσματα, τον δυαδικό προβολέα F και το διάνυσμα κατάστασης ψ. Τα k και $U = \begin{bmatrix} u_0 & u_1 \\ u_2 & u_3 \end{bmatrix}$ του τελεστή $[F : k : U]$ είναι καθολικές μεταβλητές. Η συνάρτηση **Apply** είναι καλείται αναδρομικά, μέχρι να φτάσει στον κόμβο που ανήκεια στα qubits πριν το k, όπου και η διαδικασία μεταφέρεται στην **ApplySwap**. Η **ApplySwap** διαφέρει στο γεγονός ότι δέχεται ένα ζεύγος από διανύσματα κατάστασης και επιστρέφει επίσης ένα ζεύγος διανυσμάτων κατάστασης. Η γενική ιδέα είναι να αποφευχθεί άσκοπη κλήση της συνάρτησης **Node**, αφού είναι αρκετά ακριβή (απαιτεί δέσμευση μνήμης και συχνή παραπομπή στον πίνακα κατακερματισμού).

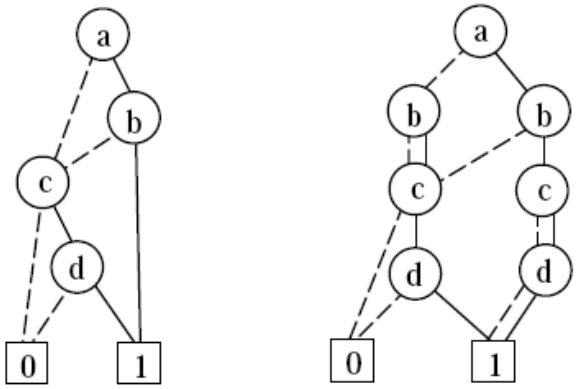
2.5 Εναλλακτικές μορφές BDDs

Κατά τη χρήση των BDDs, υπάρχουν περιπτώσεις όπου το μέγεθός του αυξάνεται δραματικά, δυσκολεύοντας τη διαδικασία και καθιστώντας πολλές φορές αδύνατη την επεξεργασία των δεδομένων. Ιδιαίτερα σε περιπτώσεις όπου τα δεδομένα είναι κατανεμημένα διάσπαρτα στο χώρο, το μέγεθος των εν λόγω δομών αυξάνεται εκθετικά. Για την επίλυση του προβλήματος αυτού, είναι επιτακτική η ανάγκη εύρεσης και χρήσης εναλλακτικών μορφών δομών για τη διαχείριση τέτοιου είδους δεδομένων.

Οι εναλλακτικές μορφές διαγραμμάτων, που χρησιμοποιούνται στην παρούσα εργασία είναι δύο. Τα Zero Suppressed Binary Decision Diagrams (ZSDD ή ZDD) [4], στα οποία ένας κόμβος αφαιρείται όταν ο γείτονας-1 είναι ίσος με μηδέν, σε αντίθεση με τα MTBDDs, όπου ένας κόμβος αφαι-

ρείται όταν οι δύο γείτονες είναι ίδιοι. Η διαγραμματική διαφορά μεταξύ των δύο μορφών φαίνεται στο σχήμα 2.8.

Η δεύτερη μορφή BDDs που χρησιμοποιείται, διαφέρει ελάχιστα από τα ZDDs και λέγεται One Suppressed Binary Decision Diagrams (OSDD ή ODD). Στην περίπτωση των ODDs, ένας κόμβος αφαιρείται από το γράφο, όταν ο γείτονας-0 είναι ίσος με μηδέν.



Σχήμα 2.8: Το BDD και ZDD για τη συνάρτηση F = ab + cd

Η στοιχειώδης αυτή διαφοροποίηση του κανόνα δημιουργίας μεταξύ των BDDs και των ZDDs και ODDs, εξηγεί τον πρηγούμενο ισχυρισμό μας για καλύτερη διαχείριση δεδομένων και αποδοτικότερη επεξεργασία μεταξύ των δομών αυτών, στην περίπτωση όπου έχουμε αραιά κατανεμημένα δεδομένα.

Algorithm 1 Αλγόριθμος MTBDD εφαρμογής ενός τελεστή.

Apply(F**,**ψ**) {**
    if $F = 0$ **or** $\psi = 0$**: return** ψ
    if Lookup(F**,**ψ**)** $\neq$ **nil: return Lookup(**F**,**ψ**)**
    v **:=** $\min(\text{var}(F), \text{var}(\psi))$
    ψ_0, ψ_1 **:= Decompose(**ψ, v**)**
    if $k \leq v$**:**
        ψ_a, ψ_b **:= ApplySwap(**F, ψ_0, ψ_1**)**
        ψ' **:= Node(**ψ_a, ψ_b, v**)**
    else:
        F_0, F_1 **:= Decompose(**F, v**)**
        ψ' **:= Node(Apply(**F_0, ψ_0**), Apply(**F_1, ψ_1**),** v**)**
    Cache((F, ψ)**,** ψ'**)**
    return ψ'
}

ApplySwap(F**,** ψ_0**,** ψ_1**) {**
    if $F = 0$**: return** (ψ_0, ψ_1)
    if Lookup(F, ψ_0, ψ_1**)** $\neq$ **nil: return Lookup(**F, ψ_0, ψ_1**)**
    v **:=** $\min(\text{var}(F), \text{var}(\psi_0), \text{var}(\psi_1))$
    if $v = \infty$**:** *# terminals reached*
        ψ_a, ψ_b **:= (TNode(**$u_0 \,\text{val}(\psi_0) + u_1 \,\text{val}(\psi_1)$**), TNode(**$u_2 \,\text{val}(\psi_0) + u_3 \,\text{val}(\psi_1)$**))**
    else:
        F_0, F_1 **:= Decompose(**F, v**)**
        y_0, y_1 **:= Decompose(**ψ_0, v**)**
        y_2, y_3 **:= Decompose(**ψ_1, v**)**
        q_0, q_1 **:= ApplySwap(**F_0, y_0, y_2**)**
        q_2, q_3 **:= ApplySwap(**F_0, y_1, y_3**)**
        ψ_a, ψ_b **:= (Node(**q_0, q_2, v**), Node(**q_1, q_3, v**))**
    Cache($(F, \psi_0, \psi_1), (\psi_a, \psi_b)$**)**
    return (ψ_a, ψ_b)
}

Κεφάλαιο 3

Αλγόριθμος προσομοίωσης

Στο κεφάλαιο αυτό περιγράφουμε τις λεπτομέρειες υλοποίησης του αλγορίθμου προσομοίωσης στα πλαίσια της παρούσας εργασίας. Η αρχική σχεδίαση και υλοποίηση του αλγορίθμου των συμβολικών αναπαραστάσεων [1] έγινε με τέτοιο τρόπο ώστε να επιτρέπεται και να διευκολύνεται η χρήση Multi Terminal Binary Decision Diagrams, χωρίς να υπάρχει δυνατότητα χρήσης εναλλακτικών μορφών διαγραμμάτων με κάποια μορφής suppression. Για το λόγο αυτό, έγιναν κάποιες απαραίτητες αλλαγές στον κώδικα του αλγορίθμου, ώστε να καταστεί εφικτή η εισαγωγή δομών για τη διαχείριση των δεδομένων, διαφορετικών από αυτές των MTBDDs.

