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

Σχεδιασμός εργαλείου για την εκτίμηση της απόδοσης και της ενέργειας σε προγράμματα μέσω στατικής ανάλυσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Χ.Σαλάπας

Επιβλέπων : Δημήτριος Ι.Σούντρης

Καθηγητής Ε.Μ.Π

Αθήνα, Μάρτιος 2020

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

Σχεδιασμός εργαλείου για την εκτίμηση της απόδοσης και της ενέργειας σε προγράμματα μέσω στατικής ανάλυσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Χ.Σαλάπας

Επιβλέπων : Δημήτριος Ι.Σούντρης
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Μαρτίου 2020.

.....	.....	.....
Δημήτριος Ι.Σούντρης	Νικόλαος Σ.Παπασπύρου	Γεώργιος Ι.Γκούμας
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Επίκουρος Καθηγητής

Αθήνα, Μάρτιος 2020

.....
Κωνσταντίνος Χ.Σαλάπας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια υπάρχει μεγάλο ενδιαφέρον για τον χρόνο εκτέλεσης και ενεργειακή κατανάλωση σε προγράμματα. Ο χρόνος εκτέλεσης είναι ένας παράγοντας που απασχολεί όλους τους κατασκευαστές, καθώς η βιομηχανία απαιτεί εφαρμογές που εκτελούνται με υψηλές ταχύτητες και περιλαμβάνουν πολλές δυνατότητες. Παράλληλα, όσο οι τεχνολογίες γίνονται όλο και πιο πολύπλοκες, οι σχεδιαστές αντιμετωπίζουν το πρόβλημα της αύξησης της κλίμακας των εφαρμογών, κάτι που επιβαρύνει την ενεργειακή κατανάλωση.

Στο πιο βασικό επίπεδο προγραμματισμού, οι σχεδιαστές αναζητούν τρόπους να βελτιώνουν τα παραπάνω μεγέθη διατηρώντας τις δυνατότητες των προγραμμάτων τους. Αυτό απαιτεί μία καλή γνώση του κώδικα και της γλώσσας που χρησιμοποιείται, αλλά και ένα διαρκή έλεγχο για τον εντοπισμό σφαλμάτων και την δημιουργία των αποτελεσμάτων. Μία μέθοδος που χρησιμοποιείται ευρέως είναι οι συνεχόμενες δοκιμές του προγράμματος με διάφορες εισόδους και η μελέτη των χρόνων που χρειάζεται για να δώσει αντίστοιχες εξόδους. Ωστόσο, αυτή η διαδικασία, αν και αρκετά αξιόπιστη, απαιτεί αρκετό χρόνο για να πραγματοποιηθεί και η καλύτερη χρήση της εξαρτάται από τον προγραμματιστή.

Ο κύριος σκοπός της εργασίας είναι να ερευνηθεί η δυνατότητα καθορισμού της απόδοσης του προγράμματος χωρίς την εκτέλεσή του. Έχουν φτιαχτεί μοντέλα που δεδομένης κάποιας γνώσης του κώδικα και χωρίς την εκτέλεση του προγράμματος επιχειρούν να προσφέρουν μία αρκετά ακριβή πρόβλεψη της ζητούμενης εξόδου. Το εργαλείο που επιλέχθηκε, εξετάζει το object αρχείο του προγράμματος και παράγει κάποιες μετρικές που χαρακτηρίζουν τον κώδικα. Τα μοντέλα χρησιμοποιούν τις μετρικές και τον αριθμό των εντολών και προβλέπουν το χρόνο εκτέλεσης και την ενεργειακή απόδοση σε περιβάλλον CPU.

Κάθε μοντέλο επιλέγεται ανάλογα με τον αριθμό πυρήνων που χρησιμοποιούνται στη μνήμη CPU και μπορεί να χρησιμοποιηθεί για προβλέψεις σε προγράμματα που εκτελούνται σε διαφορετικά συστήματα. Οι προβλέψεις γίνονται πάνω σε συγκεκριμένα κομμάτια κώδικα και στο τελικό κομμάτι της διπλωματικής επιχειρείται συνδυάζοντας τις προβλέψεις και εκτιμώντας τις εντολές να εκτιμηθεί ο συνολικός χρόνος και ενέργεια σε ένα πρόγραμμα.

Λέξεις Κλειδιά

Στατική ανάλυση, IACA, Αφηρημένο Συντακτικό Δέντρο, Παλινδρόμηση, CPU, Προσβάσεις στη μνήμη, συμβολική γλώσσα, Ενσωματωμένες πλατφόρμες

Abstract

During the last years there has been a great interest in the execution time and energy consumption of programs. Execution time is a factor that concerns most manufacturers, since the technology industry requires fast applications with many capabilities. Moreover, as the technologies get more complex , the designers face the problem of increasing the scale of the applications, something that comes at the expense of energy consumption.

In the most basic level of programming, designers seek ways to improve these features while preserving most of their programs capabilities. This requires a good understanding of the code and the language that is used and a continuous checking for the detection of errors and the production of results. A method that is broadly used is the continuous checks of an application with different inputs and the analysis of the execution that is required to produce outputs. However, this procedure, although reliable enough, requires some time to be utilized and its best usage is dependent on the skills of the programmer.

The main goal of this study is to examine the capabilities of estimating the performance of the program without executing it. We have constructed models that with some knowledge of the code and without executing the program make the attempt to produce an accurate enough prediction of the requested output. The tool that was selected analyzes the object file of the program and produces some metrics that define the code. The models use these metrics and the number of instructions and predict the execution time and energy consumption in CPU environment.

Each model is selected based on the number of cores that are used in the CPU memory and can be utilized to for predictions in programs that are executed in different systems. The predictions are made upon specific blocks of code and in the final part of this study it is attempted to estimate the execution time and energy consumption in the whole program by combining these predictions and the estimating the total number of instructions.

.

Key words

Static Analysis, IACA , AST, Regression, CPU, Memory accesses, Assembly language, Embedded Platforms

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή Δημήτριο Σούντρη για την ευκαιρία που μου έδωσε να εκπονήσω την διπλωματική εργασία στο συγκεκριμένο εργαστήριο, καθώς και για τις οδηγίες και συμβουλές που μου προσέφερε κατά τη διάρκεια των σπουδών μου.

Επιπλέον, ευχαριστώ τον υποψήφιο διδάκτορα Χαράλαμπο Μαράντο και τον διδάκτορα ερευνητή Λάζαρο Παπαδόπουλο για τη βοήθεια και την καθοδήγηση που μου προσέφεραν κατά τη διάρκεια της διπλωματικής μου εργασίας.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τα παραπάνω πρόσωπα που με οδήγησαν στην επιλογή αυτής της διπλωματικής, καθώς απεδείχθη ότι είχε πολύ ενδιαφέρον και αποτελεί ένα αντικείμενο που μπορεί να με απασχολήσει στο μέλλον.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου, ανθρώπους που με έχουν υποστηρίξει όλα αυτά τα χρόνια και μου έχουν μάθει να προσπαθώ πάντα περισσότερο.

Πίνακας περιεχομένων

Κεφάλαιο 5

Κεφάλαιο 6

Κεφάλαιο 7

Κατάλογος Σχημάτων

Σχεδιασμός εργαλείου για την εκτίμηση της απόδοσης και της ενέργειας σε προγράμματα με χρήση στατικής ανάλυσης

Κεφάλαιο 1

Εισαγωγικές έννοιες

Τις τελευταίες δεκαετίες έχει σημειωθεί μία ραγδαία αύξηση στις απαιτήσεις του γενικού κοινού για ηλεκτρονικές εφαρμογές. Χαρακτηριστικό παράδειγμα αποτελούν τα κινητά τηλέφωνα που μέσα σε λίγα χρόνια έχουν εξελιχθεί από την χρήση πλήκτρων σε οθόνες αφής και μεταβάλλονται διαρκώς για να προσφέρουν περισσότερες εφαρμογές σε υψηλότερες ταχύτητες. Ωστόσο, η τεχνολογία βρίσκεται τώρα σε ένα σημείο που υποχρεώνεται να ικανοποιεί κάποιες δυνατότητες σε βάρος κάποιων άλλων. Στο συγκεκριμένο παράδειγμα, η μεγάλη παροχή εφαρμογών και χώρου για την αποθήκευσή τους οδηγεί την πολύ γρήγορη κατανάλωση της μπαταρίας των τηλεφώνων σχετικά με πριν μία δεκαετία. Επίσης, η συνεχής επαναφόρτιση των μπαταριών οδηγεί στην επιδείνωση της μακροχρόνιας απόδοσής τους.

Για αυτό το λόγο, η εξισορρόπηση των πόρων που καταναλώνουν οι εφαρμογές είναι ένα ζήτημα που η σημασία του έχει αυξηθεί ραγδαία τα τελευταία χρόνια. Προφανώς, ο χρόνος εκτέλεσης είναι ένας παράγοντας που μελετάται σε κάθε επίπεδο προγραμματισμού. Ο κύριος σκοπός του προγραμματιστή είναι να παράγει κατανοητό κώδικα που εκτελείται χωρίς σφάλματα και σε λογική ταχύτητα. Σε αρκετές περιπτώσεις ο προγραμματιστής μπορεί να αξιολογήσει την απόδοση της εφαρμογής του με βάση τη λογική της και την κατανόηση των μεθόδων και δομών που χρησιμοποιεί. Ωστόσο, υπάρχουν λεπτομέρειες που είναι δύσκολο να εντοπιστούν και απαιτείται ο προγραμματιστής να κάνει δοκιμές στο πρόγραμμα για να αποκτήσει εικόνα για την ταχύτητα και τα σφάλματα.

Σε επίπεδο hardware, πέρα από τον χρόνο εκτέλεσης, εξετάζονται κι άλλοι παράγοντες που έχουν σχέση με το σύστημα, όπως η κατανάλωση ισχύος και μνήμης. Οι τεχνολογίες που χρησιμοποιούνται έχουν περιορισμένες δυνατότητες και διαφορετικά συστήματα έχουν διαφορετικά πλεονεκτήματα και αδυναμίες. Αυτό σημαίνει ότι όταν υπάρχει βελτίωση σε κάποια χαρακτηριστικά θα υπάρχει επιβάρυνση σε άλλα. Στόχος των μηχανικών είναι η εξισορρόπηση των διαφόρων παραγόντων ανάλογα με τους σκοπού της εφαρμογής και αυτό απαιτεί τη γνώση της τιμής των παραγόντων κατά την διάρκεια εκτέλεσης.

Φαίνεται ότι και σε επίπεδο λογισμικού και hardware, οι προγραμματιστές πρέπει να κάνουν αρκετές δοκιμές στο εκτελέσιμο πρόγραμμά τους για να προσδιορίσουν τα σημεία που χρειάζονται αλλαγές. Η μέθοδος που χρησιμοποιείται κυρίως για την εύρεση των παραγόντων είναι η δυναμική προγραμματιστική ανάλυση.

1.1 Δυναμική προγραμματιστική ανάλυση

Δυναμική ανάλυση είναι η ανάλυση μίας εφαρμογής κατά την εκτέλεσή της πάνω σε πραγματικούς ή εικονικούς επεξεργαστές. Η διαδικασία της περιλαμβάνει την προετοιμασία δεδομένων εισόδου, την εκτέλεση ενός προγράμματος δοκιμής, την συγκέντρωση των απαραίτητων παραμέτρων και την ανάλυση των δεδομένων εξόδου. Τα αποτελέσματα μπορεί να δώσουν πληροφορία για την απόδοση του προγράμματος, δηλαδή τους πόρους που έχουν καταναλωθεί, για την πολυπλοκότητα του κώδικα και για λογικά σφάλματα και αδυναμίες. Για να παραχθούν σωστά αποτελέσματα απαιτούνται αρκετές δοκιμές του προγράμματος με διαφορετικές εισόδους. Το σύνολο εισόδων πρέπει να είναι προσεκτικά επιλεγμένο έτσι ώστε να καλύπτει τις διάφορες περιπτώσεις που η εκτέλεση ακολουθεί διαφορετικό μονοπάτι. Όταν καλύπτονται οι περισσότερες περιπτώσεις τα στοιχεία που εντοπίζονται από την ανάλυση είναι γενικά γνήσια.

Για αυτούς τους λόγους, η δυναμική ανάλυση χρησιμοποιείται συχνά μετά την κατασκευή του κώδικα για τον υπολογισμό των πόρων που καταναλώνει και τον σφαλμάτων που περιέχει. Πολλά περιβάλλοντα ανάπτυξης περιλαμβάνουν εργαλεία δυναμικής ανάλυσης ως μονάδες τους με τη μορφή debuggers και profilers. Από εργαλεία open source, πολύ γνωστό είναι το valgrind [1] που χρησιμοποιείται για τον εντοπισμό σφαλμάτων που έχουν σχέση με τη μνήμη, όπως τα memory leaks.

Ωστόσο, η δυναμική ανάλυση έχει μερικά ελαττώματα και το κύριο μειονέκτημα αποτελεί η χρησιμοποίηση πολλών υπολογιστικών πόρων. Επίσης απαιτεί την εκτέλεση του προγράμματος σε συγκεκριμένη πλατφόρμα ή σε simulators, που έχουν αργή ταχύτητα. Το γεγονός αυτό δε διευκολύνει ούτε την άμεση μέτρηση της απόδοσης στο χρόνο σχεδιασμού, ούτε την δοκιμή εναλλακτικών πλατφορμών και δεν επιτρέπει στις δυναμικές μεθόδους να γίνουν μέρος ενός προγράμματος SDK(software development kit). Για αυτό το λόγο οι προγραμματιστές χρησιμοποιούν πιο οικονομικές μεθόδους για τον εντοπισμό κάποιων σφαλμάτων.

1.2 Στατική προγραμματιστική ανάλυση

Στατική ανάλυση είναι η ανάλυση του λογισμικού που πραγματοποιείται χωρίς την εκτέλεση των προγραμμάτων σε αντίθεση με την δυναμική ανάλυση [2][3]. Στις περισσότερες περιπτώσεις εφαρμόζεται πάνω στον πηγαίο κώδικα και σε κάποιες άλλες πάνω σε κάποια μορφή του object αρχείου. Αν και οι προγραμματιστές μπορούν να παρατηρήσουν προβλήματα με μία καλή κατανόηση του κώδικα, αυτή η ανάλυση πραγματοποιείται με τη χρήση αυτοματοποιημένων εργαλείων που μπορούν να προσφέρουν περισσότερες λεπτομέρειες.

Οι πληροφορίες που προσφέρει ποικίλουν από την εύρεση πιθανών σφαλμάτων μέσα στον κώδικα στη δημιουργία επίσημων μεθόδων που αποδεικνύουν μαθηματικά τις ιδιότητες ενός προγράμματος. Η χρήση μαθηματικών μεθόδων περιλαμβάνει για παράδειγμα την αφηρημένη ερμηνεία, η οποία μοντελοποιεί το αποτέλεσμα που κάθε δήλωση έχει στην κατάσταση της αφηρημένης μηχανής, δηλαδή στην εκτέλεση του λογισμικού με βάση τις μαθηματικές ιδιότητες κάθε δήλωσης. Οι συμπεριφορές του συστήματος προσεγγίζονται από την αφηρημένη μηχανή και αν και αυτό δεν καλύπτει απαραίτητα κάθε περίπτωση, η καλή του χρήση κάνει πιο απλό το σύστημα και δίνει αρκετά αντιπροσωπευτικά αποτελέσματα [4].

Από τα εργαλεία στατικής ανάλυσης γνωστό είναι το SonarQube και η αντίστοιχη online έκδοσή του το SonarCloud. Τα εργαλεία μπορούν εξετάζοντας το κώδικα να βρουν πληροφορίες που βοηθούν στην εξασφάλιση της διατηρισιμότητας και αξιοπιστίας. Πιο συγκεκριμένα, από άποψη αξιοπιστίας εξετάζουν για σφάλματα που μπορεί να εμποδίζουν τη σωστή εκτέλεση του προγράμματος, όπως όταν σε μία πράξη χρησιμοποιείται μεταβλητή που δεν έχει αρχικοποιηθεί. Επίσης, εξετάζουν για αδυναμίες στον κώδικα που μειώνουν την ασφάλεια του και για αχρείαστα κομμάτια κώδικα που κάνουν πιο εκτενή τη μορφή του και πρέπει να αφαιρεθούν για να βελτιωθεί η διατηρισιμότητά του. Γενικά η στατική ανάλυση δίνει κάποιες μετρικές για την πολυπλοκότητα του κώδικα που βελτιώνονται όσο βελτιώνονται τα παραπάνω χαρακτηριστικά.

Όπως φαίνεται η στατική ανάλυση προσφέρει πληροφορίες που μπορούν να βοηθήσουν στη διατήρηση της καλής ποιότητας μίας εφαρμογής. Για αυτό το λόγο, χρησιμοποιείται από κρίσιμα και πολύπλοκα συστήματα, όπως το φαρμακευτικό [5] και πυρηνικό λογισμικό [6]. Επιπλέον, η αποτελεσματική χρήση πόρων για την εκτέλεσή της έχει οδηγήσει σε μία αυξανόμενη χρήση της από προγραμματιστές σε όλο τον κόσμο [7]. Σε αυτήν την διπλωματική γίνονται βήματα για εφαρμογή σε run time μεγέθη όπως η κατανάλωση ενέργειας.

1.3 Σκοπός

Στόχος της διπλωματικής είναι να ερευνήσει για μεθόδους που δίνουν πληροφορίες για την απόδοση μίας εφαρμογής με οικονομικό και αποτελεσματικό τρόπο από άποψη κατανάλωσης πόρων. Για αυτό το λόγο επιχειρείται να γίνει χρήση της στατικής ανάλυσης που χρησιμοποιεί πολύ λιγότερους πόρους σε σχέση με τη δυναμική. Με τη χρήση κάποιων μετρικών που προκύπτουν από στατική ανάλυση και κάποιων χαρακτηριστικών που συνδέονται με το μέγεθος του προγράμματος θα γίνουν κάποιες εκτιμήσεις και εξακρίβωση της ακρίβειάς των τελευταίων.

Κεφάλαιο 2

Περιγραφή προβλήματος

Για τους σκοπούς της διπλωματικής έχει γίνει μία έρευνα για το τι είδους εργαλεία στατική ανάλυσης θα χρησιμοποιηθούν. Αυτό που επιλέχθηκε είναι ένα open-source εργαλείο της Intel που ονομάζεται Architectural Code Analyzer. Το εργαλείο αυτό δέχεται ένα object αρχείο ενός προγράμματος και παράγοντας κώδικα assembly υλοποιεί πάνω του στατική ανάλυση για να συγκεντρώσει κάποιες μετρικές. Αυτές οι μετρικές προκύπτουν είτε από συγκέντρωση συγκεκριμένων στοιχείων από τον κώδικα assembly είτε από εκτιμήσεις που το εργαλείο κάνει πάνω στον τύπο των εντολών. Τα προγράμματα που χρησιμοποιούνται είναι γραμμένα κυρίως σε C και C++, που αποτελούν υψηλού επίπεδου γλώσσες που χρησιμοποιούνται σε επίπεδο επεξεργαστή. Ο μεταγλωττιστής gcc χρησιμοποιείται χωρίς να υπάρχουν ειδικές σημαίες.

Οι μετρικές που παράγονται από την εκτέλεση του εργαλείου χρησιμοποιούνται ως χαρακτηριστικά (features) σε μία πρόβλεψη. Αυτή η πρόβλεψη γίνεται από κάποια μονάδα που χρησιμοποιείται στη μηχανική μάθηση και τα στοιχεία που αναζητούνται ως έξοδοι είναι κυρίως ο χρόνος και εν μέρει η ενέργεια. Για να γίνει ικανή πρόβλεψη χρειάζεται ένα ακόμα feature, οι συνολικές εντολές του προγράμματος, που δίνουν μία εικόνα για το μέγεθος του προγράμματος και σχετίζονται με την απόδοσή του. Θεωρητικά ο προγραμματιστής θα μπορούσε να μην χρησιμοποιήσει προβλέψεις για τον χρόνο εκτέλεσης σε επίπεδο μικροεπεξεργαστών. Με ανάλυση του κώδικα assembly θα έβρισκε τα ποσοστά συγκεκριμένων εντολών και με τις εντολές θα υπολόγιζε τον συνολικό αριθμό τους. Προσδίδοντας συγκεκριμένους χρόνους σε κάθε εντολή και αθροίζοντας τους θα έκανε μία προσέγγιση του πραγματικού χρόνου εκτέλεσης του προγράμματος. Ωστόσο, εκτιμήσεις μπορούν να δοθούν μόνο σε πολύ απλούς επεξεργαστές, καθώς σε πιο περίπλοκους υπάρχουν διαρκώς διεργασίες που τρέχουν στο σύστημα, οπότε ο υπολογισμός των χρόνων εκτέλεσης δεν είναι ακριβής.

Για την εκπαίδευση του μοντέλου χρησιμοποιείται ένα σύνολο από τυχαία συνθετικά προγράμματα που έχουν παραχθεί από μία συνάρτηση παραγωγής τυχαίων προγραμμάτων. Η μορφή τους θυμίζει απλά σώματα επαναλήψεων που κάνουν αριθμητικές πράξεις. Για σύνολο δοκιμής (test set) χρησιμοποιείται είτε ένα υποσύνολο του συνόλου συνθετικών προγραμμάτων είτε ένα σύνολο από υπάρχοντα προγράμματα.

Εκτός του παραπάνω μοντέλου χρησιμοποιείται και ένα μοντέλο που χρησιμοποιεί ως χαρακτηριστικό τις εντολές σε ένα συγκεκριμένο τμήμα του κώδικα αντί για τις συνολικές, δηλαδή αγνοεί φαινόμενα επαναλήψεων. Σε αυτό προκύπτει μία πρόβλεψη για το συγκεκριμένο κομμάτι κώδικα και αναλογικά χρησιμοποιώντας τις συνολικές εντολές υπολογίζονται οι έξοδοι για τον ευρύτερο κώδικα. Και σε αυτό το μοντέλο χρησιμοποιούνται τα ίδια σύνολα ως training και test σύνολα.

Τα δύο μοντέλα εξετάζονται και επιλέγεται εκείνο που οδηγεί σε καλύτερη πρόβλεψη. Σε κάθε μοντέλο για την δημιουργία της καλύτερης πρόβλεψης εξετάζονται διάφορες μέθοδοι και επιλέγεται εκείνη που δίνει τις καλύτερες μετρικές από άποψη ακρίβειας και σφάλματος. Πιο συγκεκριμένα, επιλέγεται μίας μέθοδος με βάση τις προβλέψεις πάνω στο υποσύνολο των τυχαίων προγραμμάτων και μετά μελετάται η ακρίβεια της πάνω σε πραγματικά προγράμματα. Από τα δύο μοντέλα συγκρίνονται τα αποτελέσματα με την μέθοδο που έχει επιλεχθεί στο καθένα και επιλέγεται το μοντέλο με την καλύτερη πρόβλεψη.

Τα υπάρχοντα προγράμματα έχουν εκτελεστεί πάνω σε μία πλακέτα που υποστηρίζει και λειτουργίες για την εύρεση ενέργειας πέρα από τον χρόνο. Οι μετρικές της απόδοσης που προκύπτουν χρησιμοποιούνται ως μέτρο σύγκρισης για τις εξόδους από την πρόβλεψη. Ωστόσο, τα προγράμματα είναι γραμμένα σε γλώσσα που μπορεί να εκτελεστεί και σε διαφορετικά περιβάλλοντα πλακετών. Αυτό σημαίνει ότι το μοντέλο πρόβλεψης μπορεί να κάνει προβλέψεις για τους χρόνους εκτέλεσης και σε άλλες πλακέτες.

Με τη χρήση του μοντέλου μπορεί να γίνει η πρόβλεψη της συνολικής απόδοσης του προγράμματος υπολογίζοντας τα χαρακτηριστικά σε συγκεκριμένα κομμάτια κώδικα και συνδυάζοντας τις προβλέψεις για μία εκτίμηση του συνολικού αποτελέσματος. Από τα πιο σημαντικά κομμάτια του κώδικα είναι τα σώματα επαναλήψεων καθώς αυτά καταναλώνουν τους περισσότερους πόρους κατά την εκτέλεσή τους. Από τα προγράμματα που χρησιμοποιούνται για την εκπαίδευση του μοντέλου τα κομμάτια που εξετάζονται είναι σώματα επαναλήψεων. Εκτός από αυτό σημαντικές είναι και οι συναρτήσεις και τα σημεία που γίνεται η κλήση τους καθώς αυτό αυξάνει τις εντολές που εκτελούνται σε ένα τμήμα του κώδικα.

Αν ο προγραμματιστής ακολουθεί μία συγκεκριμένη μεθοδολογία μπορεί να κάνει μία εκτίμηση των εντολών ή να θέσει κάποια όρια στις τιμές που μπορούν να φτάσουν. Αυτό σημαίνει ότι θα γνωρίζει τις υψηλότερες και χαμηλότερες τιμές που μπορούν να πάρουν και να κάνει μία πρόβλεψη για τη χειρότερη και καλύτερη περίπτωση. Έτσι εξοικονομείται χρόνος και από τον υπολογισμό των εντολών που είναι μία διαδικασία που απαιτεί την εκτέλεση του προγράμματος τουλάχιστον μία φορά.

Κεφάλαιο 3

Σχετική Βιβλιογραφία

Οι Bazzaz Mostafa κ.ά [8] έχουν ακολουθήσει μία προσέγγιση βασισμένη σε μετρήσεις για τον κατασκευή ενός μοντέλου που κάνει εκτίμηση της ενεργειακής κατανάλωσης πάνω σε ενσωματωμένα συστήματα. Η υλοποίηση του μοντέλου έχει γίνει σε περιβάλλοντα μόνο μικροελεγκτών, δηλαδή οι εντολές που εξετάζονται είναι σε επίπεδο assembly και βασίζεται πάνω σε ένα συγκεκριμένο σύνολο εντολών. Το μοντέλο αυτό έχει υποστηριχθεί από την φυσική εφαρμογή του πάνω σε ένα σύνολο προγραμμάτων. Τα μεγέθη που έχουν υπολογιστεί είναι η κατανάλωση ενέργειας για τις μνήμες Flash και SRAM και έχει η ενέργεια για τα επίπεδα pipeline κάθε εντολής. Το μοντέλο που χρησιμοποιεί είναι regression πάνω στις παραμέτρους, οι οποίες αποτελούν ένα σύνολο εντολών assembly και για την εκπαίδευσή του έχει χρησιμοποιηθεί ένα μικρό σύνολο δεδομένων 60 προγραμμάτων. Το μοντέλο της διπλωματικής διαφέρει στο ότι δε γίνεται εκτίμηση της απόδοσης κάθε εντολής ξεχωριστά και χρησιμοποιείται μεγαλύτερο αλλά όχι κορεσμένο σύνολο δεδομένων για να είναι η πρόβλεψη λιγότερη εξαρτημένη από τα δεδομένα. Επίσης γίνεται προσπάθεια για γενικευμένη μέθοδο που θα προσφέρει εκτιμήσεις σε οποιαδήποτε πλατφόρμα ή αρχιτεκτονική.

Οι Zheng Xinnian κ.ά [9]-[10] κατασκευάζουν μοντέλα που χρησιμοποιούν regression σε διαφορετικές φάσεις του προγράμματος για τον υπολογισμό της ενέργειας και της ισχύος. Για αυτό ακολουθούν προσέγγιση δυναμικής ανάλυσης για να συλλέξουν χαρακτηριστικά που σχετίζονται με την εκτέλεση του προγράμματος, όπως cache misses, branch misses και τον αριθμό κύκλων. Οι προβλέψεις πάνω σε διαφορετικά σύνολα προγραμμάτων έχουν πολύ υψηλά ακρίβεια, που ξεπερνάει το 97%.

Ο Carlo Brandolese [11] έχει σχεδιάσει μία μεθοδολογία με την οποία εκτιμάται η ενεργειακή κατανάλωση και ο χρόνος εκτέλεσης σε ενσωματωμένο λογισμικό. Η μεθοδολογία χρησιμοποιεί στοιχεία δυναμικής και στατικής ανάλυσης και μπορεί να γενικευθεί για διαφορετικούς μεταγλωττιστές και αρχιτεκτονικές. Το πρώτο βήμα περιλαμβάνει την ανάλυση δέντρου και την αντιστοίχιση συνεισφορών κόστους (atoms) στους κόμβους. Κάθε atom μεταφράζεται σε kernel instructions, που αποτελούν εντολές assembly που δεν ανήκουν σε κάποια μικροεπεξεργαστή και χαρακτηρίζουν κάθε αρχιτεκτονική από άποψη κόστους. Το δεύτερο βήμα περιλαμβάνει την κατασκευή μαθηματικών μοντέλων που βασίζονται στις μετρήσεις για τον χρόνο και την ενέργεια κάθε kernel instruction στη συγκεκριμένη αρχιτεκτονική. Το τρίτο βήμα περιλαμβάνει την χρήση των μοντέλων για την κατασκευή probe συναρτήσεων για να εξασφαλίζεται η κάλυψη κάθε κόμβου. Με τους κόμβους πραγματοποιείται ενορχήστρωση του κώδικα και όταν εκτελείται καλείται το probe κάθε kernel instruction. Στο τελικό βήμα ένα εργαλείο πραγματοποιεί τις αριθμητικές πράξεις των εκτιμήσεων για να δώσει τελικό αποτέλεσμα.

Οι Meng Kewen κ.ά [12] έχουν χρησιμοποιήσει το εργαλείο Mira για την στατική ανάλυση της απόδοσης σε προγράμματα C και C++. Το εργαλείο είναι χτισμένο πάνω στον μεταγλωττιστή ROSE, του οποίου το περιβάλλον προσφέρει τη δυνατότητα για την παραγωγή αφηρημένο συντακτικού δέντρου από τον πηγαίο κώδικα. Το συντακτικό δέντρο προσφέρει πληροφορίες που χρησιμοποιούνται για την παραγωγή μετρικών που θα χρησιμοποιηθούν στο μοντέλο. Τα κομμάτια κώδικα που εξετάζονται είναι σώματα επαναλήψεων και χρησιμοποιούνται διάφορες μεθοδολογίες για τον υπολογισμό τους. Σε

κάποιες περιπτώσεις χρειάζεται εισαγωγή εισόδων από τον χρήστη για να οριστούν οι επαναλήψεις, τα μονοπάτια που ακολουθούνται και η αρχιτεκτονική που χρησιμοποιείται. Το εργαλείο δεν προβλέπει χρόνο ή ενέργεια , αλλά τις floating-point λειτουργίες σε κάθε block κώδικα.

Οι Callou Gustavo κ.ά [13] χρησιμοποιούν μεθόδους για την εκτίμηση του χρόνου εκτέλεσης και της ενεργειακής κατανάλωσης που βασίζονται σε coloured petri nets. Τα Petri Nets αποτελούν μαθηματικό εργαλείο που επιτρέπουν τη μοντελοποίηση σε παράλληλα, ασύγχρονα και μη ντετερμινιστικά συστήματα. Η μεθοδολογία που ακολουθείται περιλαμβάνει την ανάθεση τιμών πιθανότητας σε δομές επαναλήψεων και διακλαδώσεων μέσα από σημειώσεις στον κώδικα. Ο μεταγλωττισμένος κώδικας μεταφράζεται σε μοντέλο Coloured Petri Net και προσφέρει βάση για τη στοχαστική προσομοίωση του ενσωματωμένου λογισμικού. Σε κάθε τύπο block, όπως διακλαδώσεις, ορίζεται ένα τέτοιο μοντέλο. Επίσης χρησιμοποιείται ένα framework που αξιολογεί διαφορετικά σενάρια ροής ελέγχου με βάση τις πιθανότητες εισόδου. Το εργαλείο ALUPAS έχει αναπτυχθεί για να συνδυάζει τις λειτουργικότητες αυτού του framework ώστε να ανταπεξέλθει στις πολυπλοκότητες των εκτιμήσεων.

Οι Fernidan Christian κ.ά [14] έχουν χρησιμοποιήσει το εργαλείο aiT που προβλέπει τον χρόνο εκτέλεσης στη χειρότερη περίπτωση με στατική ανάλυση. Η λειτουργία του εργαλείου βασίζεται στη δημιουργία ενός γράφου ελέγχου ροής που προσδιορίζει τα διάφορα στοιχεία του προγράμματος και την ανάλυση πάνω σε αυτά. Η ανάλυση περιλαμβάνει την ανάθεση τιμών σε μεταβλητές μέσα σε κάποια όρια τιμών, την ανάθεση άνω ορίου στα σώματα επαναλήψεων και τον καθορισμό μονοπατιών στη χειρότερη περίπτωση.

Οι Xianfeng Li κ.ά [15] έχουν χρησιμοποιήσει ένα άλλο εργαλείο το Chronos που πραγματοποιεί παρόμοιες προβλέψεις για το χρόνο χρησιμοποιώντας στατική ανάλυση. Το εργαλείο είναι χτισμένο πάνω στην αρχιτεκτονική του προσομοιωτή SimpleScalar. Σε επίπεδο βασικών block πραγματοποιεί μία μοντελοποίηση για να εκτιμήσει το άνω όριο για το χρόνο εκτέλεσής του και με γραμμικό προγραμματισμό κάνει μία εκτίμηση για το συνολικό χρόνο συνδυάζοντας τον χρόνο των block.

Τα μειονεκτήματα των παραπάνω εργαλείων είναι ότι είναι πολύ αργά, η ακρίβεια τους δεν επαρκής και μπορούν να εφαρμοστούν μόνο σε συγκεκριμένα μοντέλα αρχιτεκτονικής.

Κεφάλαιο 4

Τεχνικό υπόβαθρο

1. Intel Architectural Code Analyzer

Για τους σκοπούς της διπλωματικής χρησιμοποιούμε το εργαλείο Architectural Code Analyzer [16] , που αποτελεί ένα open source εργαλείο από την Intel και έχει φτάσει στο τέλος της ζωής του. Ένα παρόμοιο εργαλείο το οποίο ακόμα δέχεται updates είναι το LLVM-MCA [17] , το οποίο ακολουθεί μία λογική που βασίζεται στο εργαλείο της Intel. Αυτό το δεύτερο εργαλείο το έχουμε χρησιμοποιήσει μόνο μερικές φορές για να συγκρίνουμε τα αποτελέσματά του με αυτά του πρώτου εργαλείου, το οποίο για συντομογραφία θα αποκαλείται IACA.

Το IACA δέχεται ως είσοδο ένα object αρχείο, το πηγαίο αρχείο του οποίου έχει μαρκαρισμένο συγκεκριμένο κομμάτι του κώδικα, και χρησιμοποιώντας στατική ανάλυση εκτιμά κάποιες μετρικές στο συγκεκριμένο κομμάτι του κώδικα. Το εργαλείο μπορεί να χρησιμοποιηθεί σε περιβάλλον Linux, που είναι και το περιβάλλον που παίρνουμε τις περισσότερες μετρήσεις μας.

1.1 Ανάλυση Throughput

Η ανάλυση γίνεται πάνω σε κομμάτια κώδικα, που αποτελούν σώματα επαναλήψεων. Το εργαλείο αντιμετωπίζει το σώμα σαν να εκτελείται άπειρες επαναλήψεις και λαμβάνει υπόψη τις εσωτερικές εξαρτήσεις που μπορεί να έχουν οι εντολές μεταξύ τους. Η ανάλυση δίνει ως έξοδο για μία επανάληψη κάποιες μετρικές για τα μεγέθη throughput και bottleneck στο σώμα του loop καθώς και μια πιο αναλυτική αναφορά για τις εντολές στο block.

Κατά την εκτέλεση του κώδικα assembly χρησιμοποιούνται διαφορετικές θύρες στις οποίες δεσμεύεται μία εντολή για κάποιο αριθμό κύκλων και αυτό εξαρτάται από τον τύπο της συγκεκριμένης εντολής. Για παράδειγμα, οι εντολές που γράφουν στην μνήμη δεσμεύονται σε διαφορετική θύρα από τις εντολές που κάνουν μία πρόσθεση. Το σύνολο των κύκλων για τους οποίους έχουν δεσμευθεί εντολές σε μία θύρα κατά την εκτέλεση του block αποτελεί και το throughput αυτής της θύρας. Η ανάλυση δίνει το throughput για κάθε θύρα επεξεργαστή και με βάση το σύνολό τους προκύπτει το throughput όλου του block. Το block throughput προκύπτει ως το μέγιστο μεταξύ των throughputs των επεξεργαστών, το μέγιστο front-end throughput και το throughput της μονάδας διαίρεσης που αποτελεί δική της θύρα. Επίσης δίνονται με τη μορφή της μετρικής bottleneck οι παράγοντες που μπορεί να περιορίσουν το throughput και οι οποίοι είναι το front-end , ο αριθμός θύρας , η μονάδα διαίρεσης ή μεγάλες αλυσίδες εξαρτήσεων.

Σε πιο αναλυτικό κομμάτι δίνονται γραμμές καθεμία από τις οποίες περιλαμβάνει μία εντολή από τον κώδικα assembly και πληροφορίες που σχετίζονται με αυτήν την εντολή. Ο αριθμός των εντολών καθώς και οι ίδιες οι εντολές εξαρτώνται από την μεταγλώττιση που έγινε για να προκύψει το object file. Ανάλογα με τις σημαίες βελτιστοποίησης που χρησιμοποιήθηκαν ο αριθμός των εντολών μπορεί να μειωθεί. Σε αυτή τη διπλωματική για λόγους απλότητας τα προγράμματα θα μεταγλωττίζονται χωρίς κάποια σημαία βελτιστοποίησης.

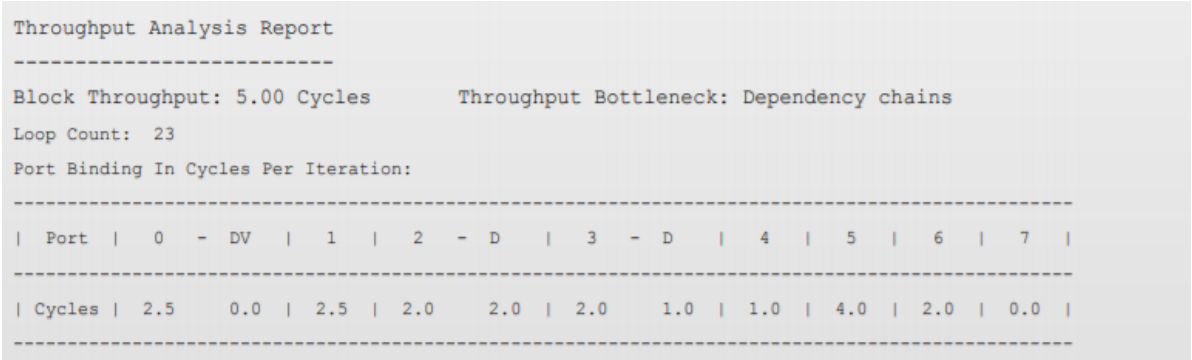
Μία από τις πληροφορίες που αφορούν τις εντολές είναι ο αριθμός των micro-operations που αντιστοιχούν σε κάθε εντολή. Τα micro-operations αποτελούν χαμηλού επιπέδου εντολές που περιγράφουν τις πιο πολύπλοκες εντολές. Για παράδειγμα, μία εντολή που γράφει στη μνήμη , χρειάζεται 2 micro-operations, μία για να αποθηκεύσει τα δεδομένα και μία για να αποθηκεύσει τη διεύθυνση. Ο μέγιστος αριθμός micro operations για το front-end είναι 4 ανά κύκλο.

Επίσης ως πληροφορία δίνεται ο μέσος αριθμός κύκλων που η εντολή δεσμεύεται σε κάθε θύρα. Κάθε γραμμή είναι χωρισμένη σε 8 στήλες, καθεμία από τις οποίες αντιπροσωπεύει μία διαφορετική θύρα. Κάποιες από αυτές τις θύρες έχουν πέρα από το βασικό και ένα δευτερεύον pipe. Η θύρα 0 έχει το pipe διαίρεσης χωρισμένο από αυτή και μπορεί να εκτελέσει κάποιο άλλο micro-operation , όσο το pipe είναι απασχολημένο με κάποια διαίρεση. Οι θύρες 2 και 3 αποτελούν θύρες για διάβασμα δεδομένων από τη μνήμη και η καθεμία έχει μία μονάδα παραγωγής διευθύνσεων χωρισμένη από αυτή. Στις περισσότερες περιπτώσεις η πληροφορία στη γραμμή είναι ο αριθμός των κύκλων που η εντολή δεσμεύτηκε σε κάποια θύρα. Σε κάποιες όμως περιπτώσεις μπορεί ένα micro-operation να εκτελεστεί σε κάποια διαφορετική θύρα ανάλογα με την επανάληψη και έτσι θα υπάρχει μερικός κύκλος στις συγκεκριμένες θύρες.

Σε μερικές περιπτώσεις τα micro-operations δεν δεσμεύουν κάποια θύρα και συμβολίζονται με ‘*’ , ενώ μπορεί να υπάρξουν συγχωνεύσεις operations. Επίσης, κάποιες από τις εντολές δεν υποστηρίζονται από το IACA και για αυτό συμβολίζονται με X. Από όλες τις μετρήσεις που πραγματοποιήθηκαν η μόνη εντολή που φάνηκε να μην αναγνωρίζεται είναι η εντολή return. Αυτό δεν αποτελεί κάποιο πρόβλημα καθώς οι μετρήσεις έγιναν πάνω σε κομμάτια κώδικα που αποτελούν σώματα επαναλήψεων και δεν περιέχουν εντολές που διακόπτουν την επανάληψη όπως break ή return. Άλλη περίπτωση αποτελεί η χρήση εντολών από διαφορετικά μοντέλα Intel , κάτι που μπορεί να προκαλέσει μεγάλα σφάλματα από άποψη κύκλων, και το εργαλείο τα αναγνωρίζει χρησιμοποιώντας το σύμβολο ‘@’ . Στις μετρήσεις που έγιναν δεν παρατηρήθηκαν τέτοιες περιπτώσεις.

Ως τελική πληροφορία δίνεται ο συνολικός αριθμός των micro-operations στο block κώδικα , καθώς και κάποιες σχετικές σημειώσεις για την ανάλυση. Σημαντικό είναι να τονιστεί ότι αν και το εργαλείο αναγνωρίζει την πιθανότητα εξαρτήσεων αντιμετωπίζει το πρόγραμμα σε ιδανικές συνθήκες· αυτό σημαίνει ότι το εκτελεί αγνοώντας εξαρτήσεις του τύπου out-of-order και συνθήκες ιεραρχίας μνήμης. Στις μετρήσεις που έχουν γίνει αυτό δεν αποτελεί κάποιο πρόβλημα για τις εντολές που εκτελούνται σε ένα block κώδικα, ωστόσο σημαίνει ότι δεν πραγματοποιούνται οι επαναλήψεις με την ίδια ταχύτητα.

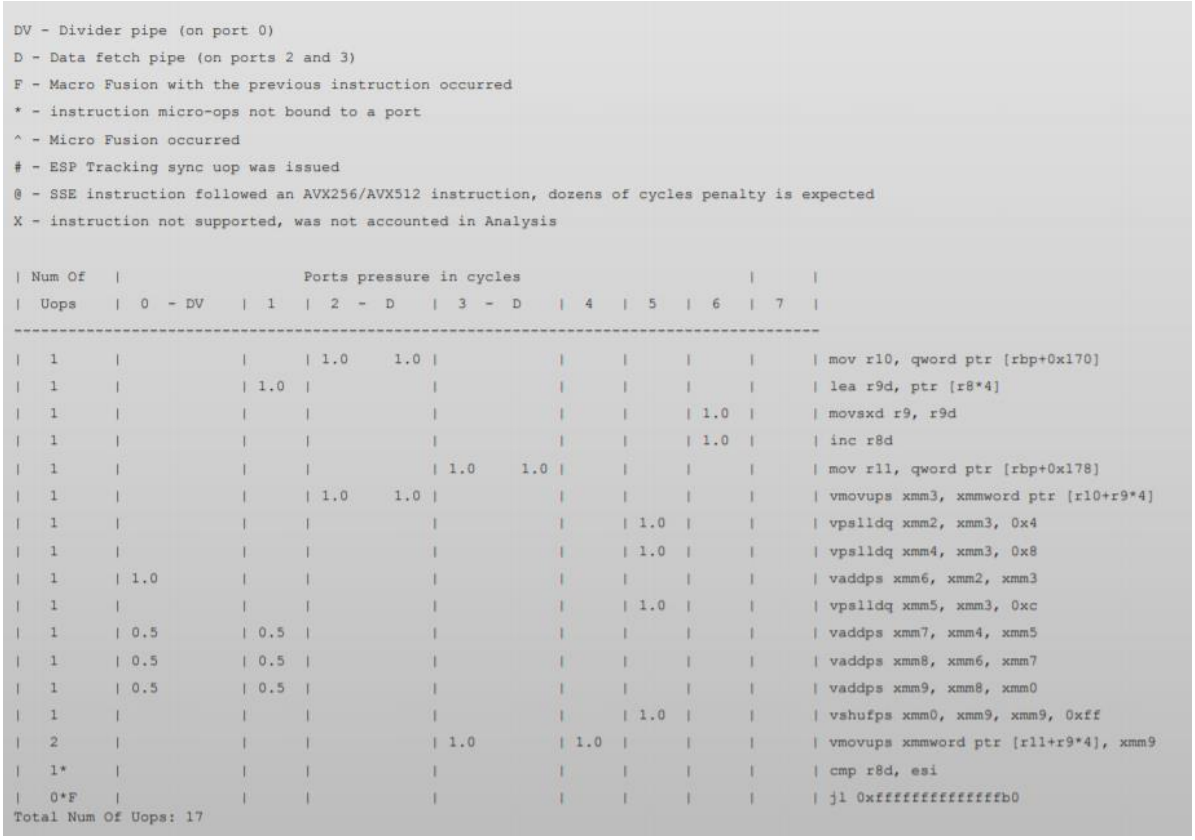
Για την καλύτερη κατανόηση των παραπάνω λειτουργιών θα χρησιμοποιηθούν τα δύο παρακάτω σχήματα που αποτελούν τα αποτελέσματα της ανάλυσης με το IACA πάνω σε ένα τυχαίο πρόγραμμα. Η πρώτη εικόνα δείχνει τα μεγέθη για το throughput του block και των θυρών , ενώ η δεύτερη την ανάλυση σε assembly.



```
Throughput Analysis Report
--------------------------
Block Throughput: 5.00 Cycles       Throughput Bottleneck: Dependency chains
Loop Count:  23
Port Binding In Cycles Per Iteration:
---------------------------------------------------------------------------------------------
|  Port  |   0   -  DV   |   1   |   2   -  D    |   3   -  D    |   4   |   5   |   6   |   7   |
---------------------------------------------------------------------------------------------
| Cycles |  2.5     0.0  |  2.5  |  2.0     2.0  |  2.0     1.0  |  1.0  |  4.0  |  2.0  |  0.0  |
---------------------------------------------------------------------------------------------
```

Σχήμα 4.1 : Μεγέθη throughput, bottleneck και port binding

Όπως φαίνεται το throughout του block είναι 5 κύκλοι και ο παράγοντας που το περιόρισε είναι πολλές συνεχόμενες εξαρτήσεις μέσα στο block. Για να υπολογίσει αυτά τα αποτελέσματα , το εργαλείο χρειάστηκε να εκτελέσει αυτό το κομμάτι κώδικα 23 φορές. Από τη δέσμευση θυρών παρατηρείται ότι η θύρα για την διαίρεση δεν έχει δεσμευθεί , άρα δεν υπάρχουν διαιρέσεις σε αυτό το block, ενώ οι θύρες για το διάβασμα από τη μνήμη έχουν δεσμευθεί για 2 κύκλους η καθεμία.



```
DV - Divider pipe (on port 0)
D - Data fetch pipe (on ports 2 and 3)
F - Macro Fusion with the previous instruction occurred
* - instruction micro-ops not bound to a port
^ - Micro Fusion occurred
# - ESP Tracking sync uop was issued
@ - SSE instruction followed an AVX256/AVX512 instruction, dozens of cycles penalty is expected
X - instruction not supported, was not accounted in Analysis

| Num Of   |                   Ports pressure in cycles                       |      |
|  Uops    |  0  - DV   |  1   |  2  -  D    |  3  -  D    |  4   |  5   |  6   |  7   |
-----------------------------------------------------------------------------------------
|   1      |            |      | 1.0    1.0  |             |      |      |      |      | mov r10, qword ptr [rbp+0x170]
|   1      |            | 1.0  |             |             |      |      |      |      | lea r9d, ptr [r8*4]
|   1      |            |      |             |             |      |      | 1.0  |      | movsxd r9, r9d
|   1      |            |      |             |             |      |      | 1.0  |      | inc r8d
|   1      |            |      |             | 1.0    1.0  |      |      |      |      | mov r11, qword ptr [rbp+0x178]
|   1      |            |      | 1.0    1.0  |             |      |      |      |      | vmovups xmm3, xmmword ptr [r10+r9*4]
|   1      |            |      |             |             |      | 1.0  |      |      | vpslldq xmm2, xmm3, 0x4
|   1      |            |      |             |             |      | 1.0  |      |      | vpslldq xmm4, xmm3, 0x8
|   1      | 1.0        |      |             |             |      |      |      |      | vaddps xmm6, xmm2, xmm3
|   1      |            |      |             |             |      | 1.0  |      |      | vpslldq xmm5, xmm3, 0xc
|   1      | 0.5        | 0.5  |             |             |      |      |      |      | vaddps xmm7, xmm4, xmm5
|   1      | 0.5        | 0.5  |             |             |      |      |      |      | vaddps xmm8, xmm6, xmm7
|   1      | 0.5        | 0.5  |             |             |      |      |      |      | vaddps xmm9, xmm8, xmm0
|   1      |            |      |             |             |      | 1.0  |      |      | vshufps xmm0, xmm9, xmm9, 0xff
|   2      |            |      |             | 1.0         | 1.0  |      |      |      | vmovups xmmword ptr [r11+r9*4], xmm9
|   1*     |            |      |             |             |      |      |      |      | cmp r8d, esi
|   0*F    |            |      |             |             |      |      |      |      | jl 0xfffffffffffffb0
Total Num Of Uops: 17
```

Σχήμα 4.2 : Εκτενής ανάλυση throughput

Στην ανάλυση σε assembly βλέπουμε πως οι εντολές δεσμεύονται στις θύρες. Από αυτήν την εικόνα μπορούν να προκύψουν κάποια ενδιαφέροντα συμπεράσματα. Υπάρχουν 3 εντολές οι οποίες αποτελούνται από 1 micro-operation , αλλά έχουν δεσμεύσει μισό κύκλο από τις θύρες 0 και 1. Αυτό σημαίνει ότι σε διαφορετικές επαναλήψεις η εντολή δε θα δεσμευόταν πάντα στην ίδια θύρα , αλλά θα εναλλασσόταν μεταξύ αυτών των δύο. Μέσα σε

όλες τις εντολές υπάρχει μία η οποία είναι πιο περίπλοκη και αποτελείται από 2 micro-operations και για αυτό δεσμεύεται σε δύο θύρες. Η προτελευταία εντολή , που είναι εντολή σύγκρισης, δεν έχει δεσμευθεί σε κάποια θύρα , και έχει συγχωνευτεί με την τελευταία , που είναι εντολή άλματος. Συνολικά σε αυτό το block κώδικα υπάρχουν 17 micro-operations, που είναι όσα και ο αριθμός των εντολών.