Στην περίπτωση της συμβολικής αναπαράστασης, τα MTBBDs παρουσιάζονται ως συναρτήσεις δυαδικών μεταβλητών. Όμως, ο συμβολισμός των συναρτήσεων είναι αρκετά περίπλοκος και δυσνόητος. Το πρόβλημα αυτό λύνεται με την εισαγωγή του τελεστή $\square$, ο οποίος μας επιτρέπει να αναπαραστήσουμε τις συναρτήσεις με πιο απλό τρόπο.

Με τη χρήση του τελεστή $\square$, μια σχέση όπως στην εξίσωση 3.1 ή στην εξίσωση 3.2,

$$3 \square 4 = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \square \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 3 \\ 4 \end{bmatrix} \tag{3.2}$$

μπορεί να αναπαρασταθεί με δενδρική μορφή, ισοδύναμη με ένα MTBDD, το οποίο συμφωνεί με τον κανόνα της διάταξης, αλλά όχι και αυτόν της μείωσης. Αυτό σημαίνει ότι δεν υπάρχουν ίδια κομμάτια του δέντρου παραπάνω από μία φορά αλλά δεν έχει γίνει κάποια απαλειφή κόμβου. Άρα, οι συναρτήσεις του αλγορίθμου μπορούν να χρησιμοποιηθούν στην περίπτωση όπου στο διάγραμμα δεν έχουμε εφαρμόσει κάποιας μορφής suppression.

Όμως δεν αρκεί αυτό, καθώς θέλουμε να εφαρμόσουμε τον αλγόριθμο χρησιμοποιώντας και εναλλακτικές δομές, στις οποίες εφαρμόζεται απαλειφή κόμβων. Για να μπορέσουμε να το κάνουμε αυτό, αρκεί η πράξη που θα γίνει πάνω στο BDD, να παίρνει σαν είσοδο διαγράμματα που έχουν υποστεί suppression και να μας επιστρέφει σαν έξοδο διαγράμματα που επίσης έχουν κάποιας μορφής suppression, ίδιας με αυτής των εισόδων. Μελετώντας ένα παράδειγμα τέτοιας πράξης, της πρόσθεσης δύο κόμβων, θα διαπιστώσουμε ότι έχουμε το δικαίωμα να εφαρμόσουμε τον αλγόριθμο και σε suppressed BDDs.

3.1 Πρόσθεση δύο κόμβων

Για την πρόσθεση δύο κόμβων στην περίπτωση όπου εφαρμόζουμε equal suppression, ισχύει η εξίσωση 3.3.

$$(a_0 \square a_0) + (b_0 \square b_0) = (a_0 + b_0) \square (a_0 + b_0) \tag{3.3}$$

Αντίστοιχα για one και zero suppression, ο συμβολισμός της πρόσθεσης δύο κόμβων φαίνεται στις εξισώσεις 3.4 και 3.5.

$$(a_0 \square 0) + (b_0 \square 0) = (a_0 + b_0) \square (0 + 0) \tag{3.4}$$

$$(0 \square a_0) + (0 \square b_0) = (0 + 0) \square (a_0 + b_0) \tag{3.5}$$

Παρατηρώντας τις τρεις εξισώσεις, συμπεραίνουμε ότι και στις τρεις περιπτώσεις suppression, όταν η πρόσθεση έχει δύο suppressed ορίσματα, επιστρέφει επίσης ένα suppressed αποτέλεσμα. Έτσι, αυτό μας δίνει το δικαίωμα να αγνοήσουμε τους κόμβους που έχουν απαλειφθεί λόγω του suppression που έχει εφαρμοστεί και να χρησιμοποιήσουμε τον αλγόριθμο της πρόσθεσης και στις περιπτώσεις αυτές. Ο ψευδοκώδικας της συνάρτησης *Add* φαίνεται στον αλγόριθμο 2.

Algorithm 2 Αλγόριθμος πρόσθεσης κόμβων.

Add(a, b**) {**
    if $a = 0$**: return** b
    if $b = 0$**: return** a
    if Lookup(a, b**)** $\neq$ **nil: return Lookup(**a, b**)**
    v **:=** $\min(\text{var}(a), \text{var}(b))$
    if $v = \infty$**:** *# terminals reached*
        ψ **:= (TNode(**$\text{val}(a) + \text{val}(b)$**))**
    else:
        a_0, a_1 **:= Decompose(**a, v**)**
        b_0, b_1 **:= Decompose(**b, v**)**
        ψ **:= Node(Add(**a_0, b_0**), Add(**a_1, b_1**),** v**)**
    Cache(a, b**)**
    return ψ
}

Όπως αναφέραμε προηγούμενως, η παραπάνω μέθοδος μπορεί να εφαρμοστεί μόνο όταν η πράξη επιστρέφει αποτελέσματα με suppression και μάλιστα ίδιο με αυτό των εισόδων της. Όταν δε συμβαίνει αυτό, δε μπορούμε να ακολουθήσουμε τον τρόπο αυτό και να αγνοήσουμε τους κόμβους που έχουν απαλειφθεί, και έτσι ο αλγόριθμος θα πρέπει να εκτελεστεί σε κάθε βήμα.

3.2 Decompose

Με τη χρήση της decompose, αυξάνουμε την ευελιξία της πρόσθεσης αφού η μορφή του suppression που θα χρησιμοποιηθεί στο BDD μεταφέρεται στη συνάρτηση αυτή και η πρόσθεση γίνεται με τον ίδιο τρόπο σε κάθε περίπτωση ανεξάρτητα με το suppression του BDD. Έτσι, ανάλογα τη μορφή suppression που έχουμε για το BDD, επιστρέφει και το αντίστοιχο ζεύγος κόμβων.

Στην αρχική περίπτωση όπου έχουμε εφαρμόσει equal suppression στο BDD, η decompose επιστρέφει το ζεύγος κόμβων (ψ, ψ), όπως φαίνεται στον αλγόριθμο 3.

Algorithm 3 Decompose στην περίπτωση του equal suppression.

Decompose(ψ, v**) {**
    if var$(\psi) = v$**:**
        return ψ_0, ψ_1
    else:
        return ψ, ψ
}

Αν έχουμε Zero Suppressed BDD, λαμβάνοντας υπόψη ότι ο κόμβος που έχει απαλειφθεί είχε μηδενικό γείτονα-1, τότε η decompose επιστρέφει το ζεύγος κόμβων $(\psi, 0)$, όπως φαίνεται στον αλγόριθμο 4.

Algorithm 4 Decompose στην περίπτωση του zero suppression.

Decompose(ψ, v**) {**
    if var$(\psi) = v$**:**
        return ψ_0, ψ_1
    else:
        return $\psi, 0$
}

Τέλος, στην περίπτωση που στο BDD εφαρμόζουμε one suppression, ο

κόμβος που έχει απαλειφθεί είχε μηδενικό γείτονα-0. Έτσι, η decompose επιστρέφει το ζεύγος κόμβων $(0, \psi)$, όπως φαίνεται στον αλγόριθμο 5.

Algorithm 5 Decompose στην περίπτωση του one suppression.

Decompose(ψ, v**) {**
    if var$(\psi) = v$**:**
        return ψ_0, ψ_1
    else:
        return $0, \psi$
}

3.3 Εισαγωγή νέου κόμβου

Κατά τη δημιουργία ενός νέου κόμβου στο BDD και σύμφωνα με τον κανόνα της μείωσης, όταν οι δύο γείτονες του κόμβου είναι ίδιοι, επιστρέφεται ο ένας από αυτούς και δε δημιουργείται νέος κόμβος, όπως φαίνεται και στον κώδικα του αλγορίθμου 6.