1.2 Ανάλυση Trace

Το εργαλείο διαθέτει τη σημαία -trace 'path', η οποία δημιουργεί ένα trace αρχείο στην τοποθεσία 'path' . Το trace αρχείο περιλαμβάνει μια πιο εκτενή ανάλυση του κώδικα assembly και δίνει πληροφορία για τα στάδια λειτουργίας κάθε εντολής μέσα στον επεξεργαστή. Αυτό είναι σημαντικό γιατί έτσι ο προγραμματιστής μπορεί να εξετάσει τις εξαρτήσεις μεταξύ των εντολών και να προσδιορίσει σημεία bottleneck.

Το παρακάτω παράδειγμα αποτελεί τμήμα του αρχείου trace για τον κώδικα της προηγούμενης ενότητας.

```
it|in|Dissasembly                                :012345678901234567890123456789012345678
 0| 0|mov r10, qword ptr [rbp+0x170]             :          |         |         |
 0| 0|    TYPE_LOAD (1 uops)                     :s---deeeew----R-------p       |
 0| 1|lea r9d, ptr [r8*4]                        :          |         |         |
 0| 1|    TYPE_OP (1 uops)                       :sdw----------R-------p       |
 0| 2|movsxd r9, r9d                             :          |         |         |
 0| 2|    TYPE_OP (1 uops)                       :A-dw----------R-------p       |
 0| 3|inc r8d                                    :          |         |         |
 0| 3|    TYPE_OP (1 uops)                       :sdw----------R-------p       |
 0| 4|mov r11, qword ptr [rbp+0x178]             :          |         |         |
 0| 4|    TYPE_LOAD (1 uops)                     : s---deeeew----R-------p      |
 0| 5|vmovups xmm3, xmmword ptr [r10+r9*4]       :          |         |         |
 0| 5|    TYPE_LOAD (1 uops)                     : A-------deeeeew----R-------p |
 0| 6|vpslldq xmm2, xmm3, 0x4                    :          |         |         |
 0| 6|    TYPE_OP (1 uops)                       : A-------------dw----R-------p|
 0| 7|vpslldq xmm4, xmm3, 0x8                    :          |         |         |
 0| 7|    TYPE_OP (1 uops)                       : A-------------cdw----R-------p
 0| 8|vaddps xmm6, xmm2, xmm3                    :          |         |         |
 0| 8|    TYPE_OP (1 uops)                       :  A-------------deeew----R-------p
 0| 9|vpslldq xmm5, xmm3, 0xc                    :          |         |         |
 0| 9|    TYPE_OP (1 uops)                       :  A------------ccdw------R-------p
```

Σχήμα 4.3 : Trace αρχείο

Οι εντολές παρουσιάζονται σε μορφή από πάνω προς τα κάτω , ενώ οι κύκλοι του επεξεργαστή τρέχουν από αριστερά προς τα δεξιά. Η πρώτη στήλη 'it' δείχνει σε ποια επανάληψη βρίσκεται το loop , η δεύτερη στήλη 'in' περιέχει τον αριθμό που αντιστοιχεί στην σειρά της εντολής μέσα στο block , η τρίτη στήλη δείχνει της εντολές σε μορφή disassembly , ενώ στα δεξιά της περιλαμβάνει πληροφορίες για την micro-αρχιτεκτονική κατάτμηση των εντολών. Το IACA ως default λειτουργία δείχνει στο αρχείο τους 150 πρώτους κύκλους της εκτέλεσης.

Για τις εντολές δίνεται ως πληροφορία η κατάτμησή τους και πόσα micro-operations αντιστοιχούν σε κάθε τμήμα. Κάθε εντολή αντιπροσωπεύεται από 4 το πολύ τμήματα, το OP, το LOAD, το STORE_DATA, και το STORE_ADDRESS. Στην κατηγορία OP πηγαίνουν όσα micro-operations δεν έχουν σχέση με διάβασμα ή γράψιμο στη μνήμη και αυτό περιλαμβάνει αριθμητικές και λογικές εντολές , καθώς και εντολές άλματος. Άμα το micro-operation ανήκει σε κάποια από τις άλλες κατηγορίες, αυτό δίνει μόνο την πιθανότητα αν θα εκτελεστεί η ενέργεια , καθώς το εργαλείο δεν προβλέπει ειδικές περιπτώσεις , όπως τη περίπτωση cache miss. Ωστόσο , στις μετρήσεις που έγιναν δε φαίνεται αυτός ο παράγοντας να επηρέασε πολύ τα τελικά αποτελέσματα. Αν και δε φαίνεται στην εικόνα , μπορεί κάποια

εντολή να αντιπροσωπεύεται από ένα τμήμα που του αντιστοιχούν περισσότερα από ένα micro-operation. Το εργαλείο αντιστοιχεί την εντολή σε κάποιο τμήμα, ακόμα και αν δεν περιέχει κάποιο micro-operation.

Το trace παρουσιάζει τα στάδια από τα οποία περνάει το τμήμα της εντολής για κάθε κύκλο από το στάδιο που γίνεται διαθέσιμο (Allocate) μέχρι το στάδιο παραίτησης (retire-post retire). Αν δύο στάδια πραγματοποιηθούν στον ίδιο κύκλο , τότε εμφανίζεται το πιο σημαντικό από τα 2.

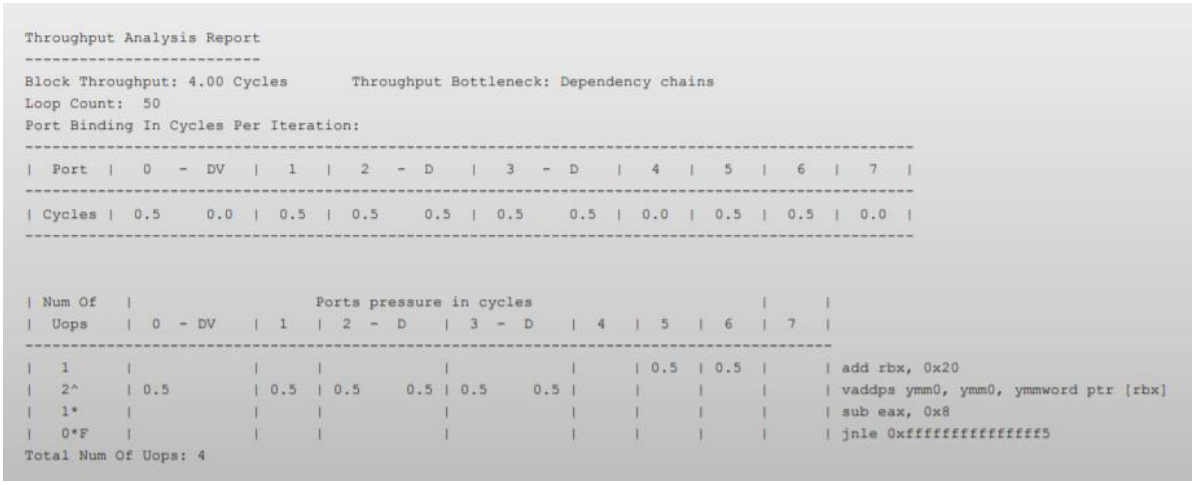
Τα στάδια και πιθανές καταστάσεις είναι τα ακόλουθα:

[A]-Allocated : Σε αυτό το στάδιο η εντολή μπαίνει στην ουρά εκτέλεσης
[s]-Sources ready : Αυτό το στάδιο επικαλύπτει ως πιο σημαντικό το A αν και τα δύο πραγματοποιούνται στον ίδιο κύκλο.
[c]-Port conflict: Σύγκρουση θυρών
[d]-Dispatch for execution: Όταν η εντολή φτάσει σε αυτό το στάδιο, στο επόμενο στάδιο θα ξεκινήσει η εκτέλεση της εντολής
[e]-Execute: Στάδιο εκτέλεσης. Μπορεί να υπάρχουν πολλά συνεχόμενα στάδια εκτέλεσης
[w]-Writeback: Γράψιμο στη μνήμη.
[R]-Retired: Στάδιο παραίτησης.
[p]-Post-retire: Στάδιο μετα-παραίτησης. Όταν αυτό το στάδιο ολοκληρωθεί η εντολή εγκαταλείπει την ουρά εκτέλεσης
[-]-pending. Αυτό το στάδιο χρησιμοποιείται για να δείξει τους κύκλους που περνάνε μεταξύ των υπόλοιπων σταδίων.
[_]-Η καθυστέρηση που μπαίνει όταν δεν υπάρχουν διαθέσιμοι πόροι.

Από την εικόνα προκύπτουν κάποια ενδιαφέροντα συμπεράσματα. Η πρώτη εντολή, η εντολή 0, ανήκει στην κατηγορία LOAD και όπως φαίνεται στο trace περνά από αρκετά στάδια εκτέλεσης. Γενικά, οι εντολές που κάνουν πρόσβαση στη μνήμη χρησιμοποιούν περισσότερο χρόνο σε σχέση με άλλου τύπου εντολές, όπως οι αριθμητικές. Η εντολή 1 είναι τύπου OP και μεταφέρει μια τιμή στον register r9d. Τα στάδια που περνάει είναι συνεχόμενα τα s, d και w και το στάδιο e προφανώς πραγματοποιήθηκε μαζί με το στάδιο w στον ίδιο κύκλο για αυτό δεν εμφανίζεται. Η εντολή 2 είναι και αυτή τύπου OP και χρησιμοποιεί τον register r9d. Για αυτόν τον λόγο δεν μπορεί να ξεκινήσει την εκτέλεση μέχρι να γίνει το στάδιο writeback στην εντολή 1. Άλλες τέτοιες εξαρτήσεις παρατηρούνται μέσα στην εικόνα. Η εντολή 5 κάνει load στην μεταβλητή xmm3 και η εντολή 6 χρησιμοποιεί την ίδια μεταβλητή οπότε χρειάζεται να περιμένει μέχρι το στάδιο writeback της εντολής 5. Μία ακόμα παρατήρηση σε αυτήν την εικόνα είναι ότι μπορεί μέχρι 4 εντολές να μπουν στην ουρά εκτέλεσης στον ίδιο κύκλο.

Οι παρατηρήσεις του trace κάθε εντολής βοηθάει στην κατανόηση των εξαρτήσεων μεταξύ των εντολών και εκτός από την αποφυγή των διαφόρων κινδύνων μπορεί να χρησιμοποιηθεί για τη βελτιστοποίηση της απόδοσης του κώδικα.

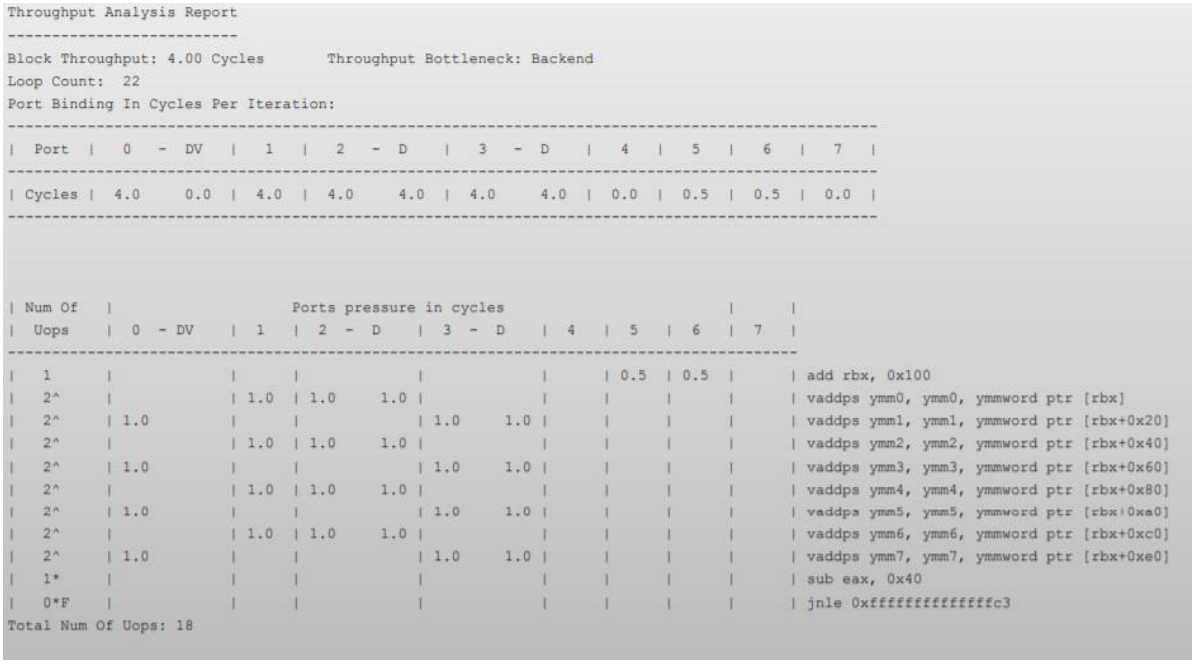
Ένα σχετικό παράδειγμα παρουσιάζεται παρακάτω:



Σχήμα 4.4 : Ανεπαρκής αξιοποίηση των θυρών

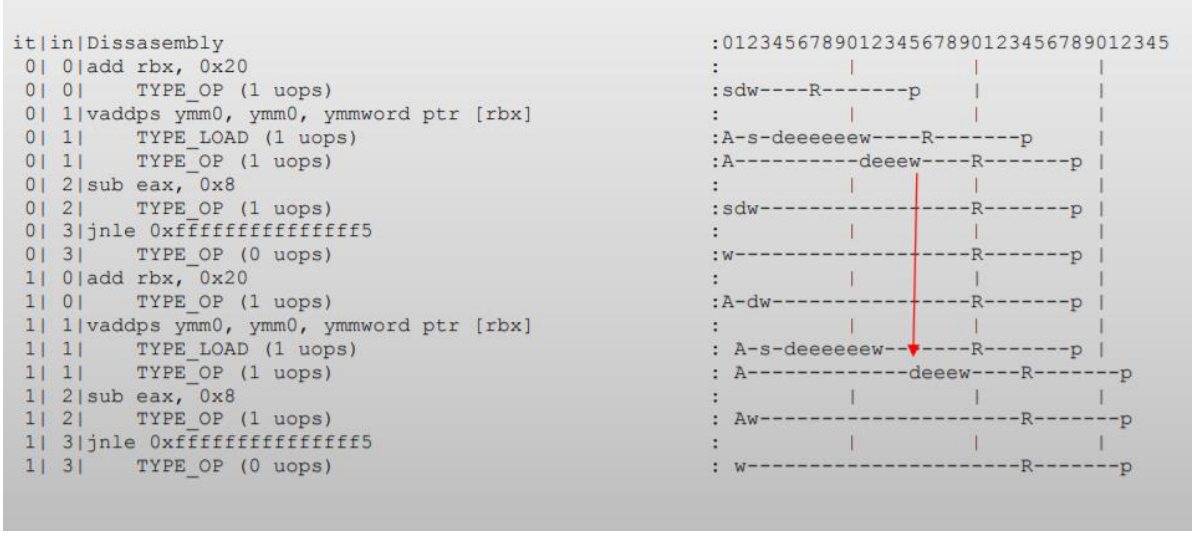
Το παραπάνω κομμάτι κώδικα αποτελεί κομμάτι ενός σώματος επαναλήψεων που χρησιμοποιείται για να υπολογίζει το άθροισμα των στοιχείων ενός πίνακα. Το throughput του block είναι 4 κύκλοι και παρατηρούμε ότι το throughput στις θύρες είναι το πολύ 0.5. Αυτό γίνεται επειδή πραγματοποιείται μόνο μία εντολή vaddps ανά επανάληψη και η εντολή δεσμεύεται σε διαφορετική θύρα ανά επανάληψη.

Μία μέθοδος που μπορεί να εφαρμοστεί για να αποφευχθεί αυτό το πρόβλημα είναι να γίνει loop unrolling και να εκτελούνται 8 τέτοιες εντολές ανά επανάληψη. Με αυτή τη μέθοδο θα αποφευχθούν οι εξαρτήσεις μεταξύ των εντολών vaddps και όπως φαίνεται από την παρακάτω εικόνα θα αυξηθεί το throughput των θυρών , ενώ το throughput του block θα παραμείνει το ίδιο.



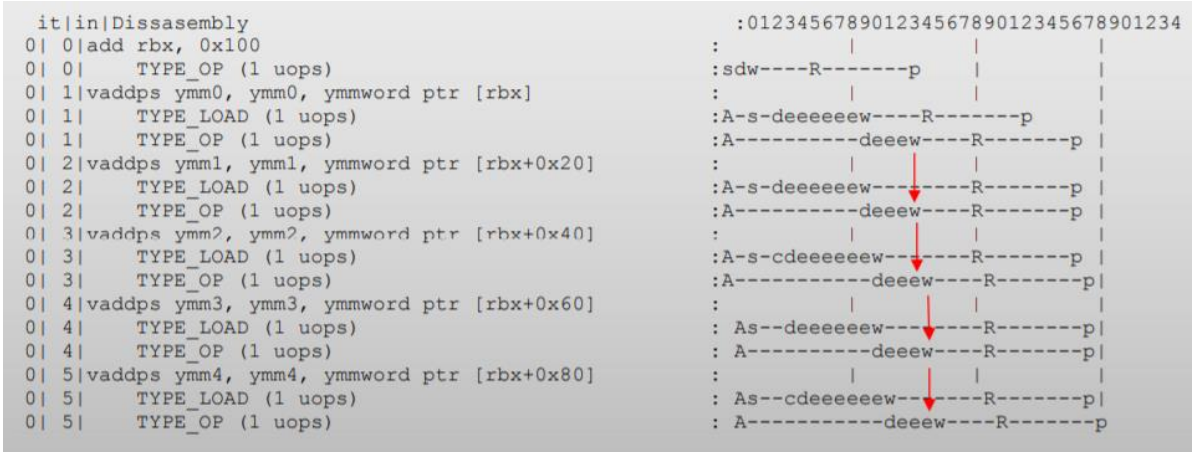
Σχήμα 4.5 : Καλύτερη αξιοποίηση των θυρών

Όπως φαίνεται από την εικόνα οι θύρες χρησιμοποιούνται για ολόκληρους κύκλους αντί για μισούς και έτσι για τους ίδιους κύκλους γίνονται πολύ περισσότερες εργασίες. Το trace αρχείο κάνει πιο κατανοητή τη διαφορά μεταξύ των δύο υλοποιήσεων.



Σχήμα 4.6 : Trace αρχείο πριν τη βελτίωση

Στην πρώτη εικόνα παρατηρείται ότι οι εντολές διατηρούν εξαρτήσεις μεταξύ τους και σε διαφορετικές επαναλήψεις που σημαίνει ότι πριν ξεκινήσει η εκτέλεση της μίας εντολής πρέπει να έχει πραγματοποιηθεί το στάδιο writeback στην προηγούμενη.



Σχήμα 4.7 : Trace αρχείο μετά την αναδιοργάνωση των εντολών

Στην δεύτερη εικόνα παρατηρείται ότι οι εντολές δεν έχουν τέτοιες εξαρτήσεις μεταξύ τους και η μόνη καθυστέρηση προκύπτει κάθε 2 εντολές που παρατηρείται port conflict. Επομένως στους ίδιους κύκλους γίνονται περισσότερες λειτουργίες και υπάρχει σημαντική βελτίωση της απόδοσης.

Στις μετρήσεις που έχουν γίνει για τη διπλωματική δεν χρησιμοποιείται αυτή η διαδικασία βελτιστοποίησης καθώς οι μετρικές αφορούν την απόδοσή προγραμμάτων σε πιο γενικό πλαίσιο.

1.3 Χρήση του IACA

Το εργαλείο χρησιμοποιείται σε αρχεία γραμμένα σε γλώσσα C ή C++ και η μορφή του κώδικα εξαρτάται από τις σημαίες βελτιστοποίησης που χρησιμοποιούνται. Τα σώματα επαναλήψεων που μπορεί να αναλύσει είναι τύπου for και while.

Για να αντιληφθεί το IACA το κομμάτι κώδικα που θα αναλυθεί χρησιμοποιούνται δύο macros , το IACA_START και IACA_END. Αυτά τα 2 macros πρέπει να είναι τοποθετημένα μέσα σε ένα σώμα επαναλήψεων με συγκεκριμένο τρόπο , διαφορετικά το εργαλείο δεν τα αναγνωρίζει ως σημαίες που ορίζουν το κομμάτι για ανάλυση. Τα macros περιέχονται μέσα στο αρχείο iacaMarks.h , το οποίο πρέπει να συμπεριληφθεί και να μεταγλωττιστεί μέσα στο αρχείο που θα γίνει η ανάλυση. Ο τρόπος που πρέπει να τοποθετηθούν τα macros μέσα στο loop είναι:

```
while ( condition )
{
    IACA_START
    <loop body>
}
IACA_END
```

Για την εκτέλεση του IACA πρέπει να γραφτεί και να εκτελεστεί στο command line η παρακάτω εντολή:

```
iaca <options> <input file name>
```

Το <input file name> αντιστοιχεί στο object αρχείο που έχει μεταγλωττιστεί από το αρχικό αρχείο

Από το <options> οι σημαίες που χρησιμοποιούνται για τη διπλωματική είναι:

-trace <file> : Αυτή η σημαία δημιουργεί ένα trace αρχείο με όνομα <file>
-trace-cycle-count : Ο default αριθμός κύκλων που παρουσιάζονται στο trace αρχείο είναι 150. Σε αρκετά αρχεία που έχουν γίνει μετρήσεις ο κώδικας disassembly είναι αρκετά μεγάλος , οπότε δεν καλύπτει ολόκληρη την επανάληψη . Με τη σημαία αυτή ρυθμίζεται πόσοι κύκλοι θα εμφανιστούν στο trace αρχείο , οπότε για μεγάλα τμήματα κώδικα θα τίθεται σε μεγαλύτερη τιμή.

Ο χρόνος εκτέλεσης του εργαλείου εξαρτάται από το μέγεθος του κώδικα που θα γίνει ανάλυση. Ακόμα και στις περιπτώσεις μεγάλου κώδικα το εργαλείο θα τον εκτελέσει μόνο μερικές φορές, οπότε ο χρόνος εκτέλεσης είναι πολύ μικρότερος από την περίπτωση που εκτελείται το ίδιο το πρόγραμμα. Αυτό σημαίνει ότι μπορεί να πάρει χιλιάδες μετρήσεις και να βγάλει αποτελέσματα μέσα σε λίγα λεπτά. Από το εργαλείο θα χρησιμοποιηθούν και το αρχικό και το trace αρχείο για να προκύψουν κάποιες σημαντικές μετρικές.

2. Clang parser

2.1 Εισαγωγή στη Clang

Η Clang [18] είναι ένας μεταγλωττιστής για τις γλώσσες που ανήκουν στην οικογένεια της C (όπως η C, C++, Objective-C) και έχει ως βάση την υποδομή του μεταγλωττιστή LLVM. Η αποδοχή της στον κόσμο του προγραμματισμού βασίζεται στο ότι κάνει πολύ γρήγορες μεταγλωττίσεις, καταναλώνοντας λιγότερη μνήμη από άλλους μεταγλωττιστές όπως ο GCC και στο ότι προσφέρει ένα περιβάλλον για τη δημιουργία εργαλείων. Οι χρήσεις και οι λειτουργίες της Clang είναι πολλές και η μελέτη όλων είναι κάτι που δε θα καλύψει η συγκεκριμένη διπλωματική, οπότε για περισσότερη μελέτη τους ο όποιος ενδιαφερόμενος πρέπει να συμβουλευτεί τη σχετική βιβλιογραφία. Προς το παρόν θα γίνει μια περιεκτική αναφορά για τις ιδιότητες της clang ως μεταγλωττιστής και ως βιβλιοθήκη.

2.2 Η Clang ως μεταγλωττιστής

Ο μεταγλωττιστής Clang υποστηρίζει τη μεταγλώττιση των προγραμματιστικών γλωσσών που ανήκουν στην οικογένεια της C καθώς και ένα σύνολο επεκτάσεων τους. Αυτές οι επεκτάσεις παρέχονται όντας συμβατές με το GCC και άλλους μεταγλωττιστές και για να βελτιώσουν την λειτουργικότητα μέσα από κάποια συγκεκριμένα χαρακτηριστικά. Αυτά τα χαρακτηριστικά είναι σχεδιασμένα ώστε να είναι συμβατά με το GCC και έτσι να είναι δυνατή η μετάβαση από GCC σε Clang. Εκτός αυτού, η Clang παρέχει και κάποια ακόμα χαρακτηριστικά τα οποία εξαρτώνται από την αρχιτεκτονική CPU και το λειτουργικό σύστημα στα οποία μεταγλωττίζεται το πρόγραμμα. Η τελευταία έκδοση της Clang υποστηρίζει τα ακόλουθα σε κάθε γλώσσα:

Γλώσσα C:
Η υποστήριξη για τη standard C περιλαμβάνει όλα τα στοιχεία εκτός από τα C99 floating-point pragmas

Επεκτάσεις που δεν έχουν πραγματοποιηθεί για τον GCC:
Δεν υποστηρίζονται οι αριθμοί τύπου δεκαδικού floating point και fixed point
Δεν υποστηρίζονται οι nested συναρτήσεις
Δεν υποστηρίζεται η αρχικοποίηση των μελών ευέλικτων πινάκων

Γλώσσα C++:
Υποστηρίζονται όλα τα χαρακτηριστικά για τη standard C++98 εκτός από τα εξαγόμενα templates

Υποστήριξη για τις διαφορετικές αρχιτεκτονικές και οι περιορισμοί τους:

x86:
Η υποστήριξη για x86 θεωρείται σταθερή σε περιβάλλοντα macOS, Linux, FreeBSD και Dragonfly BSD. Για στόχο x86 η clang διαθέτει την επιλογή -m16 στο command line που χρησιμοποιεί για τον κώδικα έξοδο 16-bit κάτι που είναι χρήσιμο για CPU που τρέχει σε περιβάλλον 16-bit.

ARM:
Η υποστήριξη για ARM, ειδικότερα για ARM6 και ARM7, θεωρείται σταθερή σε περιβάλλον iOS καθώς έχει δοκιμαστεί σε μεγάλες βάσεις κώδικα σε C, C++ , Objective-C

και Objective-C++. Η Clang υποστηρίζει πλήρως μόνο μερικές αρχιτεκτονικές ARM, δηλαδή για παράδειγμα δεν υποστηρίζει πλήρως την ARM5.
PowerPC:
Η υποστήριξη για PowerPC θεωρείται σταθερή σε περιβάλλοντα Linux και FreeBSD και έχει δοκιμαστεί επιτυχώς σε μεγάλες βάσεις κώδικα σε C και C++, αν και στερείται κάποιων χαρακτηριστικών.

Στις υπόλοιπες πλατφόρμες η Clang προσφέρει μερική υποστήριξη.

2.2.1 Βελτιστοποίηση

Στα περιβάλλοντα που υποστηρίζει επαρκώς , η Clang κάνοντας μεταγλώττιση και link με ειδικές πληροφορίες επιτρέπει να γίνονται βελτιστοποιήσεις στον κώδικα. Μία μέθοδος βελτιστοποίησης αποτελεί η βελτιστοποίηση βασιζόμενη σε προφίλ (profile) [19].

Οι πληροφορίες προφίλ επιτρέπουν καλύτερη βελτιστοποίηση, καθώς γνωρίζοντας τη συχνότητα των διακλαδώσεων ή συγκεκριμένων συναρτήσεων επιτρέπει στον μεταγλωττιστή να παίρνει καλύτερες αποφάσεις Η Clang προσφέρει βελτιστοποίηση βασιζόμενη σε προφίλ χρησιμοποιώντας δύο τεχνικές profiling, το profiling δείγματος που παράγει προφίλ με πολύ μικρό overhead του χρόνου εκτέλεσης ή μία ενορχήστρωση του κώδικα που συλλέγει πιο αναλυτικές πληροφορίες προφίλ.

Profilers δείγματος

Τα προφίλ δείγματος δεν παρέχονται από τη Clang , αλλά από κάποιο εξωτερικό εργαλείο που κάνει το profiling και το προφίλ που παράγεται πρέπει να μετατραπεί ώστε να διαβάζεται από το LLVM. Αρχικό βήμα είναι να χτιστεί ένα εκτελέσιμο αρχείο χρησιμοποιώντας τη σημαία -gline-table-only ή -g όπως φαίνεται παρακάτω:

```
$ clang++ -O2 -gline-tables-only code.cc -o code
```

Μετά το εκτελέσιμο τρέχει σε κάποιο profiler , όπως το εργαλείο perf. Τα δεδομένα του προφίλ που μαζεύονται πρέπει να μετατραπούν σε μορφή κατανοητή από το LLVM. Αυτό υποστηρίζεται από τον μετατροπέα create_llvm_prof.

```
$ create_llvm_prof --binary=./code --out=code.prof
```

Αυτή η εντολή διαβάζει τα δεδομένα που συγκεντρώθηκαν από το εργαλείο και το αρχείο ./code και μεταβιβάζει τα δεδομένα στο code.prof.
Χρησιμοποιώντας το προφίλ που συλλέχθηκε ο προγραμματιστής ξαναχτίζει τον κώδικα και το παραγόμενο εκτελέσιμο είναι πιο γρήγορο από το προηγούμενο. Η εντολή στο command line είναι:

```
$ clang++ -O2 -gline-tables-only -fprofile-sample-use=code.prof code.cc -o code
```

Profiling μέσω ενορχήστρωσης

Το profiling ενορχήστρωσης απαιτεί την ενορχήστρωση του κώδικα και ενώ προκαλεί κάποιο overhead του χρόνου εκτέλεσης , προσφέρει πιο ακριβή αποτελέσματα από την προηγούμενη μέθοδο.
Για την ενορχήστρωση του κώδικα γίνεται η μεταγλώττιση και η σύνδεση χρησιμοποιώντας τη σημαία -fprofile-instr-generate.

Όταν το αρχείο εκτελεστεί με την οποιαδήποτε είσοδο, τα δεδομένα προφίλ θα γραφτούν στο αρχείο default.profraw σε φάκελο εκτέλεσης . Ο χρήστης μπορεί να ορίσει το αρχείο αποθήκευσης βάζοντας το όνομα του αρχείου στην μεταβλητή περιβάλλοντος LLVM_PROFILE_FILE.
Για καλύτερο αποτέλεσμα ο χρήστης μπορεί να συνδυάσει προφίλ από πολλαπλές εκτελέσεις και μετά πρέπει να μετατρέψει το προφίλ σε είσοδο που περιμένει η Clang. Αυτό γίνεται χρησιμοποιώντας την εντολή merge του εργαλείου llvm-profdata.

```
$ clang++ -O2 -fprofile-instr-use=code.profdata code.cc -o code
```

Μετά από αυτό το βήμα ο χρήστης ξαναχτίζει τον κώδικα χρησιμοποιώντας τη σημαία -fprofile-instr-use για να ορίσει τα δεδομένα προφίλ που έχουν συλλεχθεί.

```
$ clang++ -O2 -fprofile-instr-use=code.profdata code.cc -o code
```

2.2.2 Clang Static Analyzer

Η Clang εκτός από βελτιστοποιήσεις και ελέγχους παρέχει για χρήση και κάποια εργαλεία. Από τα διάφορα εργαλεία αυτό που θα εξεταστεί είναι ο στατικός αναλυτής, καθώς είχε γίνει μία μικρή μελέτη στις ιδιότητές του για τους σκοπούς αυτής της διπλωματικής.

Ο στατικός αναλυτής της Clang [20] αναλύει τον πηγαίο κώδικα ενός προγράμματος σε C, C++, Objective-C και βρίσκει bugs. Βασικό τμήμα ενός αναλυτή είναι οι ελεγκτές που χρησιμοποιεί και που στη συγκεκριμένη περίπτωση χωρίζονται σε 3 κατηγορίες : default , πειραματικούς και debug.

Οι default ελεγκτές χωρίζονται σε 9 κατηγορίες. Θα εξεταστούν μερικές από αυτές. Οι πιο βασικοί ελεγκτές ανήκουν στην κατηγορία core, καθώς κάνουν ελέγχους σε βασικά χαρακτηριστικά της γλώσσας που χωρίς αυτά δεν γίνεται σωστή λειτουργία του προγράμματος. Μερικά σφάλματα που ελέγχουν είναι: λανθασμένη κλήση συναρτήσεων , διαίρεση με μηδέν , χρήση μηδενικών δεικτών σε ακατάλληλες συνθήκες, ανάθεση μη αρχικοποιημένων μεταβλητών σε πράξεις ή διακλαδώσεις, επιστροφή μη αρχικοποιημένων τιμών. Η επόμενη κατηγορία είναι η cplusplus που αφορά ελέγχους στη C++ , δηλαδή ελέγχους όπως διαρροές μνήμης ή προβλήματα χωρητικότητας με τις εντολές new και delete. Η κατηγορία deadcode ελέγχει για τιμές σε μεταβλητές που δεν ξαναδιαβάζονται μετά. Η κατηγορία optin κάνει ελέγχους για την απόδοση και για τους κανόνες κωδικοποίησης. Τέτοιοι έλεγχοι είναι : έλεγχος για μη αρχικοποίηση των πεδίων ενός αντικειμένου, έλεγχος για την κλήση εικονικών συναρτήσεων ή για τον κώδικα MPI. Η κατηγορία security περιλαμβάνει ελέγχους που σχετίζονται με την ασφάλεια και αυτό συνήθως απαιτεί ελέγχους όταν χρησιμοποιούνται συναρτήσεις όπως η bcopy, η bzero, η get, η rand ή η strcpy. Τέλος, η κατηγορία Unix περιλαμβάνει ελέγχους που έχουν σχέση με την κακή χρήση ή κλήση των συναρτήσεων της οικογένειας alloc (calloc, malloc, realloc), της open και της vfork.

Οι πειραματικοί ελεγκτές χαρακτηρίζονται από ελαττώματα και περιορισμούς και μπορεί να δώσουν ψευδείς πληροφορίες για τα αποτελέσματα των ελέγχων που πραγματοποιούν. Θα εξεταστούν μερικές από τις κατηγορίες τους. Η κατηγορία alpha.clone ελέγχει για παρόμοια κομμάτια κώδικα και για ανάθεση τιμών πέρα από 0 και 1 στις μεταβλητές αληθείας. Η κατηγορία alpha.core περιλαμβάνει μερικούς ελέγχους όπως: έλεγχο για λογικά σφάλματα στην κλήση συναρτήσεων , έλεγχο για έλλειψη ακριβείας σε μετατροπές , έλεγχο για ανάθεσης fixed διεύθυνσης σε δείκτη , έλεγχο για αφαίρεση δεικτών που δείχνουν σε διαφορετική θέση μνήμης. Η κατηγορία alpha.deadcode αφορά ελέγχους για

κώδικα που δεν θα εκτελεστεί ποτέ. Η κατηγορία alpha.security αφορά ελέγχους όπως: έλεγχος για υπερχείλιση στα πεδία της εντολές malloc, έλεγχος για κλήσεις της συνάρτησης mmap() που είναι και εκτελέσιμες και εγγράψιμες, έλεγχος για επιστροφή εκτός ορίων δείκτη. Τέλος η κατηγορία alpha.unix ελέγχει για την εσφαλμένη κλήση συναρτήσεων ή σφαλμάτων που προκύπτουν από τη λανθασμένη χρήση τους, όπως οι συναρτήσεις lock, pthread_mutex_lock, fopen, memcpy , strlen, και για προβλήματα από την ταξινόμηση ή όχι δεικτών σε επαναλήψεις.

Οι debug checkers χρησιμοποιούνται για να κάνουν debugging τον αναλυτή. Οι ελεγκτές μπορούν να καλέσουν κάποιες μεθόδους, όπως την εμφάνιση του γράφου κλήσεων ή τον έλεγχο της κατανόησης των εκφράσεων από τον αναλυτή.

2.3 Clang ως βιβλιοθήκη και Abstract Syntax Tree

Η Clang προσφέρει τις υποδομές για τη συγγραφή εργαλείων που παρέχουν συντακτικές και σημασιολογικές πληροφορίες για ένα πρόγραμμα. Υπάρχουν διάφορες μέθοδοι για τη δημιουργία clang εργαλείων και χρησιμοποιούνται σε διαφορετικές περιπτώσεις καθώς καθεμία διαθέτει συγκεκριμένες δυνατότητες και αδυναμίες.

Από τις δομές που μπορούν να υπολογίσουν τα εργαλεία της clang, αυτή στην οποία θα επικεντρωθεί η διπλωματική είναι το αφηρημένο συντακτικό δέντρο (Abstract Syntax Tree). Το AST είναι μια αναπαράσταση σε δέντρο της αφηρημένης συντακτικής δομής του πηγαίου κώδικα σε μία γλώσσα προγραμματισμού. Κάθε κόμβος του δέντρου αναπαριστά μία δομή που εμφανίζεται στον πηγαία κώδικα , όπως τις μεταβλητές , τις λογικές και αριθμητικές πράξεις , τις αναθέσεις και τις κλήσεις συναρτήσεων . Το δέντρο θεωρείται αφηρημένο με την έννοια ότι δεν συμπεριλαμβάνει ως κόμβους κάποια στοιχεία που δεν έχουν σχέση με τη δομή , όπως τις παρενθέσεις.

Θα εξεταστούν μερικές μέθοδοι για τη συγγραφή εργαλείων και η σχέση τους με το AST [21].

Το LibClang είναι μία υψηλού επιπέδου C διασύνδεση σε Clang. Τα πλεονεκτήματα του είναι ότι προσφέρει διασύνδεση σε Clang από άλλες γλώσσες πέρα από τη C++ και υψηλού επιπέδου αφηρημένες χρήσεις όπως τη διάσχιση του AST με κέρσορα. Το μειονέκτημά του είναι ότι δεν προσφέρει πλήρη έλεγχο στο AST.

Τα Clang Plugins επιτρέπουν την εκτέλεση επιπλέον πράξεων σε ένα AST ως μέρος της μεταγλώττισης. Αποτελούν δυναμικές βιβλιοθήκες που φορτώνονται κατά την εκτέλεση από τον μεταγλωττιστή και είναι εύκολο να εισαχθούν στο περιβάλλον χτισίματος. Τα Clang Plugins είναι χρήσιμα όταν το εργαλείο χρειάζεται να μπορεί να φτιάχνει ή να σταματάει ένα χτίσιμο ή όταν χρειάζεται πλήρης έλεγχος στο Clang AST. Τα μειονεκτήματά τους είναι ότι περιορίζουν το εργαλείο σε συγκεκριμένο εργαλείο χτισίματος και δεν προσφέρουν πλήρη έλεγχο για το πως θα οριστεί η Clang.

Το LibTooling αποτελεί μία C++ διασύνδεση που έχει σκοπό τη συγγραφή αυτόνομων εργαλείων, καθώς και την ενσωμάτωση σε υπηρεσίες που εκτελούν εργαλεία Clang. Τα πλεονεκτήματά του είναι ότι προσφέρει τη δυνατότητα για εκτέλεση εργαλείων πάνω σε συγκεκριμένα αρχεία ανεξαρτήτως του συστήματος χτισίματος , προσφέρει πλήρη έλεγχο στο Clang AST και μπορεί να μοιραστεί κώδικα με Clang Plugins. Οι αδυναμίες του είναι ότι δεν

αποτελεί μία σταθερή διασύνδεση οπότε επηρεάζεται αρνητικά από τις αλλαγές στα πρωτόκολλα και ότι προσφέρει μόνο τη δυνατότητα για συγγραφή σε C++.

Τα Clang Tools αποτελούν μία ομάδα εργαλείων που έχουν χτιστεί με βάση τις υποδομές του LibTooling. Τα πιο σημαντικά και αναπτυγμένα από αυτά είναι το clang-check που κάνει συντακτικό έλεγχο και το clang-format που κάνει αναδιοργάνωση του κώδικα.

2.3.1 Clang AST

Το χαρακτηριστικό του Clang AST [22] σε σχέση με AST από άλλους μεταγλωττιστές είναι ότι μοιάζει με τον κώδικα γραμμένο σε C++. Αυτό σημαίνει ότι μπορεί να συμπεριλάβει στο δέντρο εκφράσεις όπως τις παρενθέσεις και τις χρονικές σταθερές της μεταγλώττισης σε μη περιεκτική μορφή και αυτό είναι ιδανικό για εργαλεία επανασχεδιασμού.

Η Clang έχει τη δυνατότητα να εμφανίσει το AST χρησιμοποιώντας τη σημαία -ast-dump όπως φαίνεται στην παρακάτω εικόνα:

```
$ cat test.cc
int f(int x) {
  int result = (x / 42);
  return result;
}

# Clang by default is a frontend for many tools; -Xclang is used to pass
# options directly to the C++ frontend.
$ clang -Xclang -ast-dump -fsyntax-only test.cc
TranslationUnitDecl 0x5aea0d0 <<invalid sloc>>
... cutting out internal declarations of clang ...
`-FunctionDecl 0x5aeab50 <test.cc:1:1, line:4:1> f 'int (int)'
  |-ParmVarDecl 0x5aeaa90 <line:1:7, col:11> x 'int'
  `-CompoundStmt 0x5aead88 <col:14, line:4:1>
    |-DeclStmt 0x5aead10 <line:2:3, col:24>
    | `-VarDecl 0x5aeac10 <col:3, col:23> result 'int'
    |   `-ParenExpr 0x5aeacf0 <col:16, col:23> 'int'
    |     `-BinaryOperator 0x5aeacc8 <col:17, col:21> 'int' '/'
    |       |-ImplicitCastExpr 0x5aeacb0 <col:17> 'int' <LValueToRValue>
    |       | `-DeclRefExpr 0x5aeac68 <col:17> 'int' lvalue ParmVar 0x5aeaa90 'x' 'int'
    |       `-IntegerLiteral 0x5aeac90 <col:21> 'int' 42
    `-ReturnStmt 0x5aead68 <line:3:3, col:10>
      `-ImplicitCastExpr 0x5aead50 <col:10> 'int' <LValueToRValue>
        `-DeclRefExpr 0x5aead28 <col:10> 'int' lvalue Var 0x5aeac10 'result' 'int'
```

Σχήμα 4.8 : Abstract syntax tree ενός προγράμματος

Η κορυφαία δήλωση(declaration) στη μονάδα μετάφρασης είναι πάντα η δήλωση της μονάδας μετάφρασης. Στο συγκεκριμένο παράδειγμα, το πρώτο declaration στον κώδικα του χρήστη είναι η δήλωση της συνάρτησης ‘f’ . Το σώμα της συνάρτησης f είναι ένα compound statement , που έχει ως κόμβους παιδιά το declaration statement που δηλώνει την μεταβλητή result και το return statement.

Οι δύο πιο βασικοί τύποι κόμβου στο Clang AST είναι τα statements και τα declarations. Ωστόσο, οι κόμβοι σε ένα τέτοιο δέντρο μοντελοποιούνται πάνω σε μία ιεραρχία κλάσεων που δεν έχουν κάποιο κοινό πρόγονο. Υπάρχουν πολλαπλές μεγαλύτερες ιεραρχίες που προκύπτουν από βασικούς κόμβους. Πολλοί σημαντικοί κόμβοι προκύπτουν από τύπους όπως Type, Decl, DeclContext ή Stmt. Φυσικά, υπάρχουν και κόμβοι οι οποίοι δεν ανήκουν σε κάποια μεγαλύτερη ιεραρχία και μπορεί να γίνει πρόσβαση σε αυτούς μόνο από συγκεκριμένους άλλους κόμβους.

2.4 clang.cindex

Η προηγηθείσα μελέτη της Clang έχει σκοπό να κάνει κατανοητό το σύνολο των ιδιοτήτων και των διαφόρων χρήσεων της, καθώς για τις μετρήσεις της διπλωματικής θα χρησιμοποιηθεί μία βιβλιοθήκη της.

Η βιβλιοθήκη που θα χρησιμοποιηθεί είναι η clang.cindex.py [23] η οποία αποτελεί ένα μοντέλο για τη διασύνδεση με τη Clang Indexing βιβλιοθήκη σε python . Μεταξύ πολλών άλλων λειτουργιών μπορεί να χρησιμοποιηθεί για να κάνει parsing σε ένα πρόγραμμα και να δημιουργήσει ένα AST. Η βιβλιοθήκη είναι αρκετές γραμμές κώδικα , οπότε θα εξεταστούν μόνο τα κύρια στοιχεία και όσα αφορούν στο AST.

Το πιο σημαντικό αντικείμενο είναι το Index που λειτουργεί ως η βασική σύνδεση με τη Clang index βιβλιοθήκη κυρίως προσφέροντας διασύνδεση για την ανάγνωση και το parsing μονάδων μετάφρασης. Από τις μεθόδους αυτές είναι προφανές ότι χρησιμοποιούνται στη διπλωματική η μέθοδος create που δημιουργεί ένα καινούριο αντικείμενο και η μέθοδος parse.

Η μέθοδος parse δέχεται τις εξής παραμέτρους (self, path, args=None, unsaved_files=0, options = 0). Με τη μέθοδο αυτή φορτώνεται μία μονάδα μετάφρασης (Translation Unit) από το δοσμένο αρχείο πηγαίου κώδικα στο path αφού έχει εκτελεστεί clang και παραχθεί ένα AST πριν τη φόρτωση. Επιπλέον παράμετροι μπορούν να περαστούν από το command line μέσω της παραμέτρου args. Η παράμετρος unsaved_files προσφέρει εντός μνήμης περιεχόμενα για αρχεία περνώντας μία λίστα από ζεύγη. Το πρώτο μέλος του ζεύγους είναι τα ονόματα των αρχείων που θα αντιστοιχηθούν και το δεύτερο τα περιεχόμενα που θα αντικατασταθούν. Αυτή η μέθοδος επιστρέφει τη μονάδα μετάφρασης του πηγαίου αρχείου.

Το αντικείμενο TranslationUnit είναι υψηλού επιπέδου αντικείμενο που περιλαμβάνει το AST για μία μόνο μονάδα μετάφρασης. Μπορεί να προκύψει από αρχεία .ast ή κατευθείαν κάνοντας parsing.

Η πρώτη μέθοδός του που θα αναλυθεί είναι η from_source(cls, filename, args=None, unsaved_files=None, options=0, index=None) που παράγει ένα αντικείμενο TranslationUnit κάνοντας parsing. Το αντικείμενο αυτό είναι που επιστρέφεται από τη μέθοδο parse σε αντικείμενα Index. Επιπλέον ορίσματα από το command line μπορούν να περαστούν σε clang μέσω της παραμέτρου args. Περιεχόμενα εντός μνήμης αρχείων μπορούν να περαστούν μέσω του unsaved_files που αποτελείται από λίστα ζευγαριών. Το πρώτο στοιχείο είναι το όνομα του αρχείου και το δεύτερο το περιεχόμενο του αρχείου. Η επιλογή options είναι μία σημαία που ελέγχει τη συμπεριφορά parsing. Η επιλογή Index δίνει το αντικείμενο Index στο οποίο θα αντιστοιχηθεί το TranslationUnit. Αν δεν υπάρχει η μέθοδος δημιουργεί ένα καινούριο instance Index.

Από τις άλλες μεθόδους χρησιμοποιείται μόνο η μέθοδος cursor , η οποία επιστρέφει έναν κέρσορα με τη μορφή του αντικειμένου Cursor ο οποίος αντιστοιχεί στο συγκεκριμένο TranslationUnit.

Το αντικείμενο Cursor χρησιμοποιείται για την αναπαράσταση ενός κόμβου σε ένα AST και μπορεί να λειτουργήσει σαν ένα είδος μετρητή. Αυτό το αντικείμενο διαθέτει αρκετές μεθόδους , οπότε θα αναλυθούν μερικές από αυτές.

Η μέθοδος location δέχεται ως παράμετρο το ίδιο το αντικείμενο και επιστρέφει την πηγαία τοποθεσία της οντότητας που δείχνει ο κέρσορας. Αυτό σημαίνει ότι επιστρέφει ένα αντικείμενο τύπου SourceLocation.

Η μέθοδος kind επιστρέφει τον τύπο του κόμβου που δείχνει ο κέρσορας, όπως για παράδειγμα τύπο statement.

Αν ο κέρσορας κάνει αναφορά σε ένα declaration ή είναι ένα declaration σε μία οντότητα, τότε η μέθοδος get_definition θα επιστρέψει έναν κέρσορα που δείχνει στον ορισμό αυτής της οντότητας. Ένα είδος οντότητας που μπορεί να χρησιμοποιηθεί αυτό είναι οι συναρτήσεις.

Η μέθοδος displayname επιστρέφει το display όνομα για την οντότητα που αναφέρεται από τον κέρσορα. Το display όνομα περιέχει πληροφορίες που βοηθούν στον προσδιορισμό του κέρσορα, όπως οι παράμετροι σε μία συνάρτηση.

Η μέθοδος extent επιστρέφει το source range (το εύρος του κειμένου) που καταλαμβάνεται από την οντότητα που δείχνει ο κέρσορας. Πιο συγκεκριμένα, επιστρέφει ένα αντικείμενο τύπου SourceRange.

Η μέθοδος get_children επιστρέφει έναν μετρητή (iterator) για την πρόσβαση στα παιδιά του κέρσορα. Πιο συγκεκριμένα, ο iterator αποτελεί μία σειρά κόμβων που είναι παιδιά στον κόμβο που δείχνει ο κέρσορας. Αυτά τα παιδιά μπορούν να αντιμετωπιστούν σαν αντικείμενα Cursor.

Το αντικείμενο SourceRange αντιπροσωπεύει πληροφορίες για το αρχείο πηγαίου κώδικα. Πιο συγκεκριμένα, περιγράφει ένα εύρος από πηγαίες τοποθεσίες μέσα στον πηγαίο κώδικα.

Η μέθοδος start επιστρέφει ένα αντικείμενο SourceLocation που αντιπροσωπεύει τον πρώτο χαρακτήρα μέσα στο εύρος τοποθεσιών.

Η μέθοδος end επιστρέφει ένα αντικείμενο SourceLocation που αντιπροσωπεύει τον τελευταίο χαρακτήρα μέσα στο εύρος τοποθεσιών.

Το αντικείμενο SourceLocation , όπως και το SourceRange, δίνει πληροφορίες για το αρχείο πηγαίου κώδικα. Αντιπροσωπεύει μία συγκεκριμένη τοποθεσία μέσα στον πηγαίο κώδικα.

Η μέθοδος file επιστρέφει το αρχείο που αντιπροσωπεύεται από την πηγαία τοποθεσία.

Η μέθοδος line επιστρέφει τη σειρά που αντιπροσωπεύεται από την πηγαία τοποθεσία.

Το αντικείμενο CursorKind περιγράφει το είδος της οντότητας που δείχνει ο κέρσορας. Οι περισσότερες μέθοδοι κάνουν έλεγχο αν η οντότητα είναι κάποιου συγκεκριμένου τύπου , όπως για παράδειγμα αν είναι declaration συνάρτησης ή IF statement. Η χρήση της μορφής clang.cindex.CursorkKind.* όπου * μία οντότητα ,δηλώνει αυτό το είδος οντότητας.

Αυτά τα αντικείμενα μπορούν να δώσουν αναλυτικές πληροφορίες για τον κώδικα που εξετάζεται. Για παράδειγμα, αν το node είναι αντικείμενο Cursor , τότε το

node.extent.start.line δίνει τη γραμμή που εμφανίζεται ο πρώτος χαρακτήρας για την οντότητα που περιγράφει ο κέρσορας, δηλαδή τη πρώτη γραμμή της οντότητας στο κείμενο. Αυτές οι πληροφορίες θα είναι χρήσιμες για τις μετρήσεις της διπλωματικής.