Algorithm 6 Δημιουργία κόμβου με equal suppression.

Node(a, b, v**) {**
    if $a = b$**:**
        return a
    else:
        return NewNode(a, b, v**)**
}

Με τη χρήση όμως zero suppression, θα πρέπει να μετατραπεί ανάλογα και ο κανόνας δημιουργίας των κόμβων. Όπως φαίνεται και στον αλγόριθμο 7, ένας νέος κόμβος απαλείφεται όταν ο γείτονας-1 είναι ίσος με μηδέν.

Αντίστοιχα, χρησιμοποιώντας one suppression στο BDD, ένας νέος κόμβος θα εισαχθεί στο διάγραμμα μόνο αν ο γείτονας-0 δεν είναι μηδέν, διαφορετικά θα απαλειφθεί, όπως φαίνεται στον αλγόριθμο 8.

Algorithm 7 Δημιουργία κόμβου με zero suppression.

```
Node(a, b, v) {
    if b = 0:
        return a
    else:
        return NewNode(a, b, v)
}
```

Algorithm 8 Δημιουργία κόμβου με one suppression.

```
Node(a, b, v) {
    if a = 0:
        return a
    else:
        return NewNode(a, b, v)
}
```

Στους παραπάνω αλγορίθμους, η συνάρτηση *NewNode*, χρησιμοποιείται για την εισαγωγή ενός νέου κόμβου μέσα στο BDD με γείτονες τα a και b και τιμή v.

3.4 Μεταβλητό suppression

Κατά την εισαγωγή κόμβων στο δέντρο, η χρήση ενός μοναδικού είδους BDD σε ολόκληρο το γράφο, ίσως να μη μας δίνει τα βέλτιστα αποτελέσματα. Για να καταφέρουμε να έχουμε μεγαλύτερη απόδοση του αλγορίθμου, θα πρέπει δυναμικά κατά τη δημιουργία του δέντρου, να αποφασίζουμε ποια μορφή BDD μας ευνοεί ανάλογα τον κόμβο που βρισκόμαστε και να την εφαρμόζουμε σε αυτόν. Όμως, επειδή αυτό απαιτεί σάρωση ολόκληρου του γράφου και αυτό κοστίζει αλγοριθμικά, θα ήταν καλή σκέψη να εφαρμόζεται όχι σε κάθε εισαγωγή κόμβου, αλλά μόνο ανά τακτά χρονικά διαστήματα, όποτε κρίνουμε ότι υπάρχουν σημαντικές αλλαγές στη δομή του δέντρου μας.

Ο τρόπος που θα γίνει αυτό και για να είμαστε σε θέση να γνωρίζουμε ποια μορφή θα πρέπει να εφαρμόσουμε σε κάποιο κόμβο, είναι να σαρώσουμε ολόκληρο το δέντρο και να κρατάμε σε ένα πίνακα το πλήθος της κάθε μορφής suppression (equal, zero, one και καθόλου) στους κόμβους του για κάθε μεταβλητή του δέντρου, όπως επίσης και το πλήθος των κόμβων που έχουν απαλειφθεί. Έτσι, παρατηρώντας τον πίνακα αυτόν, μπορούμε εύκολα να αποφασίσουμε πλέον ποια μορφή suppression θα εφαρμόσουμε σε κάθε κόμβο ώστε να έχουμε τα καλύτερα δυνατά αποτελέσματα για μικρότερο όγκο πληροφορίας και κατ'επέκταση ταχύτερη εκτέλεση του αλγορίθμου.

Η συνάρτηση που χρησιμοποιείται για να βρει τη μορφή του suppression για κάποια μεταβλητή είναι η *sup*. Έτσι πλέον κατά τη δημιουργία νέου κόμβου, θα πρέπει να ελέγξουμε τον πίνακα και ανάλογα το suppression, να επιλεχθεί από τον αλγόριθμο και ο αντίστοιχος τρόπος δημιουργίας του κόμβου, όπως φαίνεται στον αλγόριθμο 9. Αντίστοιχα διαμορφώνεται και η decompose, όπου πλέον επιλέγεται η μορφή suppression που εφαρμόζεται στον κόμβο που βρίσκεται ο αλγόριθμος. Ο ψευδοκώδικας της νέας συνάρτησης φαίνεται στον αλγόριθμο 10.

Algorithm 9 Δημιουργία κόμβου ανάλογα το suppression.

Node(a, b, v**)** {
    if $sup(v) = equal$ **and** $a = b$**:**
        return a
    if $sup(v) = zero$ **and** $b = 0$**:**
        return a
    if $sup(v) = one$ **and** $a = 0$**:**
        return b
    return NewNode(a, b, v**)**
}

Ένα σημαντικό πλεονέκτημα της συνδυαστικής χρήσης εναλλακτικών

Algorithm 10 Decompose ανάλογα το suppression.

Decompose(ψ, v**) {**
    if var$(\psi) = v$**:**
        return ψ_0, ψ_1
    if $sup(v) = equal$**:**
        return ψ, ψ
    if $sup(v) = zero$**:**
        return $\psi, 0$
    if $sup(v) = one$**:**
        return $0, \psi$
}

μορφών BDDs, είναι ότι στην περίπτωση όπου έχουμε μόνο zero ή one suppression σε μία στήλη του πίνακα, μπορούμε πολύ εύκολα να διαχειριστούμε περιπτώσεις όπου εφαρμόζουμε μια κβαντική πύλη NOT. Όπως είδαμε προηγουμένως, μια πύλη NOT μετατρέπει την κατάσταση $|0\rangle$ σε $|1\rangle$ και αντίστροφα. Άρα μπορούμε εύκολα να διαπιστώσουμε ότι σε τέτοιες περιπτώσεις δε χρειάζεται να σαρώσουμε ολόκληρο το δέντρο για να διαπιστώσουμε τι suppression θα χρησιμοποιήσουμε, αλλά αντ'αυτού απλά αλλάζουμε τη μορφή του suppression από zero σε one και αντίστροφα, ενημερώνοντας τις αντίστοιχες τιμές του πίνακα.

Συμπεραίνουμε ότι, συνδυάζοντας με αυτόν τον τρόπο τις εναλλακτικές μορφές των BDDs που χρησιμοποιήσαμε μέχρι τώρα και εφαρμόζοντας την αντίστοιχη δομή που μας ευνοεί στην εκάστοτε περίπτωση σύμφωνα με τα κριτήρια που αναφέραμε, κάνουμε τον αλγόριθμο μας αποδοτικότερο αφού επιτυγχάνουμε την απαλειφή ακόμα μεγαλύτερου πλήθους κόμβων. Έτσι ελαττώνεται αντίστοιχα το μέγεθος του BDD και ο αλγόριθμος εκτελείται πιο γρήγορα αφού έχει να διαχειριστεί μικρότερο όγκο πληροφορίας.

Κεφάλαιο 4

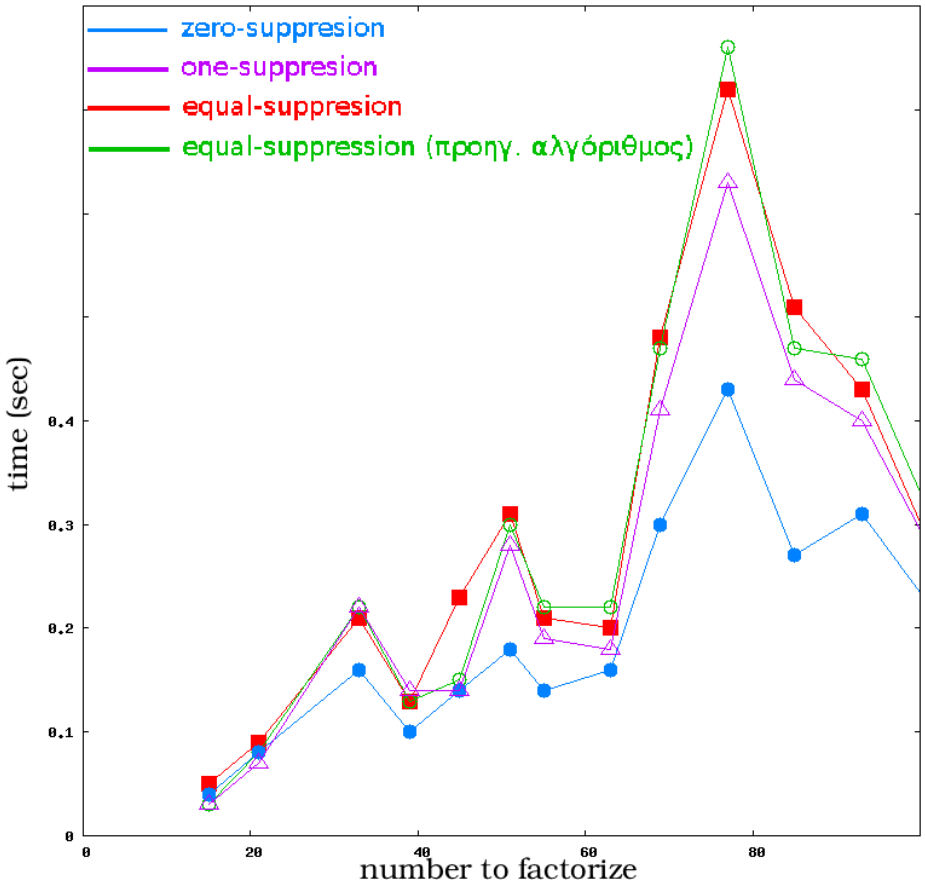
Αποτελέσματα

Στο κεφάλαιο αυτό παρουσιάζονται κάποια πειραματικά αποτελέσματα του αλγορίθμου που περιγράψηκε στο κεφάλαιο 3. Έγιναν μετρήσεις στο χρόνο εκτέλεσης του αλγορίθμου και στη μνήμη που δεσμεύτηκε για κάθε περίπτωση μορφής BDD που χρησιμοποιήσαμε. Η απόδοση του αλγορίθμου εξετάστηκε πάνω στον αλγόριθμο του Shor, για διαφορετικές εισόδους ως αριθμό προς παραγοντοποίηση.

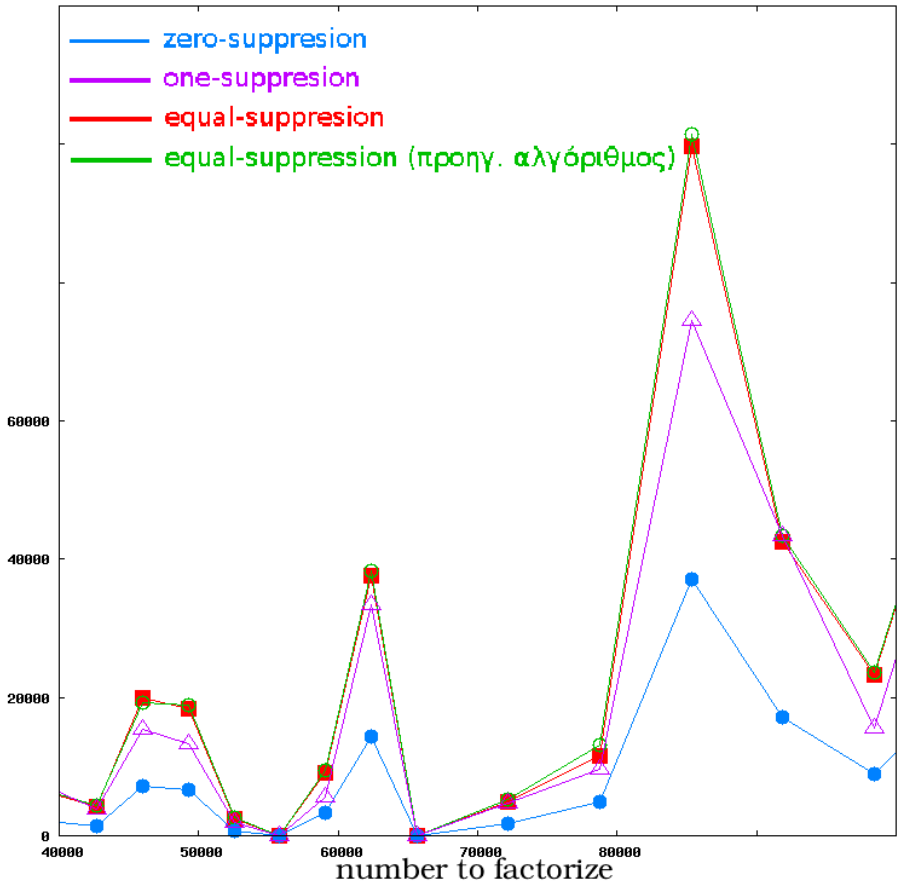
Στα σχήματα 4.1, 4.2, 4.3 και 4.4 παρουσιάζονται τα αποτελέσματα των χρόνων εκτέλεσης του αλγρίθμου σε μορφή γραφικών παραστάσεων για διάφορα πεδία αριθμών εισόδου στον αλγόριθμο του Shor.

Εκτός του χρόνου εκτέλεσης, έγιναν μετρήσεις και για τη μνήμη που χρησιμοποιεί κάθε φορά ο αλγόριθμος, η οποία μεταφράζεται σε αριθμό κόμβων που χρησιμοποιεί. Έτσι στα σχήματα 4.5, 4.6, 4.7 και 4.8 παρουσιάζονται και τα αποτελέσματα του αλγορίθμου ως προς τον αριθμό των κόμβων που χρησιμοποίησε σε μορφή γραφικών παραστάσεων για διάφορα πεδία αριθμών εισόδου στον αλγόριθμο του Shor.