3. Regression

Στη στατιστική μοντελοποίηση, ο όρος regression ανάλυση χρησιμοποιείται για να ορίσει ένα σύνολο διαδικασιών που υπολογίζουν τις σχέσεις μεταξύ μίας εξαρτημένης μεταβλητής και μίας ή περισσότερων ανεξάρτητων μεταβλητών. Από τις μεθόδους regression η πιο συνηθισμένη είναι το γραμμικό regression , όπου ο ερευνητής βρίσκει τη γραμμική συνάρτηση που ταιριάζει όσο πιο στενά γίνεται τα δεδομένα σύμφωνα με ένα συγκεκριμένο μαθηματικό κριτήριο. Για παράδειγμα, η μέθοδος των ελαχίστων τετραγώνων [24] υπολογίζει τη μοναδική γραμμή που ελαχιστοποιεί το άθροισμα των τετραγωνικών αποστάσεων μεταξύ των αληθινών δεδομένων και αυτής της γραμμής. Για συγκεκριμένες μαθηματικές συνθήκες αυτό επιτρέπει στον ερευνητή να ορίσει υπό συνθήκη προσδοκίες για την τιμή της εξαρτημένης μεταβλητής όταν οι ανεξάρτητες μεταβλητές παίρνουν ένα δοσμένο σύνολο τιμών. Επομένως η regression ανάλυση χρησιμοποιείται σε μεγάλο βαθμό για την πρόβλεψη τιμών κάτι που επικαλύπτεται με το πεδίο της μηχανικής μάθησης. Σημαντικό είναι να σημειωθεί ότι αυτή η ανάλυση προβλέπει εξόδους σε συγκεκριμένο σύνολο δεδομένων, οπότε πρέπει να δικαιολογείται γιατί οι υπάρχουσες σχέσεις οδηγούν σε γενικότερες προβλέψεις.

3.1 Μοντέλο Regression

Πρακτικά, οι ερευνητές επιλέγουν ένα μοντέλο που θέλουν να υπολογίσουν και μετά χρησιμοποιούν την μέθοδο της επιλογής τους για να υπολογίσουν τις παραμέτρους του μοντέλου. Τα μοντέλα regression περιλαμβάνουν τα ακόλουθα δομικά στοιχεία:
Τις άγνωστες παραμέτρους που συχνά υποδηλώνονται ως βαθμωτό μέγεθος ή διάνυσμα β.
Τις ανεξάρτητες μεταβλητές που παρατηρούνται στα δεδομένα και υποδηλώνονται ως ένα διάνυσμα X_i , όπου i υποδηλώνει μία γραμμή των δεδομένων.
Την εξαρτημένη μεταβλητή που παρατηρείται στα δεδομένα και εκφράζεται με το διάνυσμα Y_i.
Τα σφάλματα που δεν παρατηρούνται στα δεδομένα και εκφράζονται με το βαθμωτό μέγεθος e_i.

Τα περισσότερα μοντέλα προτείνουν ότι το διάνυσμα Y_i αποτελεί συνάρτηση των μεγεθών X_i και β , ενώ το διάνυσμα e_i αποτελεί πρόσθετο σφαλματικό όρο και χρησιμοποιείται ως τυχαίος στατιστικός θόρυβος. Αυτό εκφράζεται με τη μορφή $Y_i = f(X_i, \beta) + e_i$.

Σκοπός των ερευνητών είναι ο υπολογισμός της συνάρτησης $f(X_i, \beta)$ που ταιριάζει πιο κοντά στα δεδομένα και για αυτό χρειάζεται να προσδιοριστεί ο τύπος της συνάρτησης f. Μερικές φορές ο τύπος της συνάρτησης εξαρτάται από γνώση των σχέσεων μεταξύ των X_i και Y_i που δεν βασίζεται στα δεδομένα, αλλά όταν δεν υπάρχει τέτοια γνώση επιλέγεται ένας ευέλικτος ή βολικός τύπος για την f. Μια απλή επιλογή είναι η ευθεία , δηλαδή $f(X_i, \beta) = \beta_0 + \beta_1 X_1$.

Όταν ο ερευνητής επιλέξει το στατιστικό μοντέλο, διαφορετικές μορφές regression ανάλυσης παρέχουν εργαλεία για τον υπολογισμό των παραμέτρων β. Στη μέθοδο ελαχίστων τετραγώνων υπολογίζει τις τιμές του β για τις οποίες το άθροισμα των τετραγωνικών σφαλμάτων ελαχιστοποιείται, δηλαδή $\min \sum_i (Y_i - f(X_i, \beta))^2$. Μια δοσμένη μέθοδος θα δώσει μία εκτίμηση για την παράμετρο β , η οποία συμβολίζεται με $\hat{\beta}$ για να την ξεχωρίζει από τη γνήσια. Χρησιμοποιώντας αυτήν την εκτίμηση ο ερευνητής μπορεί να χρησιμοποιήσει την ταιριασμένη τιμή $\hat{Y} = f(X_i, \hat{\beta})$ για πρόβλεψη ή για αξιολόγηση της ακρίβειας του μοντέλου για την επεξήγηση των δεδομένων. Και οι εκτιμήσεις και η προβλεπόμενη τιμή μπορεί ή όχι να χρησιμοποιηθούν για τους σκοπούς της μελέτης.

Σημαντικό είναι να τονιστεί ότι για τον ορισμό των εκτιμήσεων απαραίτητο είναι να υπάρχουν επαρκή δεδομένα. Πιο συγκεκριμένα, στη μέθοδο των ελαχίστων τετραγώνων ο αριθμός των δειγμάτων που υπάρχουν πρέπει να είναι μεγαλύτερος ή ίσος από τον αριθμό των παραμέτρων που αναζητούνται, δηλαδή τα β, διαφορετικά θα μπορούν να δοθούν άπειρες λύσεις και το σύστημα δεν θα είναι ντετερμινιστικό. Στην περίπτωση που είναι μεγαλύτερος δεν θα υπάρχει κάποιο σύνολο που ταιριάζει τέλεια με τα δεδομένα. Επίσης για τη σωστή λειτουργία του παραπάνω μοντέλου πρέπει οι ανεξάρτητες μεταβλητές X_i να είναι γραμμικά ανεξάρτητες μεταξύ τους, δηλαδή καμία από αυτές δεν μπορεί να δημιουργηθεί από πράξεις μεταξύ των υπολοίπων.

Φυσικά για να μπορεί να χρησιμοποιηθεί η έξοδος από ένα regression ως μία στατιστική ποσότητα για μετρήσεις σε πραγματικά προβλήματα, οι ερευνητές κάνουν κάποιες υποθέσεις που θεωρούν ότι είναι δεδομένες. Αυτό περιλαμβάνει:
Το δείγμα είναι αντιπροσωπευτικό του πιο γενικού πληθυσμού.
Οι ανεξάρτητες μεταβλητές μετρούνται χωρίς σφάλμα.
Αποκλίσεις από το μοντέλο έχουν προσδοκώμενη τιμή 0.
Η διασπορά των υπολοίπων e_i παραμένει σταθερή μέσα στις παρατηρήσεις.
Τα υπόλοιπα e_i είναι ασυσχέτιστα μεταξύ τους.
Ωστόσο, αυτές οι γενικεύσεις δεν πρέπει να χρησιμοποιούνται ασύστολα καθώς σε πολλά πραγματικά πλαίσια χάνουν το νόημά τους.

3.2 Γραμμικό Regression

Δοθέντος ενός συνόλου δεδομένων, το γραμμικό regression μοντέλο [25] θεωρεί ότι η σχέση μεταξύ της εξαρτημένης μεταβλητής y και των ανεξάρτητων μεταβλητών x είναι γραμμική. Θεωρώντας ως ε το θόρυβο στη γραμμική σχέση, το μοντέλο για i γραμμές και p μεταβλητές x σε κάθε γραμμή παίρνει τη μορφή:
$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i$, για i = 1,2,.....,n

3.2.1 Γενικεύσεις

Όπως και στα άλλα μοντέλα γίνονται γενικεύσεις για τη χρήση τους σε πραγματικά πλαίσια. Εκτός από αυτές που έχουν ήδη αναφερθεί , για το γραμμικό μοντέλο γίνονται και κάποιες επιπλέον. Μία από αυτές είναι η γραμμικότητα, δηλαδή ο αριθμητικός μέσος της μεταβλητής εξόδου αποτελεί γραμμικό συνδυασμό των παραμέτρων και των μεταβλητών προβλέψεων κάτι που αποτελεί περιορισμό κυρίως στις παραμέτρους. Οι μεταβλητές πρόβλεψης μπορούν να μεταμορφωθούν αυθαιρέτως και πολλαπλές αντιγραφές της ίδιας μεταβλητής πρόβλεψης μπορούν να προστεθούν, η καθεμία μεταμορφωμένη διαφορετικά. Αυτή η τεχνική χρησιμοποιείται από το πολυωνυμικό regression που χρησιμοποιεί γραμμικό regression για να ταιριάξει την έξοδο σε μία πολυωνυμική συνάρτηση της μεταβλητής πρόβλεψης. Αυτή η χρήση του γραμμικού regression μπορεί να προκαλέσει υπερταίριασμα (overfitting) των δεδομένων, δηλαδή οι προβλέψεις κινούνται σε ένα σύνολο δεδομένων και οποιαδήποτε αλλαγή στο γενικό σύνολο δε θα επηρεάσει τις προβλέψεις. Για την αποφυγή αυτού του φαινομένου πρέπει να χρησιμοποιηθεί κάποιου είδους κανονικοποίηση, δηλαδή μετατροπή των τιμών των ανεξάρτητων μεταβλητών σε ένα πιο περιορισμένο πλαίσιο, κάτι που απορρίπτει ακραίους υπολογισμούς.

Η άλλη γενίκευση που γίνεται είναι ότι υπάρχει έλλειψη τέλειας πολυσυγγραμμικότητας μεταξύ των μεταβλητών πρόβλεψης. Αυτό παρατηρείται όταν υπάρχουν δύο ή περισσότερες μεταβλητές πρόβλεψης που είναι τέλεια συσχετισμένες μεταξύ τους και όπως αναφέρθηκε όταν υπάρχει σχετικά μικρός αριθμός δεδομένων σε σχέση με τις

παραμέτρους που μετρούνται. Σε κάθε περίπτωση το διάνυσμα παραμέτρων β δε θα γίνεται να προσδιοριστεί.

Πέρα από τις γενικεύσεις υπάρχουν και στατιστικές ιδιότητες των δεδομένων που βοηθούν στην απόδοση των μεθόδων υπολογισμού. Η στατιστική σχέση μεταξύ των σφαλμάτων και των regressors παίζει σημαντικό ρόλο για να αποφασίζει αν η διαδικασία υπολογισμού έχει επιθυμητές ιδιότητες όπως συνέπεια και αμεροληψία. Η πιθανοτική κατανομή των μεταβλητών πρόβλεψης x έχει μεγάλη επιρροή για την ακρίβεια των εκτιμήσεων β και μέθοδοι όπως η δειγματοληψία προσφέρουν τρόπους για συλλογή δεδομένων που επιτυγχάνουν αυτόν τον στόχο.

Ένα ταιριασμένο γραμμικό regression μοντέλο μπορεί να προσδιορίσει τη σχέση ανάμεσα σε μία μοναδική μεταβλητή πρόβλεψης x και τη μεταβλητή εξόδου y όταν όλες οι υπόλοιπες μεταβλητές πρόβλεψης παραμένουν "φιξαρισμένες". Αυτή η έννοια σημαίνει ότι για κάποια τιμή μίας μεταβλητής κάποια υποσύνολα των δεδομένων έχουν κοινές τιμές, οπότε αυτή η μεταβλητή δεν τα επηρεάζει. Όταν εξετάζεται η σχέση μεταξύ μεταβλητής πρόβλεψης x και εξόδου y, τότε αν αλλαγές στις τιμές της μεταβλητής επηρεάζουν ανάλογα τις τιμές την εξόδου θα αναφέρεται ότι το y είναι παράγωγο με βάση το x. Μπορεί φυσικά μεγάλες αλλαγές στις τιμές μιας μεταβλητής να μην επηρεάζουν σημαντικά την έξοδο , οπότε η μεταβλητή αυτή δεν συνεισφέρει στη διασπορά της εξόδου.

3.2.2. Επεκτάσεις

Έχουν αναπτυχθεί κάποιες επεκτάσεις για το γραμμικό regression μοντέλο που επιτρέπουν τις γενικεύσεις για το απλό μοντέλο να χαλαρώσουν. Η πιο απλή επέκταση έχει ήδη αναφερθεί και αφορά στο αν χρησιμοποιείται μόνο μία μεταβλητή πρόβλεψης ή περισσότερες. Η αμέσως επόμενη επέκταση αφορά στα γενικά γραμμικά μοντέλα όπου η μεταβλητή εξόδου είναι ένα διάνυσμα και για αυτή έχει αναπτυχθεί η μέθοδος των γενικευμένων ελαχίστων τετραγώνων.

Οι επόμενες επεκτάσεις είναι πιο περίπλοκες:

Έχουν φτιαχτεί μοντέλα που επιτρέπουν στα σφάλματα των διαφορετικών μεταβλητών εξόδου να έχουν διαφορετικές τιμές διασπορών και για αυτές τις περιπτώσεις μπορεί να χρησιμοποιηθεί η μέθοδος των ελαχίστων τετραγώνων με βάρη.

Τα γενικευμένα γραμμικά μοντέλα αποτελούν ένα πλαίσιο για τη μοντελοποίηση των μεταβλητών εξόδου που είναι φραγμένες ή διακριτές. Αυτό χρησιμοποιείται στις συγκεκριμένες περιπτώσεις:

- Όταν μοντελοποιούν θετικές ποσότητες που ποικίλουν σε μία μεγάλη κλίμακα , μεγέθη που περιγράφονται καλύτερα με κατανομές όπως η Poisson.
- Όταν μοντελοποιούν κατηγορικά δεδομένα, όπως η επιλογή ενός υποψηφίου σε μία εκλογή , όπου υπάρχει συγκεκριμένος αριθμός επιλογών (περιγράφονται καλύτερα με κατανομή bernoulli).
- Όταν μοντελοποιούν τακτικά δεδομένα , όπως βαθμολογίες σε κλίμακα από 0 σε 5, όπου οι διαφορετικές έξοδοι μπορούν να ταξινομηθούν αλλά η ποσότητα δεν έχει απόλυτη νόημα.

Μερικά μοντέλα για συγκεκριμένες περιπτώσεις είναι: το Poisson regression για δεδομένα μετρήσεων, το λογιστικό και πιθανομονάδας regression για δυαδικά δεδομένα, το πολυωνυμικό λογιστικό και πιθανομονάδας regression για κατηγορικά δεδομένα και το ταξινομημένο logit και πιθανομονάδας regression για τακτικά δεδομένα.

Τα ιεραρχικά γραμμικά μοντέλα οργανώνουν τα δεδομένα σε μία ιεραρχία regressions , όπου για παράδειγμα το A γίνεται regressed στο B και το B γίνεται regressed στο C. Αυτό χρησιμοποιείται όταν οι μεταβλητές ενδιαφέροντος έχουν μία φυσική ιεραρχική δομή, όπως σε εκπαιδευτικά στατιστικά όπου οι μαθητές είναι φωλιασμένοι σε τάξεις και οι τάξεις σε σχολεία.

Τα μοντέλα σφάλματα-σε-μεταβλητές επεκτείνουν το κλασικό γραμμικό μοντέλο για να επιτρέπει στις μεταβλητές πρόβλεψης X να παρατηρούνται με σφάλμα. Αυτό μπορεί να επηρεάσει την αντικειμενικότητα των εκτιμήσεων για το β αν και όχι σημαντικά.

3.2.3 Εκτιμήσεις

Η πιο βασική εκτίμηση είναι η μέθοδος των ελαχίστων τετραγώνων. Θεωρώντας ότι η μεταβλητή πρόβλεψης είναι ένα διάνυσμα $\vec{x} = [x_1,x_2,\ldots\ldots,x_m]$ και οι παράμετροι του μοντέλου είναι $\vec{\beta} = [\beta_0,\beta_1,\beta_2,\ldots\ldots,\beta_m]$ τότε η πρόβλεψη για την έξοδο είναι $y \approx \beta_0 + \sum_1^m \beta_i \times x_i$. Αν το $\vec{x}$ επεκταθεί σε $\vec{x} = [1, x_1,x_2,\ldots\ldots,x_m]$ τότε η πρόβλεψη γίνεται $y \approx \sum_0^m \beta_i \times x_i$. Στην περίπτωση των ελαχίστων τετραγώνων , η βέλτιστη παράμετρος ορίζεται έτσι ώστε να ελαχιστοποιεί το άθροισμα των ελαχίστων διαφορών (απωλειών) , δηλαδή:

$\hat{\vec{\beta}} = \arg\min_{\vec{\beta}} L(D,\vec{\beta}) = \arg\min_{\vec{\beta}} \sum_{i=1}^{n} (\vec{\beta} \cdot \overrightarrow{x_i} - y_i)^2$

Τοποθετώντας τις μεταβλητές πρόβλεψης και την έξοδο σε πίνακες με αντίστοιχα ονόματα X και Y, η συνάρτηση απώλειας μπορεί να γραφτεί:

$L(D, \vec{\beta}) = \left\|X\vec{\beta} - Y\right\|^2 = (X\vec{\beta} - Y)^T(X\vec{\beta} - Y) = Y^TY - Y^TX\vec{\beta} - \vec{\beta}^TX^TY + \vec{\beta}^TX^TX\vec{\beta}$

Καθώς η συνάρτηση είναι κυρτή η βέλτιστη λύση βρίσκεται εκεί όπου η παράγωγος είναι μηδενική και η οποία είναι :

$\frac{\partial L(D,\vec{\beta})}{\partial \vec{\beta}} = \frac{\partial (Y^TY - Y^TX\vec{\beta} - \vec{\beta}^TX^TY - \vec{\beta}^TX^TX\vec{\beta})}{\partial \vec{\beta}} = -2Y^TX + 2\vec{\beta}^TX^TX$

Θέτοντας την παράγωγο ίση με 0 προκύπτει:

$-2YX + 2\vec{\beta}^TX^TX = 0$ και μετά από πράξεις $\hat{\vec{\beta}} = (X^TX)^{-1}X^TY$

Κάποιες τεχνικές που έχουν συσχέτιση μεταξύ τους είναι οι παρακάτω:

Η εκτίμηση μέγιστης πιθανότητας πραγματοποιείται όταν η κατανομή των σφαλμάτων ανήκει σε μία συγκεκριμένη παραμετρική οικογένεια f_θ κατανομών πιθανότητας [26]. Όταν η f_θ είναι κανονική κατανομή με μηδενικό μέσο και διασπορά θ , τότε η τελική εκτίμηση είναι παρόμοια με την εκτίμηση ελαχίστων τετραγώνων.

Το Ridge regression [27] και το Lasso Regression [28] είναι μέθοδοι εκτίμησης με ποινές και εισάγουν μία κλίση στην εκτίμηση του β έτσι ώστε να μειωθεί η μεταβλητότητα της εκτίμησης. Χρησιμοποιούνται κυρίως για να προβλέψουν την τιμή της εξόδου y για τιμές των μεταβλητών πρόβλεψης που δεν έχουν ακόμα παρατηρηθεί.

Η ελάχιστη απόλυτη απόκλιση είναι μια μέθοδος εύρωστης εκτίμησης που είναι λιγότερο ευαίσθητη από τη μέθοδο των ελάχιστων τετραγώνων σε εξωτερικές επιρροές και ισοδυναμεί με εκτίμηση μέγιστης πιθανότητας πάνω σε μοντέλο κατανομής Laplace για σφάλμα e [29].

Σε πιο γενική κατηγορία εκτιμήσεων ανήκουν τα παρακάτω:

Το Bayesian γραμμικό regression εφαρμόζει την Bayesian στατιστική στο γραμμικό regression , δηλαδή οι παράμετροι β θεωρούνται τυχαίες μεταβλητές με συγκεκριμένη κατανομή. Η κατανομή δίνει μία κλίση στις λύσεις για τις παραμέτρους και παράγει μία κατανομή τους , όπου χρησιμοποιώντας μετρικές μπορεί να υπολογιστούν οι καλύτερες παράμετροι.

Το regression ποσοστημορίου εστιάζει σε εξαρτημένα ποσοστημόρια της εξόδου y δοσμένου του συνόλου X και ως συνάρτηση χρησιμοποιεί τον εξαρτημένο μέσο.

Τα ανάμεικτα μοντέλα χρησιμοποιούνται για να αναλύσουν γραμμικές σχέσεις regression όταν υπάρχουν εξαρτημένα δεδομένα με τις εξαρτήσεις να έχουν γνωστή δομή. Κοινές χρήσεις τους αποτελούν η ανάλυση δεδομένων τα οποία περιλαμβάνουν επαναλαμβανόμενες

μετρήσεις και γενικά λειτουργούν ως παραμετρικά μοντέλα, που μπορούν να χρησιμοποιούν εκτίμηση μέγιστης πιθανότητας και Bayesian γραμμικό regression.
Το regression κυρίως τμήματος [30] χρησιμοποιείται όταν υπάρχουν πολλές μεταβλητές πρόβλεψης ή όταν οι μεταβλητές έχουν πολύ δυνατές συσχετίσεις. Πρώτα εφαρμόζει ανάλυση κυρίου τμήματος , η οποία μειώνει τις μεταβλητές σε βαθμό που δεν έχουν πολλές σχέσεις , και μετά εφαρμόζει μέθοδο ελαχίστων τετραγώνων με τις μεταβλητές που έχουν απομείνει.

3.3 Μη γραμμικό Regression

Το μη γραμμικό regression είναι μία μορφή ανάλυσης regression όπου τα δεδομένα μοντελοποιούνται από μία συνάρτηση που αποτελεί μη γραμμικό συνδυασμό των παραμέτρων του μοντέλου.

Όπως και στο γραμμικό regression η μεταβλητή εξόδου συνδέεται με τις μεταβλητές πρόβλεψης και τις παραμέτρους του μοντέλου μέσω μίας μη γραμμικής συνάρτησης f. Για παράδειγμα η συνάρτηση

$f(x,\beta) = \frac{\beta_1 x}{\beta_2 + x}$ δεν είναι γραμμική .

Στην κατηγορία των μη γραμμικών συναρτήσεων ανήκουν συναρτήσεις όπως οι εκθετικές συναρτήσεις , οι λογαριθμικές , οι τριγωνομετρικές, οι συναρτήσεις δυνάμεων και οι συναρτήσεις Gauss. Κάποιες από αυτές μπορούν να μετατραπούν σε γραμμικές και έτσι μπορεί να εφαρμοστεί κλασικό γραμμικό regression. Γενικά δεν υπάρχει κλειστής μορφής έκφραση για τις παραμέτρους καλύτερου ταιριάσματος και συνήθως για να επιτευχθεί αυτό χρησιμοποιούνται αλγόριθμοι βελτιστοποίησης. Καθώς μπορεί να υπάρχουν πολλά τοπικά ελάχιστα για τη συνάρτηση που θα βελτιστοποιηθεί και το ολικό να μη δίνει αντικειμενική εκτίμηση , χρησιμοποιούνται εκτιμημένες τιμές των παραμέτρων μαζί με τον αλγόριθμο για να βρεθεί το ολικό ελάχιστο του αθροίσματος τετραγώνων.

3.3.1 Μη γραμμικά ελάχιστα τετράγωνα

Έστω μία κυρτή συνάρτηση $f(x,\beta)$ που εκτός από τις μεταβλητές πρόβλεψης x , m σε αριθμό , εξαρτάται και από τις παραμέτρους β, n σε αριθμό. Οι καλύτερες τιμές των παραμέτρων προκύπτουν όταν το ακόλουθο άθροισμα τετραγώνων ελαχιστοποιείται : $S = \sum_1^m r_i^2$, όπου το μέγεθος r_i περιγράφεται από τη σχέση : $r_i = y_i - f(x_i,\beta)$ για i = 1,2,…,m.
Οι καλύτερες παράμετροι είναι αυτές για τις οποίες η παράγωγος του αθροίσματος γίνεται μηδέν:

$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j}$ για j = 1,2,…..,n.

Στα μη γραμμικά μοντέλα οι παράγωγοι $\frac{\partial r_i}{\partial \beta_j}$ είναι συναρτήσεις των ανεξάρτητων μεταβλητών και των παραμέτρων και επομένως δεν έχουν κάποια κλειστή λύση. Για να υπολογιστούν οι παράμετροι τους δίνεται μια αρχικοποίηση και μετά υπολογίζονται με διαδοχικές προσεγγίσεις: $\beta_j \approx \beta_j^{k+1} = \beta_j^k + \Delta\beta_j$. Το k αποτελεί τον αριθμό της επανάληψης και το διάνυσμα αυξήσεων $\Delta\beta$ είναι γνωστό ως διάνυσμα ολίσθησης. Σε κάθε επανάληψη το μοντέλο γραμμικοποιείται από προσέγγιση σε ένα πρώτης τάξης πολυώνυμο σχετικά με το β^k:

$f(x_i,\beta) \approx f(X_i,\beta^k) + \sum_j \frac{\partial f(x_i,\beta^k)}{\partial \beta_j}(\beta_j - \beta_j^k) = f(x_i,\beta^k) + \sum_j J_{ij}\Delta\beta_j$

Η ιακωβιανή J είναι μία συνάρτηση σταθερών των ανεξάρτητων μεταβλητών και των παραμέτρων και αλλάζει σε κάθε επανάληψη. Σε όρους γραμμικού μοντέλου,

$\frac{\partial r_i}{\partial \beta_j}$ = -J_{ij} και τα υπόλοιπα δίνονται από:

$\Delta y_i = y_i - f(xi, \beta^k)$

$r_i = y_i - f(xi, \beta) = y_i - f(xi, \beta^k) + \left(f(xi, \beta^k) - f(xi, \beta)\right) \approx \mathbf{\Delta} y_i - \sum_{s=1}^{n} J_{is} \mathbf{\Delta} \beta_s$

Αντικαθιστώντας τις εκφράσεις στις παραγώγους γίνονται :

-2$\sum_{i=1}^{m} J_{ij}$ ($\mathbf{\Delta} y_i - \sum_{s=1}^{n} J_{is} \mathbf{\Delta} \beta_s$) = 0 που με αναδιοργάνωση γίνονται φυσικές ισότητες:

$\sum_{i=1}^{m} \sum_{s=1}^{n} J_{ij} J_{is} \mathbf{\Delta} \beta_s = \sum_{i=1}^{m} J_{ij} \mathbf{\Delta} y_i$.