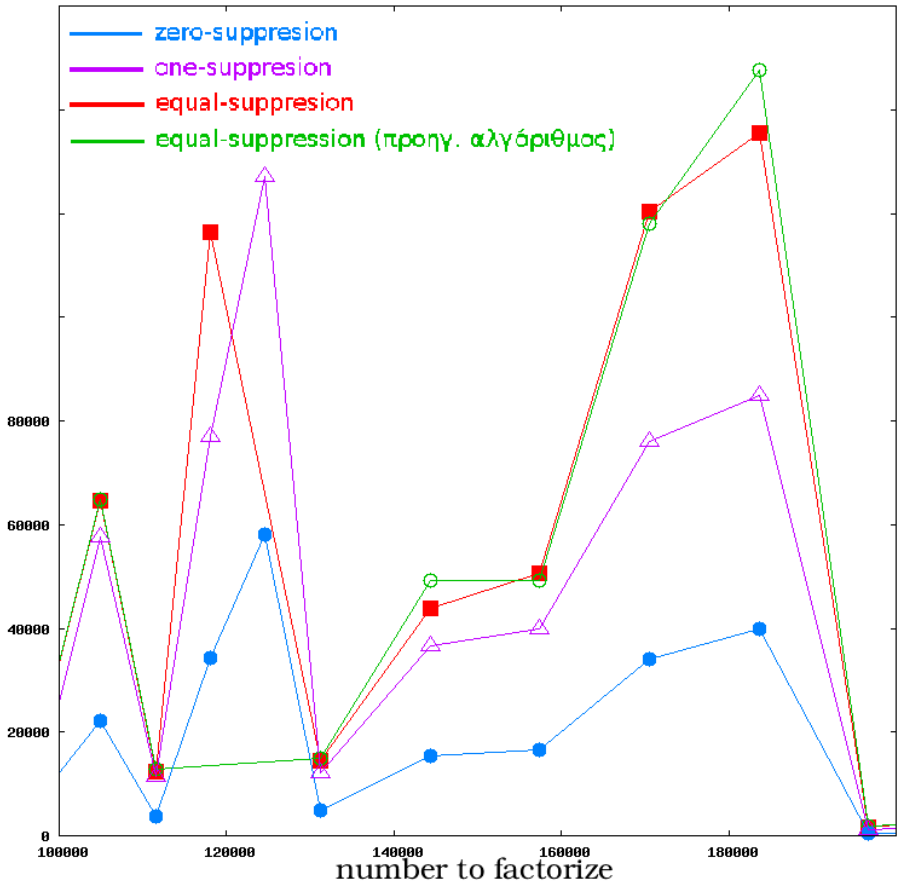
Τέλος, βάλαμε σαν είσοδο στον αλγόριθμο του Shor ένα μεγάλο αριθμό για να παραγοντοποιηθεί και έγινε μέτρηση των κόμβων που χρησιμοποίησε για κάθε περίπτωση BDD σε κάθε βήμα του αλγορίθμου. Τα αποτελέσματα για τους κόμβους εισόδου κάθε βήματος φαίνονται στο σχήμα 4.9,



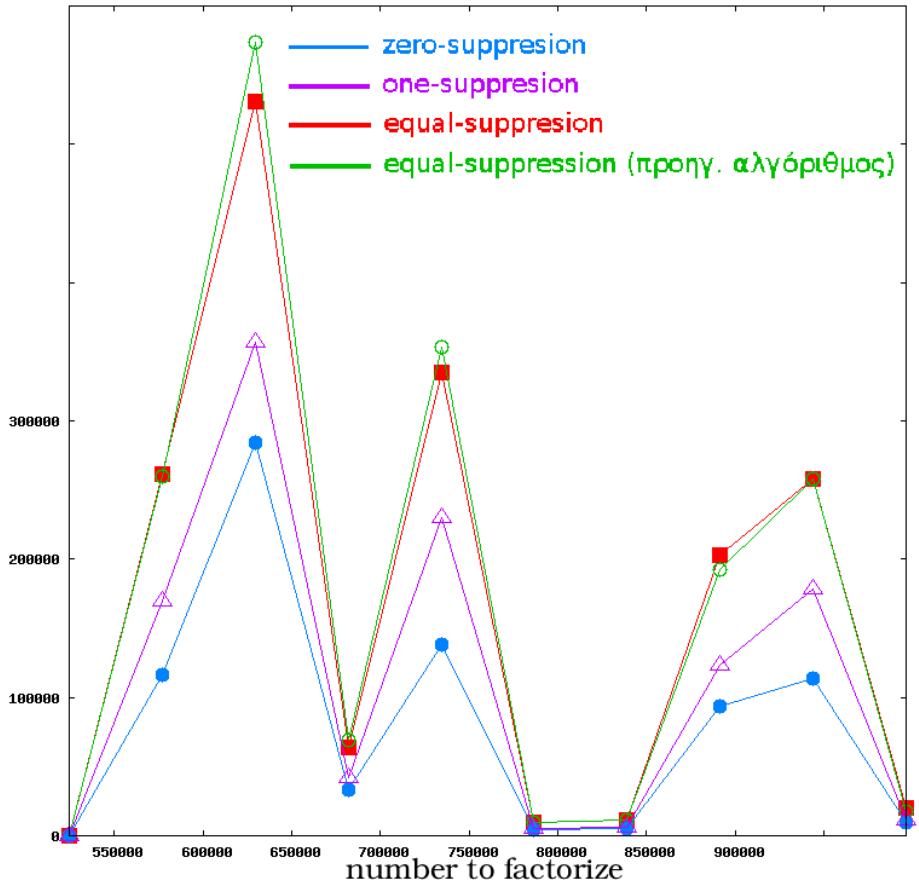
Σχήμα 4.1: Χρόνος εκτέλεσης για παραγοντοποίηση αριθμών από 0 έως 100



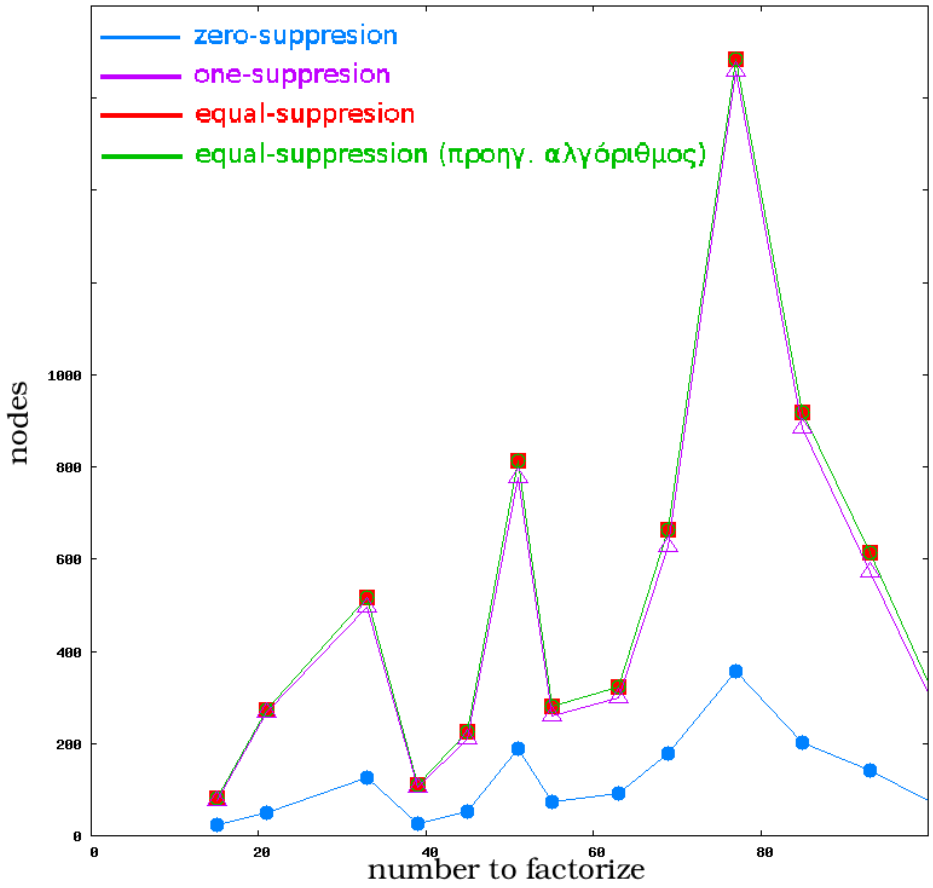
Σχήμα 4.2: Χρόνος εκτέλεσης για παραγοντοποίηση αριθμών από 40000 έως 100000



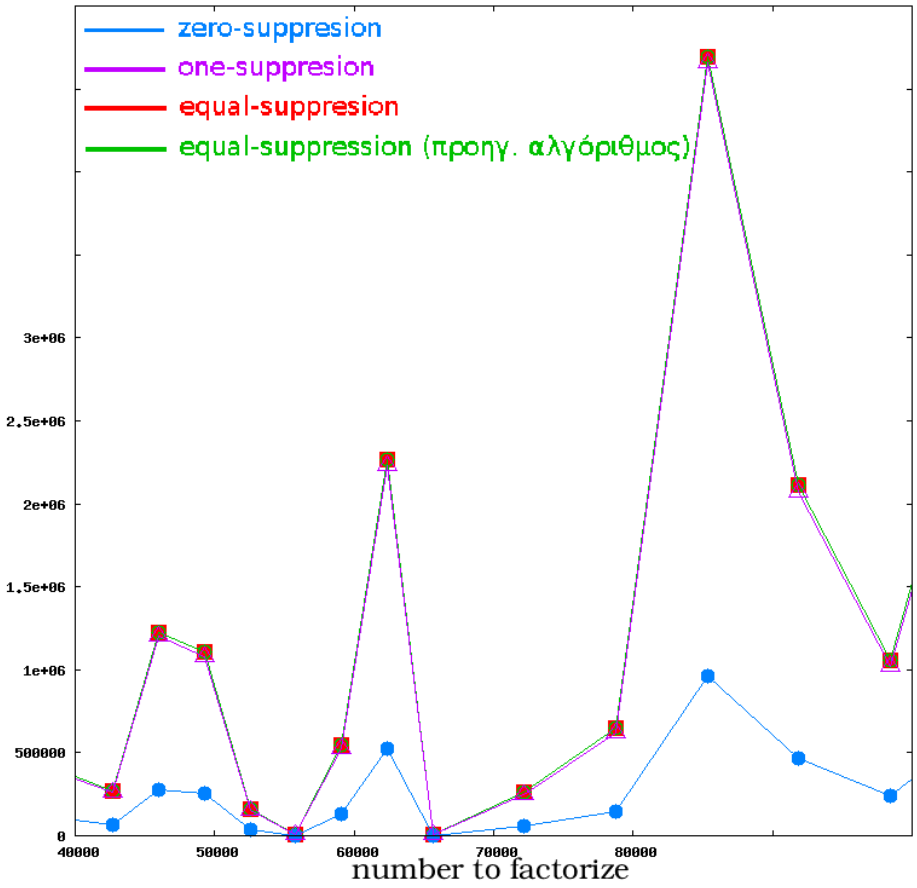
Σχήμα 4.3: Χρόνος εκτέλεσης για παραγοντοποίηση αριθμών από 100000 έως 200000



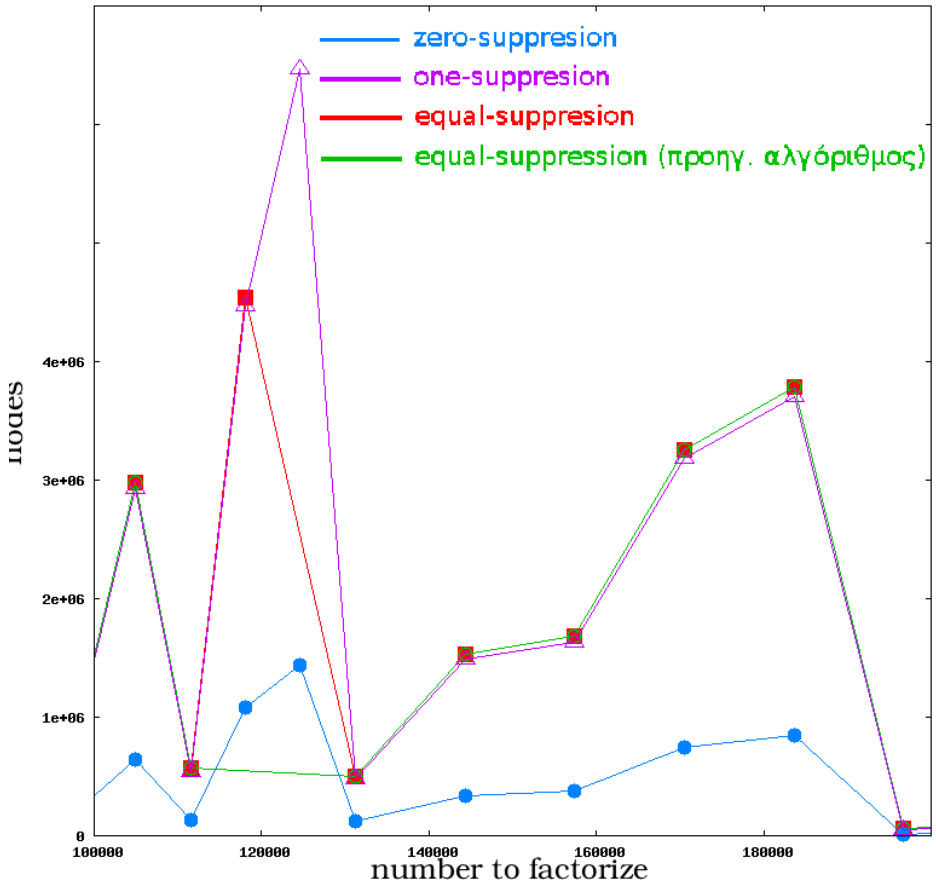
Σχήμα 4.4: Χρόνος εκτέλεσης για παραγοντοποίηση αριθμών από 524433 έως 996303