Σε μορφή πινάκων γράφονται : $(J^T J)\mathbf{\Delta}\beta = J^T \mathbf{\Delta} y$

3.4 Διαγνωστικά

Μετά την κατασκευή ενός μοντέλου regression, είναι σημαντικό να προσδιοριστεί πόσο καλό είναι το ταίριασμα στο μοντέλο και η στατιστική σημασία των παραμέτρων που εκτιμήθηκαν. Έλεγχοι που χρησιμοποιούνται συχνά για τον έλεγχο της ποιότητας του ταιριάσματος είναι η μέθοδος R-squared, ανάλυση του μοτίβου των υπολοίπων και δοκιμές υποθέσεως. Η στατιστική σημασία μπορεί να ελεγχθεί με ένα F-test του συνολικού ταιριάσματος, ακολουθούμενο από t-tests των ατομικών παραμέτρων.

Το R-squared , γνωστό και ως συντελεστής προσδιορισμού , είναι το ποσοστό της διασποράς της εξαρτημένης μεταβλητής που προβλέπεται από τις ανεξάρτητες μεταβλητές. Στη στατιστική χρησιμοποιείται στα στατιστικά μοντέλα με σκοπό την πρόβλεψη μελλοντικών εξόδων ή την δοκιμή υποθέσεων και προσφέρει μία μετρική για το πόσο καλά οι έξοδοι αναπαρίστανται από το μοντέλο, βασισμένο στο ποσοστό της συνολικής διασποράς των εξόδων [31]. Για την εξήγηση του μεγέθους έστω ότι υπάρχει ένα σύνολο n εξόδων $y_1, y_2, \ldots, y_n$ και η καθεμία αντιστοιχεί σε μία προβλεπόμενη τιμή $f_1, f_2, \ldots, f_n$. Οι διαφορές τους είναι $e_i = y_i - f_i$. Χρησιμοποιώντας το σύμβολο $\bar{y}$ για τον μέσο των παρατηρούμενων δεδομένων: $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, η μεταβλητότητα των δεδομένων μπορεί να υπολογιστεί χρησιμοποιώντας τρεις μεθόδους αθροίσματος τετραγώνων:

Το συνολικό άθροισμα τετραγώνων $SS_{tot} = \sum_i (y_i - \bar{y})^2$, το regression άθροισμα τετραγώνων , γνωστό και ως εκφρασμένο άθροισμα τετραγώνων, $SS_{reg} = \sum_i (f_i - \bar{y})^2$ και το άθροισμα τετραγώνων υπολοίπων $SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$.

Ο γενικός ορισμός των συντελεστή προσδιορισμού είναι : $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

Το R^2 είναι μια μετρική που δίνει πληροφορίες για το πόσο καλό είναι το ταίριασμα στο μοντέλο. Η μέγιστη τιμή που μπορεί να πάρει είναι 1 που δείχνει ότι οι προβλέψεις ταιριάζουν τέλεια στα δεδομένα. Ωστόσο μπορεί να πάρει και τιμές μικρότερες του 0 , όταν επιλέγεται το λάθος μοντέλο ή έχουν εφαρμοστεί άκυροι περιορισμοί. Το R^2 αυξάνεται όταν προστίθενται μεταβλητές για την πρόβλεψη, οπότε πρέπει να υπάρχει προσοχή στη χρήση του για το πότε οι μεταβλητές προσφέρουν χρήσιμη πληροφορία.

Εκτός από το R^2 , σε επόμενη ενότητα θα γίνει αναφορά και κάποιες άλλες μετρικές.

Το F-test είναι οποιοδήποτε δοκιμαστικό στατιστικό ακολουθεί μία F-κατανομή [32] σύμφωνα με τη μηδενική υπόθεση [33]. Χρησιμοποιείται κυρίως για να συγκρίνει στατιστικά μοντέλα που έχουν γίνει ταίριασμα σε ένα σύνολο δεδομένων, προκειμένου να προσδιορίσει πιο μοντέλο ταιριάζει καλύτερα στον πληθυσμό που μαζεύτηκαν τα δεδομένα. Η πιο γνωστή χρήση του είναι η ανάλυση της διασποράς, που ερμηνεύει αν οι προσδοκώμενες τιμές μιας μεταβλητής μέσα σε διάφορες ομάδες διαφέρουν μεταξύ τους. Ο τύπος για τη F-test στατιστική μετρική είναι : $F = \frac{explained\ variance}{unexplained\ variance}$.

Το explained variance , δηλαδή η ποικιλομορφία μεταξύ των ομάδων συμβολίζεται με: $\sum_{i=1}^{K} n_i(\bar{Y}_i - \bar{Y})^2/(K-1)$ όπου το $\bar{Y}_i$ είναι ο δειγματικός μέσος στην i ομάδα, n_i είναι ο αριθμός των παρατηρήσεων στην i ομάδα, Y ο συνολικός μέσος των δεδομένων και K οι ομάδες.
Το unexplained variance , δηλαδή η ποικιλομορφία μέσα στην ομάδα συμβολίζεται με: $\sum_{i=1}^{K}\sum_{j=1}^{n_i}\left(Y_{ij} - \bar{Y}_i\right)^2/(N-K)$, όπου Y_{ij} είναι η j παρατήρηση στην i από τις K ομάδες και N το συνολικό μέγεθος δείγματος.

Το t-test είναι μία στατιστική υποθετική δοκιμή , που το δοκιμαστικό στατιστικό ακολουθεί μία Student t-κατανομή [34] κάτω από τη μηδενική υπόθεση. Η Student-t κατανομή είναι οποιοδήποτε μέλος από μία οικογένεια συνεχών πιθανοτικών κατανομών που προκύπτει όταν εκτιμάται ο μέσος ενός κανονικά κατανεμημένου πληθυσμού όταν το δείγμα είναι μικρό και η απόκλιση του πληθυσμού άγνωστη. Το t-test συνήθως εφαρμόζεται όταν το δοκιμαστικό στατιστικό θα ακολουθούσε κανονική κατανομή αν η τιμή του όρου κλιμάκωσής του ήταν γνωστή. Όταν δεν είναι γνωστή αντικαθίσταται από μία εκτίμηση που προκύπτει από τα δεδομένα. Οι πιο συνηθισμένες χρήσης του t-test είναι:
Μία μονού δείγματος δοκιμή τοποθεσίας του αν ο μέσος του πληθυσμού έχει τιμή ορισμένη στη μηδενική υπόθεση.
Μία διπλού δείγματος δοκιμή τοποθεσίας της μηδενικής υπόθεσης έτσι ώστε οι μέσοι των δύο πληθυσμών να είναι ίσοι, με την προϋπόθεση ότι θεωρείται πως οι διασπορές των δύο πληθυσμών είναι επίσης ίσες.
Για την περίπτωση του μονού δείγματος που ελέγχεται αν ο μέσος του πληθυσμού είναι μ0 χρησιμοποιείται το στατιστικό t = $\frac{\bar{x}-\mu_0}{s/\sqrt{n}}$, όπου $\bar{x}$ ο δειγματικός μέσος , s η δειγματική απόκλιση και n το μέγεθος δείγματος.
Για την περίπτωση του διπλού δείγματος που ελέγχεται αν οι μέσοι των πληθυσμών είναι ίσοι χρησιμοποιείται το στατιστικό t = $\frac{\bar{X}_1-\bar{X}_2}{s_p\sqrt{\frac{2}{n}}}$, όπου sp = $\sqrt{\frac{{s_{X_1}}^2+{s_{X_2}}^2}{2}}$. Το s_p είναι η απόκλιση για n = n_1 = n_2 και τα ${s_{X_1}}^2$ και ${s_{X_2}}^2$ είναι οι αντικειμενικές εκτιμήσεις για τις διασπορές των δειγμάτων.

3.5 Regression στην Python

Για τους σκοπούς της διπλωματικής η χρησιμοποίηση των μοντέλων regression έχει γίνει μέσω Python scripts. Σε αυτήν την ενότητα εξετάζονται μερικές μέθοδοι regression και μετρικές που θα χρησιμοποιηθούν στα μοντέλα. Η βιβλιοθήκη από την οποία προέρχονται είναι η scikit-learn [35].

Η μονάδα sklearn.linear_model [36] υλοποιεί μία ποικιλία από γραμμικά μοντέλα. Από αυτά που είναι για regression έχουν εξεταστεί:

- linear_model.LinearRegression: Το κλασικό regression με χρήση ελαχίστων τετραγώνων
- linear_model.Ridge: Γραμμικό regression ελαχίστων τετραγώνων με l^2 κανονικοποίηση, δηλαδή ποινή στο μέγεθος των συντελεστών. Οι συντελεστές ελαχιστοποιούν τη συνάρτηση $\min_m (Xw-y)_2^2 + \alpha(w)_2^2$, όπου α δίνεται ως παράμετρος στο regression και $(w)_2^2$ είναι η l^2 κανονικοποίηση του συντελεστή.

- linear_model.RidgeCV: Ridge regression που χρησιμοποιεί cross validation. Το cross validation είναι μία μέθοδος που δεδομένης της παραμέτρου cv χωρίζει το σύνολο δεδομένων σε τόσα επιμέρους σύνολα όσα η τιμή της cv και κάνει προβλέψεις χρησιμοποιώντας το ένα σύνολο ως σύνολο δοκιμής και τα άλλα για να εκπαιδεύσει το μοντέλο.
- linear_model.SGDRegressor: Μοντέλο που γίνεται ταίριασμα με την ελαχιστοποίηση μίας κανονικοποιημένης εμπειρικής συνάρτησης απώλειας με SGD. SGD σημαίνει στοχαστική κάθοδος παραγώγου και περιλαμβάνει την εκτίμηση της παραγώγου της συνάρτησης για κάθε δείγμα χωριστά και ανανέωση του μοντέλου με μειούμενο βαθμό μάθησης. Η κανονικοποίηση γίνεται με την προσθήκη ποινών στην συνάρτηση που μικραίνει τις παραμέτρους του μοντέλου. Γενικά , δουλεύει καλά όταν υπάρχουν πολλά δείγματα (πάνω από 10000).
- linear_model.Lasso: Γραμμικό regression εκπαιδευμένο με κανονικοποίηση l^1. Χρησιμοποιείται καλά σε κάποιες περιστάσεις καθώς προτιμά λύσεις που υπάρχουν λιγότεροι μη μηδενικοί συντελεστές, και άρα μειώνει τον αριθμό χαρακτηριστικών από τα οποία εξαρτάται η λύση. Ελαχιστοποιεί την παρακάτω αντικειμενική συνάρτηση: $\min_w \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha\|w\|_1$, όπου α δίνεται ως παράμετρος στη μέθοδο και $\|w\|_1$ η κανονικοποίηση l^1. Υπάρχει μοντέλο LassoCV που χρησιμοποιεί cross validation.
- linear_model.ElasticNet: Γραμμικό regression με κανονικοποίηση l^1και l^2. Ελαχιστοποιεί την αντικειμενική συνάρτηση $\frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha\rho\|w\|_1 + \frac{\alpha(1-\rho)}{2}\|w\|_2^2$, όπου τα μεγέθη α και ρ δίνονται ως παράμετροι στην μέθοδο. Είναι χρήσιμο όταν πολλαπλά χαρακτηριστικά έχουν συσχετίσεις μεταξύ τους. Υπάρχει μοντέλο ElasticNetCV, που χρησιμοποιεί cross validation.
- linear_model.Lars; Μοντέλο regression ελάχιστης γωνίας. Σε κάθε βήμα, βρίσκει το χαρακτηριστικό που συσχετίζεται πιο πολύ με το στόχο. Αν δύο χαρακτηριστικά έχουν ίδια συσχέτιση , ακολουθεί μία ισογώνια κατεύθυνση μεταξύ των χαρακτηριστικών. Έχει τα πλεονεκτήματα ότι είναι γρήγορο, σταθερό , παράγει ένα μονοπάτι γραμμικής λύσης και μπορεί να μετατραπεί για να παράγει λύσεις σε άλλους εκτιμητές. Ωστόσο, είναι ευαίσθητο στο θόρυβο. Υπάρχει μοντέλο LarsCV που χρησιμοποιεί cross validation.
- linear_model.LassoLars: Μοντέλο Lasso που του εφαρμόζεται ο αλγόριθμος Lars. Αυτό σημαίνει ότι χρησιμοποιείται η κανονικοποίηση που εφαρμόζεται και στο Lasso και ακολουθούνται τα βήματα του Lars. Υπάρχει μοντέλο LassoLarsCV που χρησιμοποιεί cross validation και LassoLarsIC που χρησιμοποιεί δύο από τα κριτήρια AIC και BIC.
- linear_model.OrthogonalMatchingPursuit: Μοντέλο Orthogonal Matching Pursuit. Εφαρμόζει τον αλγόριθμο OMP για να προσεγγίσει το ταίριασμα ενός γραμμικού μοντέλου με περιορισμούς που επιβάλλονται από τον αριθμό των μη μηδενικών συντελεστών. Συγκεκριμένα: $\operatorname{argmin}_\gamma \|y - X\gamma\|_2^2$ που υπόκειται σε $\|\gamma\|_0 \leq n_{nonzerp\_coefs}$. Εναλλακτικά, το OMP μπορεί να στοχεύσει ένα συγκεκριμένο σφάλμα κάτι που εκφράζεται ως: $\operatorname{argmin}_\gamma \|\gamma\|_0$ που υπόκειται σε $\|y - X\gamma\|_2^2 \leq tol$. Το OMP βασίζεται σε άπληστο αλγόριθμο που περιλαμβάνει σε κάθε βήμα το άτομο που είναι πιο συσχετισμένο με το παρόν υπόλοιπο. Σε κάθε επανάληψη το υπόλοιπο ξαναϋπολογίζεται χρησιμοποιώντας μία ορθογωνική προβολή στο χώρο των

προηγούμενων επιλεγμένων στοιχείων. Υπάρχει μοντέλο OrthogonalMatchingPursuitCV που χρησιμοποιεί cross validation.

- Bayesian Regression: Οι τεχνικές Bayesian Regression χρησιμοποιούνται για να συμπεριλάβουν κανονικοποιημένες παραμέτρους στη διαδικασία εκτίμησης. Αυτό πραγματοποιείται με την εισαγωγή αφηρημένων πιθανοτήτων πάνω στις υπερπαραμέτρους του μοντέλου. Για την απόκτηση πλήρους πιθανοτικού μοντέλου , η έξοδος y θεωρείται ως Gaussian κατανεμημένη γύρω από το Xw : $p(y|X,w,\alpha) = \mathcal{N}(y|X,w,\alpha)$, όπου α ορίζεται ως τυχαία μεταβλητή που εκτιμάται από τα δεδομένα. Πλεονεκτήματα περιλαμβάνουν την προσαρμοστικότητα στα δεδομένα και τη χρήση παραμέτρων κανονικοποίησης. Μειονεκτήματα περιλαμβάνουν την κατανάλωση χρόνου. Υπάρχουν 2 μοντέλα στην python που χρησιμοποιούν Bayesian Regression:
  - linear_model.BayesianRidge: Χρησιμοποιεί την παραπάνω τεχνική και η πιθανότητα για τον συντελεστή w δίνεται από σφαιρική γκαουσιανή: $p(w|\lambda) = \mathcal{N}(w|0,\lambda^{-1}\mathbf{I}_p)$, όπου οι πιθανότητες πάνω σε α και λ επιλέγεται να είναι κατανομές γάμμα. Οι παράμετροι w,α και λ εκτιμώνται κατά το ταίριασμα του μοντέλου.
  - linear_model.ARDRegression: Παρόμοιο με το προηγούμενο, αλλά οδηγεί σε πιο αραιούς συντελεστές. Αντί για σφαιρική , η κατανομή γύρω από το w θεωρείται να είναι παράλληλη-αξόνων, ελλειπτική γκαουσιανή κατανομή. Η πιθανότητα του w είναι: $p(w|\lambda) = \mathcal{N}(w|0,A^{-1})$ με diag(A) = λ = $\{\lambda_1,\ldots..,\lambda_p\}$, όπου κάθε συντεταγμένη w_i έχει τη δικιά της απόκλιση λ_i .
- linear_model.RANSACRegressor: Το RANSAC είναι ένας επαναληπτικός αλγόριθμος για την εύρωστη εκτίμηση παραμέτρων από ένα σύνολο inliers από ολόκληρο το σύνολο δεδομένων. Τα inliers αντιπροσωπεύουν εσφαλμένες τιμές που βρίσκονται στο εσωτερικό της κατανομής των σωστών μετρήσεων, ενώ τα outliers τιμές στα όρια της κατανομής. Το αποτέλεσμα προκύπτει με πιθανότητα που εξαρτάται από τον αριθμό των επαναλήψεων. Πιο αναλυτικά τα βήματα του αλγορίθμου περιλαμβάνουν:
  1. Επιλογή min_samples τυχαίων δειγμάτων από τα αρχικά δεδομένα και έλεγχος αν το σύνολο είναι βάσιμο.
  2. Επιλογή μοντέλου από τυχαίου υποσύνολο και έλεγχος αν το εκτιμώμενο μοντέλο είναι έγκυρο.
  3. Καθορισμός των δεδομένων σε inliers και outliers με υπολογισμό των υπολοίπων από το εκτιμώμενο μοντέλο. Όλα τα δείγματα με απόλυτα υπόλοιπα μικρότερα του residual_threshold θεωρούνται inliers.
  4. Ορισμός του ταιριασμένου μοντέλου ως το καλύτερο αν ο αριθμός δειγμάτων inlier είναι μέγιστος.
- linear_model.TheilSenRegressor: Αυτός ο εκτιμητής χρησιμοποιεί μία γενίκευση του μέσου σε πολλαπλές διαστάσεις και είναι εύρωστος σε πολυπαραγοντικά outliers. Η μέθοδος είναι συγκρίσιμη με τη μέθοδο ελαχίστων τετραγώνων, αλλά είναι μη παραμετρική , άρα δεν κάνει υποθέσεις για την κατανομή δεδομένων. Πιο αναλυτικά ο αλγόριθμος υπολογίζει λύσεις σε υποσύνολα με μέγεθος n_subsamples των δειγμάτων. Η τιμή του n_subsamples μεταξύ χαρακτηριστικών και δειγμάτων δημιουργεί συμβιβασμό ανάμεσα σε αποδοτικότητα και δύναμη. Ο αριθμός των λύσεων μπορεί να περιοριστεί από το max_subpopulation, και αν φτάσει το όριο , τα δείγματα επιλέγονται τυχαία. Ο χωρικός μέσος υπολογίζεται από όλες τις ελαχίστων τετραγώνων λύσεις.
- linear_model.HuberRegressor: Γραμμικό regression , που είναι εύρωστο στα outliers. Εφαρμόζει μία συνάρτηση απώλειας σε δείγματα που ορίζονται ως outliers, και δεν τα αγνοεί σε αντίθεση με τις δύο προηγούμενες μεθόδους. Η συνάρτηση απώλειας

που ελαχιστοποιείται δίνεται από: $\min_{w,\sigma} \sum_{i=1}^{n} \left(\sigma + H_\varepsilon\left(\frac{X_i w - y_i}{\sigma}\right)\sigma\right) + \alpha(w)_2^2$, όπου $H_\varepsilon = \begin{cases} z^2 & , if\ |z| < \varepsilon \\ 2\varepsilon|z| - \varepsilon^2, & otherwise \end{cases}$

- linear_model.PassiveAggressiveRegressor: Οι αλγόριθμοι passive aggressive είναι οικογένεια αλγορίθμων για μεγάλης κλίμακας μάθηση. Δεν απαιτούν ρυθμό μάθησης , αλλά περιλαμβάνουν μία παράμετρο κανονικοποίησης C.

Η μονάδα sklearn.neighbors [37] παρέχει λειτουργικότητα για μεθόδους μάθησης που βασίζονται σε γείτονες. Η ιδέα πίσω από τους κοντινότερους γείτονες είναι η εύρεση ενός ορισμένου αριθμού από δείγματα του training συνόλου κοντινότερα σε απόσταση στο καινούριο σημείο και την πρόβλεψη της εξόδου-ετικέτας από αυτά. Ο αριθμός των δειγμάτων μπορεί να εξαρτάται από τον χρήστη ή από την τοπική πυκνότητα των σημείων. Η απόσταση μπορεί να είναι οποιαδήποτε μετρική μέτρησης, συνήθως όμως είναι η ευκλείδεια απόσταση.

Το Neighbors regression χρησιμοποιείται σε περιπτώσεις όπου οι έξοδοι-ετικέτες για τα δεδομένα είναι συνεχείς αντί για διακριτές μεταβλητές. Η ετικέτα σε ένα σημείο υπολογίζεται βασισμένη στον μέσο των ετικετών των κοντινότερων γειτόνων. Ο KNeighborsRegressor εφαρμόζει μάθηση βασισμένη στους k κοντινότερους γείτονες κάθε σημείου, όπου το k ορίζεται από το χρήστη μέσω της παραμέτρου n_neighbors. Στη βασική του μορφή χρησιμοποιεί τα ίδια βάρη στους γείτονες, δηλαδή κάθε σημείο στην τοπική γειτονιά συνεισφέρει το ίδιο στον καθορισμό του ζητούμενου σημείου. Ωστόσο μπορεί να έχει σημασία τα κοντινότερα σημεία να συνεισφέρουν περισσότερο, οπότε χρησιμοποιείται η παράμετρος weights. Η τιμή weights = 'uniform' βάζει παντού τα ίδια βάρη, ενώ η weights = 'distance' βάζει μεγαλύτερα βάρη σε κοντινότερα σημεία.

Η μονάδα sklearn.neural_network [38] περιλαμβάνει μοντέλα που βασίζονται σε νευρωνικά δίκτυα. Από αυτά αυτό που θα εξεταστεί είναι το Multi-layer Perceptron (MLP) που είναι ένας αλγόριθμος που μαθαίνει μία συνάρτηση f(·): $R^m \rightarrow R^o$ με εκπαίδευση σε σύνολο δεδομένων, όπου m ο αριθμός διαστάσεων για είσοδο και ο για έξοδο. Η προσέγγιση μεταξύ της εξόδου και εισόδου γίνεται μέσω κρυφών μη γραμμικών επιπέδων. Τα πλεονεκτήματα του MLP είναι ότι υποστηρίζει μη γραμμικά μοντέλα και σε πραγματικό χρόνο, ενώ τα μειονεκτήματα είναι ότι υστερεί σε ακρίβεια επιβεβαίωσης και δεν υποστηρίζει κάποιες υπερπαραμέτρους. Επίσης υποστηρίζει regression μέσω του neural_network.MLPRegressor. Οι αλγόριθμοι που χρησιμοποιεί είναι οι SGD,Adam και LBFGS. Το SGD ανανεώνει παραμέτρους με τη χρήση της συνάρτησης απώλειας σεβόμενο μία παράμετρο προσαρμογής: $w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$, όπου η ο βαθμός μάθησης που ελέγχει το μέγεθος βήματος και Loss η συνάρτηση απώλειας που χρησιμοποιείται στο νευρωνικό.

Το Adam είναι σαν το SGD , αλλά μπορεί αυτόματα να προσαρμόζει την ποσότητα που ανανεώνει τις παραμέτρους βασισμένο σε εκτιμήσεις των χαμηλότερης τάξης στιγμών. Το LBFGS είναι ένας λύτης που προσεγγίζει τον Hessian πίνακα που αναπαριστά την δεύτερης τάξης μερική παράγωγο μίας συνάρτησης. Προσεγγίζει και τον αντίστροφο του πίνακα για να πραγματοποιεί ανανεώσεις παραμέτρων.

Η μονάδα sklearn.tree [39] περιλαμβάνει μοντέλα που βασίζονται σε δέντρα απόφασης. Τα δέντρα απόφασης είναι μία μη παραμετρική μέθοδος μάθησης που προβλέπει την τιμή της εξόδου μαθαίνοντας κανόνες απόφασης από τα χαρακτηριστικά των δεδομένων. Όσο πιο βαθύ είναι το δέντρο , τόσο πιο περίπλοκοι είναι οι κανόνες απόφασης. Τα πλεονεκτήματα των δέντρων είναι η ευκολία της κατανόησής τους, η ταχύτητα

προετοιμασίας τους, η υποστήριξη διαφορετικών τύπων δεδομένων και γενικά καλή απόδοση. Τα μειονεκτήματα τους είναι ότι μπορεί να προκαλέσουν υπερταίριασμα, είναι ασταθή, είναι δύσκολο να βελτιστοποιηθούν και αν κυριαρχούν κάποιες κλάσεις τα δέντρα δεν είναι αντικειμενικά. Για τη λύση των προβλημάτων πρέπει τα δεδομένα να έχουν περάσει από μία προηγούμενη διαδικασία. Η μέθοδος tree.DecisionTreeRegressor παράγει την έξοδο χρησιμοποιώντας ένα δέντρο απόφασης. Η μέθοδος tree.ExtraTreeRegressor χρησιμοποιεί εντελώς τυχαία δέντρα. Κατά το χτίσιμο , όταν αναζητείται η καλύτερη τομή για διαχωρισμό των δειγμάτων σε δύο ομάδες , γίνονται τυχαίες τομές για κάθε από τα max_features τυχαία επιλεγμένα χαρακτηριστικά και επιλέγεται η καλύτερη τομή. Όταν max_features = 1, το δέντρο είναι εντελώς τυχαίο.