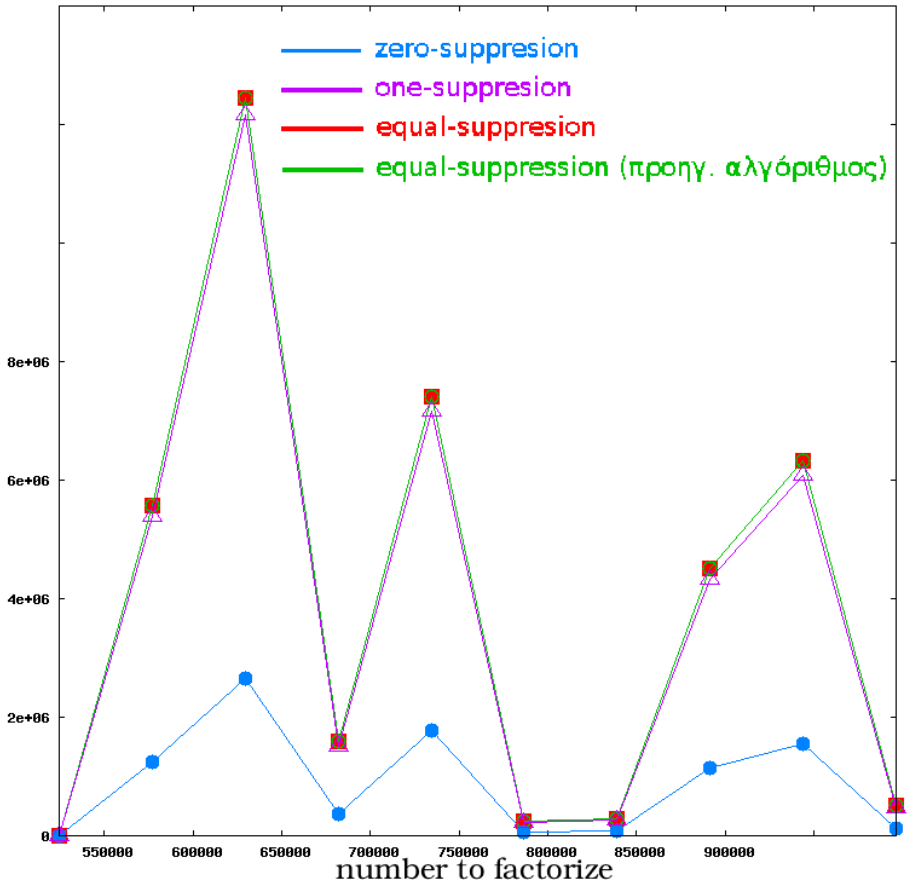
Σχήμα 4.5: Χρήση κόμβων για παραγοντοποίηση αριθμών από 0 έως 100



Σχήμα 4.6: Χρήση κόμβων για παραγοντοποίηση αριθμών από 40000 έως 100000

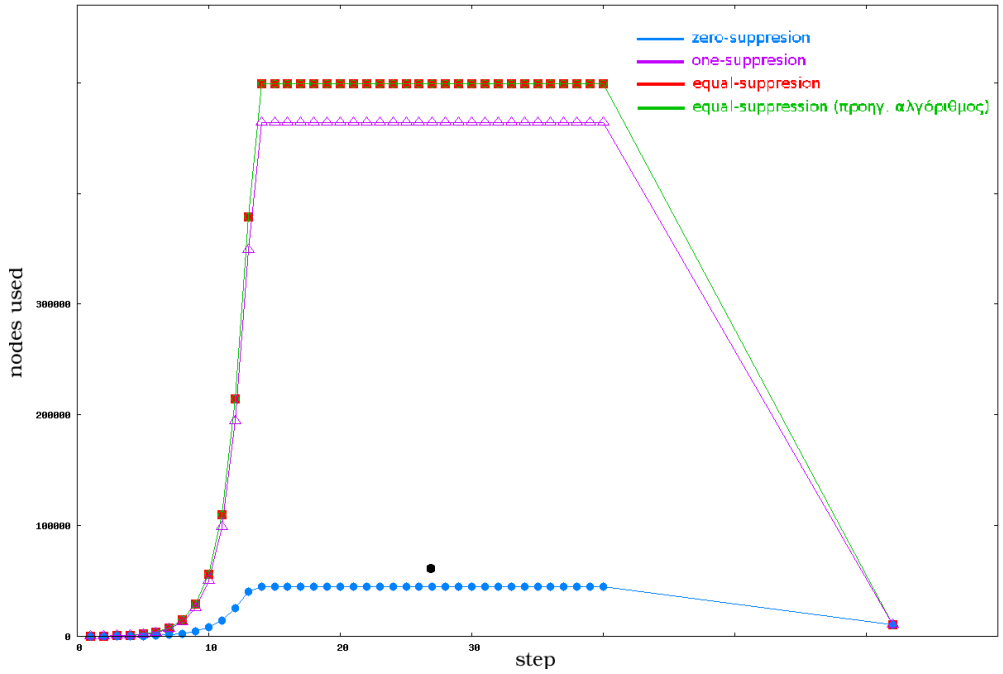


Σχήμα 4.7: Χρήση κόμβων για παραγοντοποίηση αριθμών από 100000 έως 200000



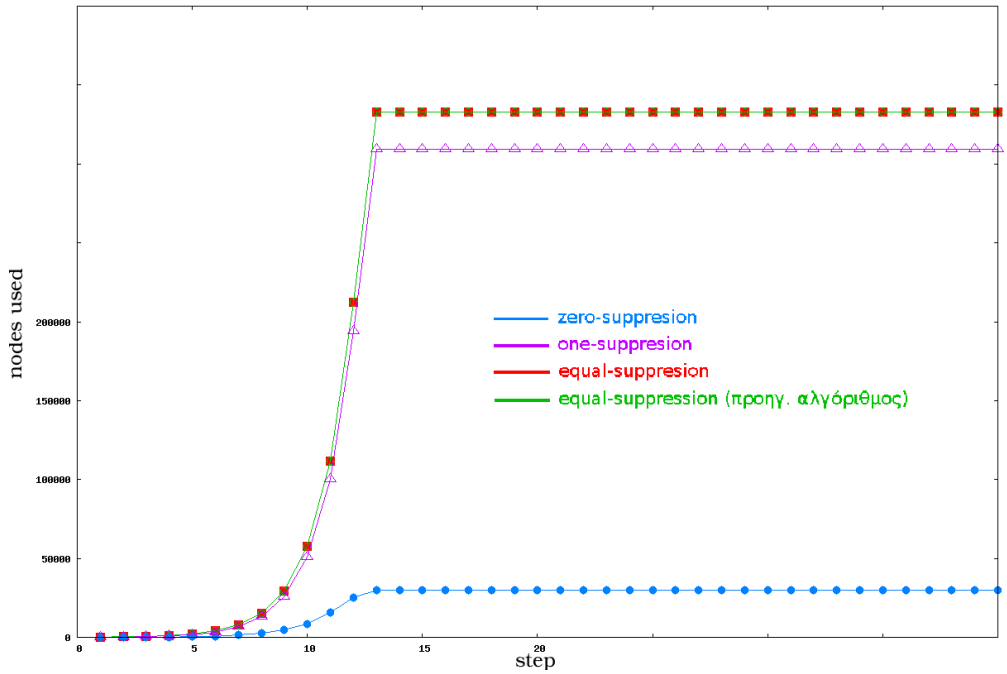
Σχήμα 4.8: Χρήση κόμβων για παραγοντοποίηση αριθμών από 524433 έως 996303

ενώ τα αποτελέσματα για τους κόμβους εξόδου κάθε βήματος φαίνονται στο σχήμα 4.10.



Σχήμα 4.9: Κόμβοι εισόδου για κάθε βήμα του αλγορίθμου

Όπως γίνεται εύκολα αντιληπτό από τα αποτελέσματα, οι εναλλακτικές μορφές δομών BDD, βελτίωσαν αρκετά την απόδοση του αλγορίθμου. Στην περίπτωση του one-suppression, η χρησιμοποιούμενη μνήμη και ο χρόνος εκτέλεσης μειώθηκαν αισθητά σε σχέση με την προηγούμενη υλοποίηση του αλγορίθμου, ενώ στην περίπτωση του zero-suppression, τα νούμερα αυτά μειώθηκαν ακόμα περισσότερο, σημειώνοντας θεαματική βελτίωση στην απόδοση του αλγορίθμου.