Η μονάδα sklearn.svm [40] περιλαμβάνει αλγορίθμους για μηχανές υποστήριξης διανύσματος. Τα πλεονεκτήματα των μηχανών είναι η αποτελεσματικότητα σε πολλαπλές διαστάσεις, η καλή αξιοποίηση μνήμης και η ευελιξία. Τα μειονεκτήματα είναι ότι λίγα δείγματα οδηγούν σε υπερταίριασμα και η μέθοδος πρόβλεψης βασίζεται σε cross validation και άρα είναι ακριβή. Για το regression το μοντέλο που παράγεται εξαρτάται μόνο από ένα υποσύνολο των train δεδομένων , επειδή το κόστος της συνάρτησης για το χτίσιμο αγνοεί train δεδομένα κοντά στην πρόβλεψη του μοντέλου. Οι μέθοδοι που υποστηρίζει είναι οι ακόλουθες:

- svm.SVR: Epsilon-Support Vector regression που έχει ως ελεύθερες παραμέτρους τις C και epsilon. Η εφαρμογή γίνεται με τη βιβλιοθήκη libsvm και η πολυπλοκότητα χρόνου είναι δευτέρου βαθμού σε σχέση με τον αριθμό δειγμάτων και είναι δύσκολο να κλιμακωθεί με μεγάλα σύνολα δεδομένων.
- svm.NuSVR: Nu Support Vector regression που χρησιμοποιεί την παράμετρο nu για να ελέγξει τον αριθμό των διανυσμάτων υποστήριξης. Το nu αντικαθιστά την παράμετρο epsilon στο epsilon-SVR και η εφαρμογή γίνεται με βιβλιοθήκη libsvm.
- svm.LinearSVR: Γραμμικό Support Vector regression που είναι παρόμοια με το SVR με παράμετρο kernel = ‘linear’, αλλά εφαρμόζεται με βιβλιοθήκη liblinear αντί για libsvm. Αυτό σημαίνει ότι έχει περισσότερη ευελιξία στην επιλογή ποινών και συναρτήσεων απώλειας και κλιμακώνεται καλύτερα σε μεγάλους αριθμούς δειγμάτων.

Η μονάδα sklearn.ensemble [41] περιλαμβάνει ensemble μεθόδους. Στόχος τους είναι ο συνδυασμός των προβλέψεων κάποιων βασικών εκτιμητών χτισμένους με έναν αλγόριθμο μάθησης με σκοπό τη βελτίωση της δύναμης σε σχέση με έναν μόνο εκτιμητή. Υπάρχουν δύο κατηγορίες μεθόδων:

- Μέσου όρου, όπου χτίζονται διάφοροι εκτιμητές και βρίσκεται ο μέσος όρος των προβλέψεων.
- Ενδυνάμωσης, όπου οι βασικοί εκτιμητές χτίζονται σειριακά και ο καθένας προσπαθεί να μειώσει την κλίση του τελικού εκτιμητή.

Για το regression παρέχονται μέθοδοι που ακολουθούν την παραπάνω λογική:

- ensemble.BaggingRegressor: Regressor που χτίζει διάφορα παραδείγματα ενός εκτιμητή σε τυχαία υποσύνολα του αρχικού συνόλου και συγκεντρώνει τις προβλέψεις σε μία τελική. Με αυτό το τρόπο μειώνεται η διασπορά του βασικού εκτιμητή με την εισαγωγή τυχαιότητας. Το Bagging είναι ένας απλός τρόπος να βελτιώσει ένα μοντέλο και να μειώσει το υπερταίριασμα , χωρίς να προσαρμόζει τον βασικό αλγόριθμο.
- ensemble.RandomForestRegression: Κατασκευάζει τυχαία δάση, όπου κάθε δέντρο χτίζεται από ένα δείγμα σχεδιασμένο με αντικατάσταση από το σύνολο εκπαίδευσης. Κατά το διαχωρισμό των κόμβων, η καλύτερη τομή βρίσκεται από όλα τα

χαρακτηριστικά εισόδου ή τυχαίο υποσύνολο μεγέθους max_features. Αυτό μειώνει τη διασπορά του εκτιμητή.

- ensemble.ExtraTreesRegressor: Όπως και στο RandomForestRegression χρησιμοποιούνται ένα τυχαίο υποσύνολο χαρακτηριστικών, αλλά αντί για αναζήτηση των πλέον διακριτών κατωφλίων, κατώφλια σχεδιάζονται τυχαία για κάθε χαρακτηριστικό και το καλύτερο επιλέγεται για τον κανόνα τομής. Αυτό μειώνει τη διασπορά λίγο περισσότερο, αλλά με αντίτιμο λίγο μεγαλύτερη αύξηση στην κλίση.
- ensemble.VotingRegressor: Συνδυάζει ριζικά διαφορετικά regressors μηχανικής μάθησης και επιστρέφει την μέση προβλεπόμενη τιμή. Με αυτό τον τρόπο τα μοντέλα μπορούν να καλύψουν τις αδυναμίες τους.

Η μονάδα sklearn.metrics [42] περιλαμβάνει αποτελέσματα συναρτήσεων, υπολογισμούς αποστάσεων και μετρικές απόδοσης. Για το regression κάποιες μετρικές είναι:

- metrics.explained_variance_score: Συνάρτηση που βρίσκει το explained variance score. Αν $\hat{y}$ είναι η εκτιμώμενη έξοδος, y η πραγματική και Var η διασπορά τότε το explained variance είναι: $1 - \frac{Var\{y-\hat{y}\}}{Var\{y\}}$. Η καλύτερη τιμή είναι 1.
- metrics.max_error: Συνάρτηση που υπολογίζει το μέσο σφάλμα υπολοίπων, δηλαδή το χειρότερης περίπτωσης σφάλμα μεταξύ της προβλεπόμενης τιμής και της πραγματικής. Η καλύτερη τιμή είναι 0. Πιο συγκεκριμένα, Max Error$(y, \hat{y})$ = max$(|y_i - \hat{y_i}|)$.
- metrics.mean_absolute_error: Υπολογίζει το μέσο απόλυτο σφάλμα, μία μετρική ρίσκου που αναλογεί στην προσδοκώμενη τιμή της απώλειας απόλυτου σφάλματος ή της l^1-norm απώλειας. Αν $\hat{y}_i$ η προβλεπόμενη τιμή του δείγματος i και y_i η πραγματική τιμή , τότε το μέσο απόλυτο σφάλμα πάνω σε $n_{samples}$ είναι: MAE$(y, \hat{y})$ $= \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}-1} |\hat{y}_i - y_i|$.
- metrics.mean_squared_error: Υπολογίζει το μέσο τετραγωνικό σφάλμα, μία μετρική ρίσκου που αναλογεί στην προσδοκώμενη τιμή του τετραγωνικού σφάλματος ή απώλειας. Αν $\hat{y}_i$ η προβλεπόμενη τιμή του δείγματος i και y_i η πραγματική τιμή, τότε το μέσο τετραγωνικό σφάλμα πάνω σε $n_{samples}$ είναι: MSE$(y, \hat{y})$ $= \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}-1} (\hat{y}_i - y_i)^2$.
- metrics.mean_squared_log_error: Υπολογίζει μία μετρική ρίσκου που αναλογεί στην προσδοκώμενη τιμή του τετραγωνικού λογαριθμικού σφάλματος ή απώλειας. Αν $\hat{y}_i$ η προβλεπόμενη τιμή του δείγματος i και y_i η πραγματική τιμή, τότε το μέσο τετραγωνικό λογαριθμικό σφάλμα πάνω σε $n_{samples}$ είναι: MSLE$(y, \hat{y})$ $= \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2$. Χρησιμοποιείται καλύτερα όταν οι έξοδοι έχουν εκθετική αύξηση και τα σφάλματα μειωμένης πρόβλεψης είναι μεγαλύτερα από τα αυξημένης πρόβλεψης.
- metrics.median_absolute_error: Είναι εύρωστη στα outliers και η απώλεια υπολογίζεται παίρνοντας τον διάμεσο όλων των απόλυτων διαφορών ανάμεσα στο στόχο και την πρόβλεψη. Αν $\hat{y}_i$ η προβλεπόμενη τιμή του δείγματος i και y_i η πραγματική τιμή, τότε το μέσο τετραγωνικό λογαριθμικό σφάλμα πάνω σε $n_{samples}$ είναι: MedAE$(y, \hat{y})$ = median$((y_1 - \hat{y}_1), \dots, (y_n - \hat{y}_n))$.
- metrics.r2_score: Υπολογίζει το r^2 score. Έχει ήδη γίνει μία μελέτη πάνω στον τύπο και τις ιδιότητές του. Να σημειωθεί η αδυναμία του r^2 score να μην μειώνεται όταν προστίθενται χαρακτηριστικά. Για αυτό χρησιμοποιείται το adjusted r^2 score, το

οποίο δίνεται από τον τύπο $R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-k-1}$, όπου το n είναι ο αριθμός των δεδομένων και k ο αριθμός των ανεξάρτητων χαρακτηριστικών που χρησιμοποιούνται για την πρόβλεψη.

Κάποιες από τις παραπάνω μεθόδους και μετρικές θα χρησιμοποιηθούν και στις μετρήσεις της διπλωματικής.

Κεφάλαιο 5

Προτεινόμενο πλαίσιο

1.Εισαγωγή

Οι μετρήσεις της διπλωματικής έχουν γίνει με σκοπό να προσφέρουν τη δυνατότητα να πραγματοποιούνται προβλέψεις για τον χρόνο εκτέλεσης και την κατανάλωση ενέργειας ενός κομματιού κώδικα σε κάποια πλατφόρμα. Τα προγράμματα πάνω στα οποία γίνονται προβλέψεις ανήκουν σε δύο benchmarks ,το Polybench [43] και το Rodinia-3.1 [44] , τα οποία μπορούν να βρεθούν και στο github. Τα κομμάτια κώδικα που εξετάζονται αποτελούν σώματα επαναλήψεων, καθώς σε πραγματικά προγράμματα οι επαναλήψεις είναι που καταναλώνουν τον περισσότερο χρόνο. Τα προγράμματα του Polybench έχουν μία πιο απλή μορφή και τα κομμάτια κώδικα που εξετάζονται αποτελούν κυρίως σώματα for επαναλήψεων , απλών ή εμφωλευμένων , που εκτελούν αριθμητικές πράξεις. Στα προγράμματα του Rodinia εξετάζονται κομμάτια κώδικα που αποτελούν και διαφορετικούς τύπους επαναλήψεων , όπως while και do-while. Ο κώδικας στο εσωτερικό των επαναλήψεων είναι πιο περίπλοκος, καθώς περιλαμβάνει πράξεις μεταξύ δομών και γίνονται κλήσεις συναρτήσεων, που εισάγουν τις δικές τους δομές. Γενικά, τα προγράμματα του Rodinia είναι πιο αντιπροσωπευτικά των προγραμμάτων στα οποία γενικά πραγματοποιείται ανάλυση , και διαφέρουν αισθητά μεταξύ τους. Τα μεγέθη που προβλέπονται είναι ο χρόνος εκτέλεσης μετρημένος σε msec και η ενέργεια μετρημένη σε Joule σε επίπεδο CPU. Τα χαρακτηριστικά που χρησιμοποιούνται για την πρόβλεψη είναι ο αριθμός των εντολών και ένα σύνολο χαρακτηριστικών που προκύπτουν από στατική ανάλυση του κώδικα και θα αναλυθούν παρακάτω.

2. Κώδικες υπολογισμών

2.1 Clang parsing

Πριν γίνουν οι μετρήσεις . πρέπει να βρεθούν τα κομμάτια του κώδικα που θα εξεταστούν. Για αυτό χρησιμοποιείται ένα script σε python, το οποίο βρίσκει το clang AST του προγράμματος που του δίνεται ως είσοδος. Για την πραγματοποίηση των ελέγχων χρησιμοποιείται η βιβλιοθήκη clang.cindex. Από το δέντρο που προκύπτει τα statements που έχουν ενδιαφέρον είναι οι δηλώσεις συναρτήσεων, οι for και while statements, και οι if statements. Ανάλογα με την τιμή μίας παραμέτρου γίνεται έλεγχος για να βρεθούν συγκεκριμένα statements και σε ποιες γραμμές εντοπίζονται. Αυτή η διαδικασία πραγματοποιείται για όλα τους κόμβους του AST . Ένα παράδειγμα εξόδου είναι το ακόλουθο:

```
Source file: openmp/srad/srad_v2/srad.cpp Type/Name: random_matrix(float *, int, int), First line: 226, Last line: 242
Source file: openmp/srad/srad_v2/srad.cpp Type/Name: CursorKind.IF_STMT, First line: 48, Last line: 66
```

Δίνεται το όνομα του αρχείου που στην συγκεκριμένη περίπτωση είναι srad.cpp, το όνομα της συνάρτησης είναι random_matrix, δέχεται τρεις παραμέτρους όπου η μία είναι δείκτης float και οι άλλες δύο είναι int, η πρώτη γραμμή όπου γίνεται η δήλωσή της είναι η 226 και η γραμμή στην οποία τερματίζει η δήλωση είναι 242.

2.2 Αναγνώριση χαρακτηριστικών

Σε κάποια από τα κομμάτια κώδικα που εντοπίζονται με τη Clang, γίνεται χρήση του architectural code analyzer για να υπολογιστούν τα χαρακτηριστικά σε ένα block του κώδικα για μία επανάληψη. Αυτά τα γνωρίσματα θα χρησιμοποιηθούν για την πρόβλεψη, οπότε θα γίνει μία μελέτη που εξηγεί την επιλογή τους.

Στα προγράμματα από τα benchmarks τα αρχεία που τα μεταγλωττίζουν χρησιμοποιούν διαφορετικές σημαίες βελτιστοποίησης, οπότε για λόγους απλοποίησης και την πιο καλή πρόβλεψη θα μεταγλωττιστούν όλα χωρίς τη χρήση σημαίων βελτιστοποίησης. Πιο αναλυτικά, στον μεταγλωττιστή gcc οι σημαίες βελτιστοποίσης ορίζονται ως -Oi, όπου i είναι ένα συγκεκριμένο ψηφίο ή γράμμα, με -O0 την default σημαία, δηλαδή αυτή με την οποία μεταγλωττίζεται το πρόγραμμα όταν δεν τοποθετείται η επιλογή -O. Όλες οι σημαίες έχουν διαφορετικά πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα, η σημαία -O3 μειώνει τον χρόνο εκτέλεσης του προγράμματος με αντίτιμο περισσότερη χρήση μνήμης και αυξημένο χρόνο μεταγλώττισης [45].

Με τη χρήση του architectural code analyzer παρατηρείται ότι ο κώδικας assembly που προκύπτει με μεταγλώττιση -O3, είναι μικρότερος από τον default , καθώς χρησιμοποιούνται διαφορετικές εντολές assembly όχι μόνο σε λειτουργία αλλά και σε τύπο. Για αυτό το λόγο , σε όλα τα προγράμματα που έχουν αναλυθεί και χρησιμοποιηθεί σαν είσοδος στα μοντέλα γίνεται μεταγλώττιση με την ίδια σημαία. Αφού στο βασικό επίπεδο δεν υπάρχει σημαία, αυτός είναι και ο τρόπος μεταγλώττισης.

Όπως έχει αναφερθεί , το IACA δεχόμενο το object file, δίνει τις εντολές σε assembly , το throughput των θυρών και του block, κάποιους παράγοντες που επηρεάζουν το αποτέλεσμα και τον αριθμό των micro-operations. Από αυτά τα χαρακτηριστικά, χρήσιμα είναι το block throughput, καθώς προσφέρει ένα άνω κατώφλι για τους κύκλους που δεσμεύονται στις θύρες, και ο αριθμός των micro-operations, αφού είναι μία μετρική που δείχνει το μέγεθος του κώδικα. Αν και αυτά τα γνωρίσματα είναι χρήσιμα, δεν παρέχουν πληροφορίες για την μορφή του κώδικα και τις εντολές του, οπότε χρησιμοποιείται η σημαία -trace <file> στο IACA, που δίνει τον τύπο των εντολών σε κάθε επανάληψη. Επίσης, χρησιμοποιείται η επιλογή -trace-cycle-count 300 αντί για την default(150), επειδή κάποια αρχεία έχουν πολλές εντολές σε κάθε επανάληψη και καταλαμβάνουν περισσότερους από 150 κύκλους. Στο trace file που προκύπτει δίνονται οι τύποι των εντολών, δηλαδή OP, LOAD, STORE_DATA, STORE_ADDRESS.

Σε αυτό το σημείο πρέπει να τονιστεί ότι είχε γίνει μία μελέτη σε μία πλακέτα για τους χρόνους εκτέλεσης κάθε εντολής. Έχουν φτιαχτεί προγράμματα σε C που εκτελούν πολλαπλές επαναλήψεις κάποιας συγκεκριμένης εντολής, επαναλήψεις της τάξης των εκατοντάδων εκατομμυρίων, και χρήση συγκεκριμένων βιβλιοθηκών για τον υπολογισμό του χρόνου εκτέλεσής τους. Έχοντας ως γνωστό τον αριθμό των εντολών σε όλο το πρόγραμμα και τα ποσοστά που εμφανίζεται κάθε τύπος εντολής, θα ήταν δυνατό να προβλεφθεί ο χρόνος εκτέλεσης όλου του προγράμματος. Ωστόσο, η πλακέτα δεν αποτελεί απλό μικροεπεξεργαστή και δεν είναι δυνατό να πραγματοποιείται ακριβής μέτρηση κάθε εντολής λόγω του λειτουργικού συστήματος. Η προσέγγιση που πραγματοποιείται επιχειρείται να είναι cross-platform και cross-architecture, οπότε δεν έχει σημασία η ακριβής μέτρηση για κάθε εντολή σε μία πλακέτα, αλλά να ομαδοποιηθούν οι εντολές παράγοντας αντιπροσωπευτικά features της συμπεριφοράς του προγράμματος ως προς το πώς αυτή θα επηρεάζει το χρόνο και την ενέργεια.

Τα loads και stores είναι χρήσιμα στους υπολογισμούς, επειδή είναι εντολές που απαιτούν αρκετό χρόνο για την εκτέλεσή τους. Τα STORE_DATA και STORE_ADDRESS καταχωρούνται μαζί, καθώς πραγματοποιούνται μαζί και το STORE_ADDRESS

καταναλώνει τον περισσότερο χρόνο. Όλες οι υπόλοιπες εντολές είναι τύπου OP και για αυτό χωρίζονται σε κατηγορίες. Οι πιο γρήγορες εντολές είναι οι εντολές πρόσθεσης, που αποτελούν τις εντολές add και τα παράγωγά της, και αποτελούν και την πλειοψηφία των αριθμητικών εντολών στα προγράμματα που εξετάστηκαν. Οι εντολές αφαίρεσης αποτελούν και αυτές εντολές ίδιου τύπου και έχουν συμπεριληφθεί μαζί. Οι εντολές σύγκρισης περιέχουν κυρίως αφαιρέσεις, αλλά στα προγράμματα η χρήση τους είναι πολύ περιορισμένη και η συνεισφορά τους μηδαμινή, οπότε δε συμπεριλαμβάνονται. Οι εντολές ολίσθησης έχουν συμπεριληφθεί μαζί με τις εντολές πρόσθεσης, καθώς η λειτουργία τους είναι αρκετά απλή και δεν παρατηρήθηκε σημαντική διαφορά χρόνου με τις εντολές πρόσθεσης. Οι εντολές πολλαπλασιασμού είναι μία ειδική περίπτωση, αφού θεωρητικά διαρκούν περισσότερο χρόνο από τις εντολές πρόσθεσης λόγω του ότι είναι πολλές εντολές πρόσθεσης. Η μικρή τους παρουσία στα περισσότερα προγράμματα και το γεγονός ότι δε φαίνεται να απαιτούν σημαντικά περισσότερο χρόνο από τις εντολές πρόσθεσης, σημαίνει ότι δεν υπολογίζονται στα χαρακτηριστικά της πρόβλεψης ή έχουν ενωθεί με τις εντολές πρόσθεσης.

Σε όλα τα προγράμματα γίνεται χρήση πινάκων που ο χρόνος εκτέλεσής τους ποικίλλει. Άμα φορτώνεται για πρώτη φορά τιμή στο στοιχείο ενός πίνακα τότε γίνεται cache miss, οπότε ο χρόνος που απαιτείται για την ανάθεση είναι πολύ μεγάλος. Αν υπάρχει ήδη τιμή στο στοιχείο και δεν γίνεται cache miss, επιπλέον χρόνος απαιτείται για την όποια αλλαγή του στοιχείου του πίνακα αλλά είναι σημαντικά μικρότερος. Υπάρχουν και περιπτώσεις που η χρήση πινάκων δεν επιφέρει ουσιώδεις αυξήσεις στον χρόνο εκτέλεσης. Στις μετρήσεις της διπλωματικής τα φαινόμενα cache misses έχουν αγνοηθεί, αλλά η διαφορά της χρήσης πινάκων σε σχέση με τις άλλες αριθμητικές εντολές τους τοποθετούν σε άλλη κατηγορία. Σύμφωνα με το IACA, η χρήση πινάκων πραγματοποιείται σε σημεία που υπάρχει η εντολή μετατροπής cdqe που μετατρέπει από doubleword σε quadword. Στην ίδια κατηγορία τοποθετείται η εντολή μετατροπής cvtsi2sd που μετατρέπει doubleword ακέραιο σε διπλής ακριβείας floating-point τιμή, αν και η μικρή χρήση της σε όλα τα προγράμματα σημαίνει ότι δεν συνεισφέρει σημαντικά στις προβλέψεις.

Η εντολή div και τα παράγωγά της χρησιμοποιούν pipeline και περνούν από πολλά στάδια για την εκτέλεσή τους, οπότε ο χρόνος εκτέλεσής τους είναι σημαντικά μεγαλύτερος από τις άλλες αριθμητικές εντολές. Σε σχέση όμως με όλες τις υπόλοιπες εντολές η χρήση τους παρατηρείται σε μόνο λίγα προγράμματα και αυτή σε πολύ μικρά ποσοστά. Για αυτό το λόγο σε προγράμματα που υπάρχει τοποθετείται στην ίδια κατηγορία με την εντολή cdqe , καθώς αποτελούν εντολές τύπου OP που σχετίζονται με υψηλούς χρόνους εκτέλεσης.

Από τις υπόλοιπες εντολές δεν εξετάζονται ξεχωριστά οι ακόλουθες:

- Οι εντολές άλματος jmp και οι παράγωγές τους, οι οποίες έχουν περιορισμένη χρήση μέσα στον κώδικα και δεν συνεισφέρουν στις πράξεις που εκτελούνται.
- Οι εντολές lea, που παρατηρούνται όταν χρησιμοποιούνται δείκτες. Ο χρόνος για την εκτέλεσή τους είναι συγκρίσιμος με των αριθμητικών εντολών.
- Οι επεκτάσεις κάποιων από τις αριθμητικές εντολές. Για παράδειγμα υπάρχουν επεκτάσεις της cmp που δεν έχουν μετρηθεί επειδή δεν συνεισφέρουν σημαντικά στο χρόνο εκτέλεσης.
- Οι εντολές mov. Οι περισσότερες εντολές mov χρησιμοποιούνται όταν υπάρχει πρόσβαση στη μνήμη, δηλαδή στις περιπτώσεις LOAD και STORE , αλλά υπάρχουν και αναθέσεις μεταξύ καταχωρητών. Αυτές οι εντολές καταχωρούνται ως εντολές OP , αλλά ο ρόλος τους είναι λιγότερο σημαντικός από την περίπτωση της πρόσβασης στη μνήμη, οπότε αγνοούνται.

  Ωστόσο, όλες αυτές οι εντολές καταχωρούνται στην κατηγορία OP.

Για την καλύτερη πρόβλεψη των εξόδων , χρήσιμα είναι και τα ποσοστά που εμφανίζονται οι εντολές σε ολόκληρο τον κώδικα. Αν μία εντολή καταναλώνει πολύ χρόνο,

αλλά εμφανίζεται σε πολύ μικρά ποσοστά, τότε η συνεισφορά της στο συνολικό χρόνο δεν είναι σημαντική. Οι εντολές Load εμφανίζονται και στα benchmarks και στις μετρήσεις σε αρκετά υψηλά ποσοστά σε σχέση με άλλες χρονοβόρες εντολές, όπως τα stores και τα divs, οπότε είναι ο παράγοντας που επηρεάζει περισσότερο τα αποτελέσματα. Η ενέργεια εκτέλεσης συνδέεται στενά με το χρόνο ιδιαίτερα σε κώδικα C ,που είναι από τους πιο γρήγορους και ενεργειακά αποδοτικούς, οπότε οι παράγοντες που επηρεάζουν το χρόνο θα επηρεάζουν ανάλογα και την ενέργεια.

Επομένως για την πρόβλεψη τα χαρακτηριστικά (features) που θα χρησιμοποιηθούν περιλαμβάνουν τα ακόλουθα:

- Το προβλεπόμενο από το IACA throughput για μία συγκεκριμένη x86 αρχιτεκτονική.
- Ο αριθμός των micro-operations
- Ο αριθμός των εντολών LOAD
- Ο αριθμός των εντολών τύπου OP
- Ο αριθμός των εντολών STORE
- Ο αριθμός των εντολών που ανήκουν στην κατηγορία add. Αυτή περιλαμβάνει εντολές πρόσθεσης, αφαίρεσης, ολίσθησης και πολλαπλασιασμού.
- Ο αριθμός των εντολών που ανήκουν στην κατηγορία conv. Αυτή περιλαμβάνει τις εντολές μετατροπής που σχετίζονται με πίνακες και τις εντολές διαίρεσης, δηλαδή τις περιπτώσεις υψηλότερης πολυπλοκότητας.
- Τα ποσοστά για όλους τους τύπους των εντολών. Τα ποσοστά υπολογίζονται με βάση τον συνολικό αριθμό micro-operations.

Για την εύρεση των παραπάνω χαρακτηριστικών στα προγράμματα που δίνονται χρησιμοποιείται και άλλο script σε python. Αυτό χρησιμοποιεί τη clang για να υπολογίσει τα σημεία κώδικα που υπάρχουν τα κρίσιμα statements, δηλαδή οι δηλώσεις συναρτήσεων και τα σημεία που γίνονται επαναλήψεις. Όταν βρεθούν αυτά τα κρίσιμα σημεία τοποθετούνται οι κατάλληλοι δείκτες και γίνεται μεταγλώττιση χωρίς σημαία επιλογών σε object αρχείο για να μπορεί να εκτελεστεί το IACA.