Σχήμα 4.10: Κόμβοι εξόδου για κάθε βήμα του αλγορίθμου

Κεφάλαιο 5

Συμπεράσματα

5.1 Ανακεφαλαίωση

Οι κβαντικοί υπολογιστές αποτελούν εκ των πραγμάτων δύσκολο πεδίο έρευνας λόγω της έλειψης υλικού. Όμως, αποτελεί επιτακτική ανάγκη η εξεύρεση αποδοτικών αλγορίθμων προσωμοίωσης κβαντικών κυκλωμάτων, οι οποίοι θα μας βοηθήσουν να κατανοήσουμε τη λειτουργία και να μελετήσουμε την απόδοση κβαντικών υπολογιστών. Στην παρούσα εργασία, βασιζόμενοι στον αλγόριθμο προσωμοίωσης κβαντικών κυκλωμάτων με χρήση συμβολικών αναπαραστάσεων με MTBDDs, προτείναμε τη χρήση εναλλακτικών μορφών BDDs για την αναπαράσταση της πληροφορίας, οι οποίες ήταν τα zero-suppressed BDDs, τα one-suppressed BDDs, καθώς και συνδυασμός τους.

Η αναζήτηση αποδοτικότερων μοντέλων αναπαράστασης αποτελεί μονόδρομο στην προσωμοίωση τέτοιου είδους κυκλωμάτων καθώς η πληροφορία που διαχειριζόμαστε αυξάνεται δραματικά όσο μεγαλύτερο είναι το πλήθος των qubits εισόδου. Τα αποτελέσματα από την πειραματική διαδικασία του αλγορίθμου μας, έδειξαν ότι μειώθηκαν αισθητά οι χρόνοι εκτέλεσής του καθώς και η μνήμη που απαιτείται για την ολοκλήρωση της διαδικασίας. Σε ορισμένες περιπτώσεις μάλιστα, η μείωση αυτή ήταν θεαματική, φτάνοντας σε μεγέθη τάξεις μεγέθους μικρότερα από τα αντίστοιχα

του προηγούμενου αλγορίθμου.

Τα αποτελέσματα που εξάγαμε από την παρούσα εργασία φαίνονται να έχουν μεγάλη θεωρητική και πρακτική αξία. Η βελτίωση της απόδοσης του αλγορίθμου προσωμοίωσης κβαντικών κυκλωμάτων, θα βοηθήσει αρκετά στην περαιτέρω μελέτη των κβαντικών υπολογιστών. Επίσης, η δυνατότητα χρήσης εναλλακτικών μορφών δομών δεδομένων που έχουν τη δυνατότητα να διαχειρίζονται πιο εύκολα πληροφορία που βρίσκονται στις τιμές 0 και 1, μας επιτρέπει την επεξεργασία μεγάλου όγκου δεδομένων που βρίσκεται σε κλασική κατάσταση συνδυασμένη με κβαντική πληροφορία.

5.2 Μελλοντικές επεκτάσεις

Η εργασία αυτή ανοίγει το δρόμο για ακόμα περισσότερες βελτιώσεις των αλγορίθμων κβαντικής προσωμοίωσης, καθώς μας απαλάσσει από την αναπαράσταση των κβαντικών τελεστών με τη μορφή πινάκων. Έτσι μελλοντικές επεκτάσεις του αλγορίθμου, μπορούν να επικεντρωθούν στη χρήση περισσότερο αποδοτικών αναπαράστασεων των διανυσμάτων κατάστασης, καθώς και συνδυασμός τους ανάλογα το είδος και τον όγκο της πληροφορίας που έχουμε να διαχειριστούμε.

Επίσης, θα ήταν ενδιαφέρουσα η μελέτη της προσωμοίωσης κβαντικών κυκλωμάτων με τη χρήση παράλληλης επεξεργασίας για εξαγωγή ακόμα καλύτερων αποτελεσμάτων, καθώς οι τεχνικές που έχουν αναπτυχθεί σχετικά με τις δομές των BDDs, έχουν αποδείξει ότι μπορεί να γίνει κάτι τέτοιο από τη στιγμή που σε κάθε λειτουργία λαμβάνουν μέρος δύο διαφορετικοί γράφοι.

Τέλος απλές δομές δεδομένων που λαμβάνουν μέρος κατά τη διαδικασία εκτέλεσης των αλγορίθμων, θα μπορούσαν να μεταφερθούν και να υλοποιηθούν σε υλικό (για παράδειγμα σε μια FPGA), ώστε να επιτύχουμε

μεγαλύτερη βελτιστοποίηση και ακόμα ταχύτερη εκτέλεση.

Παραπομπές

[1] Vasilis Samoladas Improved BDD algorithms for the simulation of quantum circuits Technical University of Crete, 2008.

[2] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, Fabio Somenzi Algebraic Decision Diagrams and their Applications University of Colorado, 1993

[3] Henrik Reif Andersen An Introduction to Binary Decision Diagrams Technical University of Denmark, 1997

[4] Alan Mishchenko An Introduction to Zero-Suppressed Binary Decision Diagrams Portland State University, 2001

[5] Dirac P. (1958) The Principles Of Quantum Mechanics (4th ed.) Oxford University Press

[6] Fraunhofer Quantum Computing Portal
`http://www.qc.fraunhofer.de`

[7] List of quantum computers simulators
`http://www.quantiki.org/wiki/index.php/List_of_QC_simulators`

[8] Afshin Abdollahi and Massoud Pedram. Analysis and synthesis of quantum circuits by using quantum decision diagrams. In *DATE '06: Proc. of the Conf. on Design, Automation and Test in Europe*, pages

317--322, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[9] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Phys. Rev. A*, 54(2):1034--1063, Aug 1996.

[10] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677--691, 1986.

[11] M. Fujita, P. C. McGeer, and J. C.-Y. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Form. Methods Syst. Des.*, 10(2-3):149--169, 1997.

[12] Shin ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 272--277, New York, NY, USA, 1993. ACM Press.

[13] Christos Koufogiannakis. Techniques for simulating quantum computers. Master's thesis, Technical U. of Crete, 2004.

[14] D. M. Miller, M. A. Thornton, and D. Goodman. Qmdd: A decision diagram structure for reversible and quantum circuits. In *IEEE Int'l Symp. on Multiple-Valued Logic (ISMVL)*, pages 30--30, 2006.

[15] Jayadev Misra. Powerlist: a structure for parallel recursion. *ACM Trans. Program. Lang. Syst.*, 16(6):1737--1767, 1994.

[16] Jumpei Niwa, Keiji Matsumoto, and Hiroshi Imai. General-purpose parallel simulator for quantum computing. *Phys. Rev. A*, 66(6):062317, Dec 2002.

[17] G. Viamontes, I. Markov, and J. Hayes. Improving gate-level simulation of quantum circuits. *Quantum Information Processing*, 2(5):347--380, 2003.

[18] G. F. Viamontes, I. L. Markov, and J. P. Hayes. Graph-based simulation of quantum computation in the density matrix representation. *Quantum Information Processing*, 5(2):113--130, 2005.

[19] George F. Viamontes. Gate-level simulation of quantum circuits. In *Proc. of the 6th Intl. Conference on Quantum Communication, Measurement, and Computing*, pages 311--314, 2002.

[20] George F. Viamontes. *Efficient Quantum Circuit Simulation*. PhD thesis, University of Michigan, 2007.