Ως επιπλέον σημαίες στο IACA χρησιμοποιούνται το -trace για την δημιουργία trace αρχείου και η –trace-cycle-count 300 για την εμφάνιση περισσότερων κύκλων ως αποτέλεσμα, σε περίπτωση που κάποιο αρχείο έχει πολύ μεγάλο κώδικα. Ακόμα και στις περιπτώσεις μεγάλου κώδικα , η όλη διαδικασία της μεταγλώττισης και της εκτέλεσης του εργαλείου είναι πολύ γρήγορη και διαρκεί λιγότερο από δευτερόλεπτο. Τα αποτελέσματα της ανάλυσης αποθηκεύονται σε ένα αρχείο και δημιουργείται το trace αρχείο.

Από αυτά τα αρχεία γίνεται υπολογισμός των features ξεχωριστά για κάθε συνάρτηση και σώμα επαναλήψεων. Στην περίπτωση που γίνεται κλήση συναρτήσεων μέσα σε ένα σώμα επαναλήψεων τα χαρακτηριστικά των συναρτήσεων προστίθενται σε αυτά των επαναλήψεων

Η εφαρμογή του παραπάνω script στα προγράμματα των benchmarks επιστρέφει ένα αρχείο που περιέχει όλες τις συναρτήσεις και επαναλήψεις μαζί με τα γνωρίσματά τους. Από τα προγράμματα επιλέγονται μόνο μία ή μερικές επαναλήψεις οπότε όλα τα υπόλοιπα αγνοούνται. Ακόμα και σε περιπτώσεις που το πρόγραμμα περιέχει πολλά και μεγάλα κομμάτια κώδικα, ο υπολογισμός όλων των γνωρισμάτων διαρκεί μερικά δευτερόλεπτα.

2.3 Μετρικές εκτέλεσης

Τα γνωρίσματα που απορρέουν από το IACA προκύπτουν σε λίγο χρόνο, αλλά δεν δίνουν επαρκείς πληροφορίες για το χρόνο και την ενέργεια εκτέλεσης. Για να είναι δυνατή

η πρόβλεψη χρειάζεται και ο αριθμός των εντολών που είναι το πιο κρίσιμο χαρακτηριστικό. Για τον υπολογισμό των εντολών χρησιμοποιείται το εργαλείο pin [46]. Το pin είναι ένα εργαλείο που χρησιμοποιεί ένα αρχείο συνήθως σε C ή C++ αποκαλούμενο pintool με το οποίο κάνει υπολογισμούς πάνω σε ένα εκτελέσιμο αρχείο. Το πιο βασικό αρχείο pintool είναι εκείνο που υπολογίζει τον αριθμό των εντολών σε ένα εκτελέσιμο πρόγραμμα. Χρησιμοποιώντας τις κατάλληλες σημαίες μπορεί να υπολογιστούν οι εντολές σε συγκεκριμένα κομμάτια κώδικα [47].

Αυτές οι εντολές χρησιμοποιούνται ως το τελικό feature για την πραγματοποίηση της πρόβλεψης. Από αυτό το σημείο όταν γίνεται αναφορά στο σύνολο των features θα σημαίνουν τα ακόλουθα:

- Πρόβλεψη με 2 ή 3 features σημαίνει ότι πέρα από τις εντολές χρησιμοποιείται το throughput ή ο αριθμός micro-operations ή και τα δύο. Αυτό γίνεται επειδή είναι τα πιο σημαντικά χαρακτηριστικά που προκύπτουν από το IACA.
- Πρόβλεψη με χρήση από 4 μέχρι 8 features σημαίνει ότι ως επιπλέον χαρακτηριστικά χρησιμοποιούνται οι αριθμοί των εντολών. Πέρα από την περίπτωση των 8 features, σε όλες τις άλλες περιπτώσεις θα δηλώνεται ακριβώς ποιες εντολές χρησιμοποιήθηκαν.
- Πρόβλεψη με χρήση από 9 μέχρι 13 features σημαίνει ότι ως επιπλέον χαρακτηριστικά χρησιμοποιούνται τα ποσοστά των εντολών. Χρησιμοποιούνται τα ποσοστά των εντολών που έχουν οριστεί ως features.

Τα παραπάνω θα έχουν αυτή τη σημασία όταν αναφέρονται μέσα στη διπλωματική εκτός από όταν δηλώνεται διαφορετικά με σαφήνεια.

Ο χρόνος και η ενέργεια πρέπει να υπολογιστούν στην πλακέτα. Η πλακέτα που εξετάζεται είναι η Nvidia Tegra TX1 και οι μετρήσεις αφορούν τον επεξεργαστή ARM Cortex A57. Ο υπολογισμός του χρόνου είναι σχετικά απλός και μπορεί να γίνει με την εισαγωγή συναρτήσεων υπολογισμού χρόνου στη C, όπως το clock() και το gettimeofday(). Ο υπολογισμός της ενέργειας μπορεί να γίνει σε αυτήν την πλακέτα καθώς είναι αρκετά περίπλοκη, περιλαμβάνει αισθητήρα (INA3221) και παρέχει software accesses στα δεδομένα του για να γίνει ο υπολογισμός της ισχύος . Αυτά μαζί με το χρόνο δίνουν την ενέργεια σε επίπεδο module, CPU και GPU. Από αυτές η πιο σημαντική είναι η ενέργεια CPU, ενώ οι άλλες έχουν πολύ μικρή τιμή για να επηρεάσουν σημαντικά την συνολική ενέργεια. Πέρα από ειδικές περιστάσεις η ενέργεια θα προκύπτει από το άθροισμα των ενεργειών module, CPU και GPU.

Επομένως ως έξοδοι για την πρόβλεψη χρησιμοποιούνται τα δύο μεγέθη:

- Χρόνος ανά CPU που μετριέται σε milliseconds.
- Συνολική ενέργεια που καταναλώθηκε μετρημένη σε Joule

Όλα τα γνωρίσματα και οι έξοδοι για τα προγράμματα τοποθετούνται μαζί σε ένα dictionary. Σε αρκετά προγράμματα έχουν γίνει υπολογισμοί για διαφορετικές τιμές εισόδων, οπότε οι χρόνοι και οι ενέργειες εκτέλεσης είναι διαφορετικές, αλλά όλα τα υπόλοιπα γνωρίσματα διατηρούνται ίδια καθώς εξάγονται και εξαρτώνται μόνο από τον πηγαίο κώδικα.

2.4 Δεδομένα για την πρόβλεψη – Συνθετικό dataset

Δημιουργούνται μερικά συνθετικά προγράμματα επαναλήψεων που χρησιμοποιούνται ως training σύνολο, ενώ τα προγράμματα από τα benchmarks χρησιμοποιούνται ως test σύνολο. Τα dummy προγράμματα είναι απλά προγράμματα σε C

που περιλαμβάνουν μόνο τη main συνάρτηση, μερικές αναθέσεις και κάποια επανάληψη for που μπορεί να είναι απλή ή να περιέχει κάποιες εμφωλευμένες.

Για την κατασκευή των τυχαίων συνθετικών προγραμμάτων χρησιμοποιείται ακόμα ένα script σε python. Αυτό δημιουργεί ένα τυχαίο σύνολο μεταβλητών που αρχικοποιούνται με τυχαίες τιμές από ένα σύνολο πραγματικών αριθμών. Ο αριθμός των μεταβλητών δεν είναι πολύ μικρός για να μην υπάρχουν σφάλματα της μορφής memory misses και κυμαίνεται από 5 σε 13 μεταβλητές. Οι τιμές κυμαίνονται από 1 σε 2000 για να είναι πιο τυχαίοι οι χρόνοι που χρειάζονται οι αριθμητικές πράξεις.

Το script , επίσης , επιλέγει τυχαία πόσα εμφωλευμένα σώματα επαναλήψεων θα υπάρχουν και πόσες επαναλήψεις θα είναι το καθένα. Αυτό είναι σημαντικό γιατί προσδιορίζει πόσο χρονοβόρο θα είναι το πρόγραμμα. Στις περισσότερες περιπτώσεις καλύπτεται ένα μεγάλο εύρος επαναλήψεων από μικρότερες σε μεγαλύτερες, αλλά σε περιπτώσεις που ζητούνται πολύ γρήγορα ή αργά προγράμματα το εύρος που καλύπτεται μειώνεται για να συμπεριλαμβάνει μόνο χαμηλές ή υψηλές τιμές επαναλήψεων.

Το επόμενο στάδιο είναι η κατασκευή του κώδικα που βρίσκεται μέσα στα for. Η κατασκευή χρειάζεται δύο μεγέθη: τον αριθμό των εντολών σε C και τον αριθμό των πράξεων σε κάθε εντολή. Επειδή η κλίμακα αυτών των μεγεθών είναι πολύ μικρότερη από τον αριθμό των επαναλήψεων, υπάρχει μεγαλύτερη ποικιλία στην επιλογή των τιμών τους. Όλες οι εντολές είναι εντολές ανάθεσης, που αποθηκεύουν το αποτέλεσμα κάποιων πράξεων σε μία μεταβλητή. Οι πράξεις γίνονται μεταξύ μεταβλητών και είναι αριθμητικές πράξεις που περιέχουν προσθέσεις, αφαιρέσεις και πολλαπλασιασμούς. Οι προσθέσεις και αφαιρέσεις υπολογίζονται στην ίδια κατηγορία , αλλά οι πολλαπλασιασμοί δεν χρησιμοποιούνται σε καμία κατηγορία. Όπως έχει αναφερθεί η μικρή παρουσία πολλαπλασιασμών στα προγράμματα των benchmarks, οδηγεί στο να αγνοούνται ως κατηγορία. Ωστόσο, η ισχυρή παρουσία τους στα τυχαία for οδηγεί σε μία καλή αναπαράσταση των προγραμμάτων στα benchmarks, όπου ένα σημαντικό μέρος των εντολών δεν καλύπτεται από τις εντολές πρόσθεσης. Οι πολλαπλασιασμοί , δηλαδή , αναπαριστούν αρκετά καλά αυτές τις εντολές χωρίς να οδηγούν σε μεγάλη αύξηση στο χρόνο εκτέλεσης. Η κάθε πράξη επιλέγεται τυχαία.

Οι εντολές div δεν υπάρχουν στα for , καθώς έχουν πολύ χαμηλά ποσοστά σε όλα τα προγράμματα και η μεγάλη παρουσία τους μπορεί να επηρεάσει σημαντικά τους χρόνους εκτέλεσης των for. Για να συμπεριληφθούν οι μετατροπές χρησιμοποιούνται πίνακες. Πίνακες μπορούν να χρησιμοποιηθούν ως right και left value σε μία ανάθεση με πιθανότητα μικρότερη από τις απλές μεταβλητές. Πέρα από την πρώτη επανάληψη που δεν υπάρχει τιμή στο στοιχείο του πίνακα που γίνεται ανάθεση, δεν υπάρχουν cache misses, οπότε ο χρόνος δεν αυξάνεται από αυτό. Στα περισσότερα for οι πίνακες κάνουν μία αισθητή αλλά όχι κυρίαρχη παρουσία και αυτό είναι κατάλληλο για να παρατηρηθεί η επιρροή τους στην πρόβλεψη.

Μετά από αυτό το στάδιο ο κώδικας ολοκληρώνεται.

Αυτό το script επαναλαμβάνεται πολλές φορές για να καλύψει μία μεγάλη ποικιλία από μεγέθη προγραμμάτων. Τα προγράμματα αυτά μεταφέρονται στην πλακέτα, όπου επιλέγεται το καθένα και υπολογίζονται η ενέργεια και ο χρόνος εκτέλεσής του.

Παρόμοια διαδικασία χρησιμοποιείται και για τον υπολογισμό των εντολών με το pin. Η διαδικασία υπολογισμού εντολών με το pin διαρκεί περισσότερο από τον υπολογισμό του χρόνου εκτέλεσης. Στα προγράμματα των benchmarks οι εντολές υπολογίστηκαν με το pin λόγω της περίπλοκης μορφής τους, αλλά η απλότητα των for οδηγεί σε έναν πολύ πιο γρήγορο υπολογισμό τους. Με τη χρήση του IACA μπορούν να υπολογιστούν οι εντολές στο

εσωτερικό κάθε σώματος επαναλήψεων και αν είναι γνωστός ο αριθμός επαναλήψεων μπορεί να υπολογιστεί ο συνολικός αριθμός εντολών πολλαπλασιάζοντας τα δύο μεγέθη. Η εύρεση του αριθμού επαναλήψεων πραγματοποιείται με ένα απλό διάβασμα των for που έχουν όλα την ίδια μορφή, οπότε οι επαναλήψεις βρίσκονται στο ίδιο σημείο. Συγκρίνοντας τα αποτελέσματα με τις μετρήσεις από το pin παρατηρείται ότι η διαφορά τους είναι πολύ μικρή, άρα η μέθοδος είναι αξιόπιστη και ταυτόχρονα πολύ πιο γρήγορη.

Τα προγράμματα από τα benchmarks καλύπτουν συγκεκριμένους χρόνους εκτέλεσης και τα dummy προγράμματα αρκεί να καλύπτουν αυτό το εύρος. Για αυτό το λόγο τα συνθετικά προγράμματα ικανοποιούν τις παρακάτω ιδιότητες:

- Το μεγαλύτερο πλήθος προγραμμάτων καλύπτουν χρόνους εκτέλεσης από 1 second σε μερικές δεκάδες seconds .
- Κάποια προγράμματα καλύπτουν χρόνους λιγότερους του ενός second και αυτό επειδή έχουν κατασκευαστεί με περιορισμένο εύρος τυχαιότητας.
- Μερικά προγράμματα καλύπτουν χρόνους από δεκάδες σε εκατοντάδες seconds και αυτά έχουν επίσης κατασκευαστεί με περιορισμένο εύρος τυχαιότητας.

Για την καλύτερη κάλυψη όλων των περιπτώσεων και για να υπάρχει περισσότερη ελευθερία για μελλοντικές αλλαγές στο σύνολο δεδομένων, ο αριθμός των μετρήσεων έχει φτάσει τις μερικές χιλιάδες.

3. Πρόβλεψη

3.1 Μοντέλο regression σε όλες τις επαναλήψεις

Για την πρόβλεψη χρειάζονται τα ακόλουθα δεδομένα: ένα σύνολο χαρακτηριστικών που χρησιμοποιούνται ως παράγοντες για την πρόβλεψη, οι έξοδοι, δηλαδή τα μεγέθη που προβλέπονται από το μοντέλο, ένα σύνολο δειγμάτων που λειτουργούν ως training σύνολο και ένα σύνολο μετρήσεων που λειτουργούν ως test σύνολο. Το training σύνολο αποτελεί το σύνολο των συνθετικών προγραμμάτων μαζί με τα χαρακτηριστικά τους. Το test σύνολο αποτελεί είτε το σύνολο των προγραμμάτων από τα benchmarks μαζί με τα γνωρίσματά τους είτε ένα υποσύνολο του συνόλου των συνθετικών προγραμμάτων.

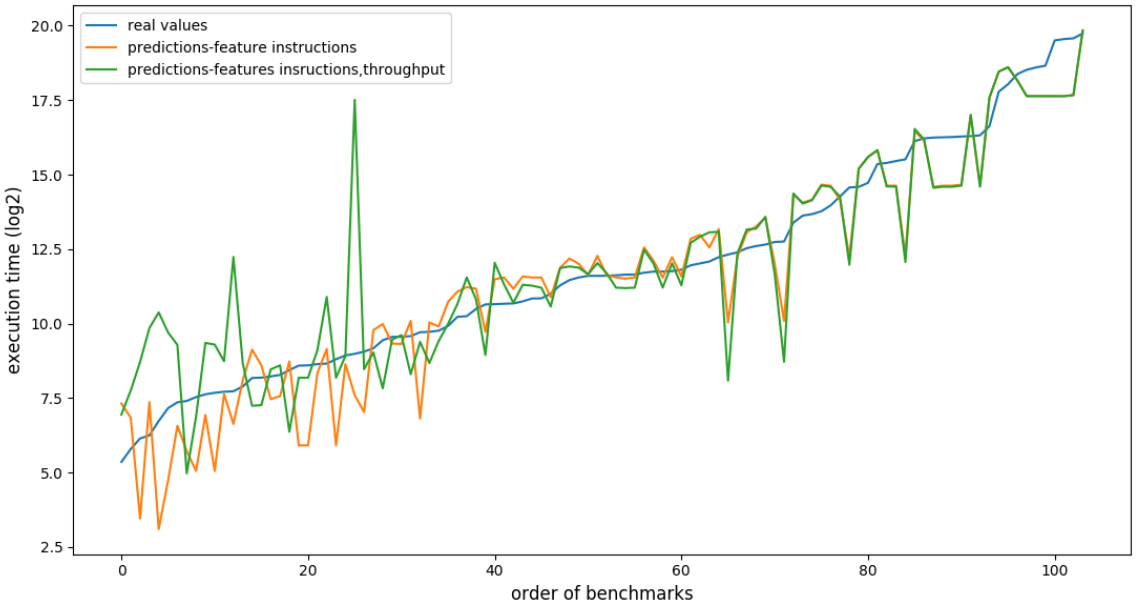
Ένα script σε python δέχεται όλα τα δεδομένα και τα διαχωρίζει σε training και test σύνολα σύμφωνα με την παραπάνω τεχνική. Επειδή τα μεγέθη των features διαφέρουν πολύ μεταξύ τους για να γίνει καλύτερη πρόβλεψη περνάνε από μία διαδικασία κανονικοποίησης. Η πρόβλεψη γίνεται χρησιμοποιώντας μία μέθοδο regression, η οποία κάνει ταίριασμα στα δεδομένα του training συνόλου και χρησιμοποιώντας τα features του test συνόλου κάνει πρόβλεψη για τις τιμές των εξόδων. Πραγματοποιούνται δύο είδη προβλέψεων: μία που έχει ως feature μόνο τον αριθμό των εντολών και μία που χρησιμοποιεί περισσότερα features. Αυτό γίνεται επειδή οι εντολές είναι το πιο σημαντικό από τα χαρακτηριστικά και συγκρίνοντας τις προβλέψεις φαίνεται η συνεισφορά των άλλων features. Ο αριθμός των features που εξετάζεται μπορεί να αλλάξει ανάλογα με την πρόβλεψη που πραγματοποιείται.

Οι πραγματικές τιμές των εξόδων του test συνόλου και οι τιμές που έχουν προβλεφθεί μπορούν να χρησιμοποιηθούν για τον υπολογισμό μετρικών που δείχνει τη σχέση μεταξύ τους. Μετρικές όπως το μέσο απόλυτο σφάλμα δεν είναι χρήσιμες, αφού οι έξοδοι παίρνουν τιμές από μερικές δεκάδες μέχρι εκατοντάδες χιλιάδες και το σφάλμα που επιστρέφει επηρεάζεται κυρίως από τις πολύ υψηλές τιμές και δεν δίνει κάποια πληροφορία για τις προβλέψεις των υπόλοιπων τιμών. Από τις διάφορες μετρικές αυτές που έχουν κάποια σημασία είναι το explained_variance_score και το r2_score, καθώς δίνουν μία εικόνα για το πόσο απέχουν οι προβλέψεις από τις πραγματικές τιμές μέσω της διασποράς. Γενικά, το explained_variance_score είναι πάντα μεγαλύτερο του r2_score και έχει μικρότερες μεταβολές μεταξύ των προβλέψεων, οπότε δεν του δίνεται τόση σημασία.

Για την παρουσίαση των προβλέψεων σε γράφημα πραγματοποιείται λογαριθμική μετατροπή όλων των εξόδων, διότι κάποιες τιμές είναι πολύ μεγάλες και σε διαφορετική περίπτωση θα εμφανίζονταν μόνο αυτές οι λίγες και οι υπόλοιπες δεν θα διακρίνονταν. Οι έξοδοι ταξινομούνται για να είναι πιο σαφής η διαφορά μεταξύ πραγματικών τιμών και προβλέψεων. Σε κάθε γράφημα παρουσιάζονται σε μία καμπύλη οι πραγματικές τιμές, οι προβλεπόμενες τιμές χρησιμοποιώντας ως feature μόνο τις εντολές και οι προβλεπόμενες τιμές με features επιπλέον χαρακτηριστικά.

Ο βασικός παράγοντας που επηρεάζει αν θα βγει καλή πρόβλεψη είναι η μέθοδος regression που καταχωρείται στη μεταβλητή. Έχουν δοκιμαστεί μερικές μέθοδοι για να εντοπιστεί η καλύτερη μέθοδος καθώς και ο αριθμός των χαρακτηριστικών που βελτιστοποιούν την πρόβλεψη. Στις παρακάτω μεθόδους οι προβλέψεις γίνονται με training σύνολο το σύνολο όλων των for και με test σύνολο το σύνολο των benchmarks. Η έξοδος που αναζητείται είναι ο χρόνος εκτέλεσης. Ο σκοπός τους δεν είναι η επιλογή της καλύτερης μεθόδου, αλλά να εξεταστεί η επιρροή που έχει η προσθήκη επιπλέον χαρακτηριστικών στην πρόβλεψη.

LinearRegression() : Αποτελεί την πιο βασική μέθοδος πρόβλεψης και στο παρακάτω διάγραμμα δίνονται οι πραγματικές τιμές για το χρόνο , η πρόβλεψη χρησιμοποιώντας μόνο εντολές και η πρόβλεψη χρησιμοποιώντας εντολές και throughput:



Σχήμα 5.1 : Προβλέψεις σε σχέση με τις πραγματικές τιμές του χρόνου εκτέλεσης με χρήση LinearRegression

Η μπλε γραμμή συμβολίζει τις πραγματικές τιμές, η πορτοκαλί τις προβλέψεις χρησιμοποιώντας μόνο τις εντολές και η πράσινη τις προβλέψεις χρησιμοποιώντας τις εντολές και το throughput. Όπως φαίνεται οι προβλέψεις είναι πολύ κοντά για τις υψηλότερες τιμές και έχουν μεγαλύτερες διακυμάνσεις για χαμηλότερες τιμές. Ιδιαίτερα η πράσινη καμπύλη κάνει πολύ μεγαλύτερες διακυμάνσεις σε αυτό το σημείο , ενώ η πορτοκαλί κάνει διακυμάνσεις παρόμοιας κλίμακας σε όλο το διάγραμμα. Το συμπέρασμα είναι ότι το μοντέλο χρησιμοποιεί τις εντολές ως το πιο ισχυρό χαρακτηριστικό και δεν μπορεί να αφομοιώσει οποιοδήποτε άλλο χαρακτηριστικό με αποτέλεσμα να χαλάει η ποιότητα της καμπύλης. Για περισσότερα γνωρίσματα , δηλαδή, η καμπύλη σε αυτές τις τιμές γίνεται πολύ πιο ανακριβής.

Ridge(): Η μέθοδος Ridge διαφέρει από το LinearRegression σε κάποια σημεία: Η πρόβλεψη με feature μόνο τις εντολές έχει μία απόκλιση από τις πραγματικές τιμές που μεγαλώνει όσο μικραίνει η τιμή τους. Οι προβλέψεις με χρήση παραπάνω features διατηρούν κάποιες διακυμάνσεις, αλλά αποκλίνουν λιγότερο από την καμπύλη των πραγματικών τιμών.

Το RidgeCV φαίνεται να δίνει αποτελέσματα ανάλογα με το LinearRegression, δηλαδή οι προβλέψεις αποκλίνουν λιγότερο για μικρότερο αριθμό χαρακτηριστικών. Αν ως features χρησιμοποιούνται και τα ποσοστά εμφάνισης των εντολών, τότε η απόκλιση γίνεται πολύ μεγαλύτερη. Αυτό παρατηρείται ανεξάρτητα από την τιμή που παίρνει η παράμετρος για το cross validation.

SGDRegressor(): Λειτουργεί όπως και το Ridge, με τις προβλέψεις γενικά να έχουν μία μικρότερη απόκλιση ακόμα και στις περιπτώσεις όλων των χαρακτηριστικών.

ElasticNet(): Η μέθοδος ElasticNet είναι μία περίπτωση που δεν μπορεί να γίνει πρόβλεψη στα δεδομένα.

Το μοντέλο δεν μπορεί να αξιοποιήσει τις εντολές για να κάνει πρόβλεψη, οπότε αφού αυτές είναι το σημαντικότερο χαρακτηριστικό όλες οι υπόλοιπες δοκιμές για πρόβλεψη θα οδηγήσουν σε αποτυχία.

Το ElasticNetCV δε δίνει διαφορετικά αποτελέσματα.

Lars(): Η μέθοδος δίνει προβλέψεις που ακολουθούν τη λογική του LinearRegression, δηλαδή τα επιπλέον χαρακτηριστικά δεν προσφέρουν καλύτερη πρόβλεψη. Παρόμοιο είναι και το LarsCV με τη χρήση cross validation, αν και η απόκλιση είναι λίγο μικρότερη.

Ενδιαφέρον παρατηρείται στην περίπτωση με LarsCV αλλά χωρίς παράμετρο cross validation, όπου τα επιπλέον χαρακτηριστικά βοηθούν την πρόβλεψη και παρατηρείται μικρότερη διασπορά και απόκλιση από τις πραγματικές τιμές σε σχέση με την πρόβλεψη με feature μόνο τις εντολές.

Lasso(): Παρατηρείται ότι οι προβλέψεις με feature μόνο τις εντολές έχουν περίπου την ίδια απόκλιση με τις προβλέψεις με χρήση 8 features. Για τις μικρότερες τιμές οι προβλέψεις με χρήση μόνο εντολών παίρνουν τιμές μικρότερες, ενώ οι προβλέψεις με 8 features παίρνουν τιμές μεγαλύτερες.

LassoCV(): Σε αυτήν την περίπτωση παρατηρείται οι προβλέψεις να παίρνουν ένα σύνολο τιμών παρόμοιο με το LinearRegression. Πιο συγκεκριμένα, όταν χρησιμοποιούνται 3 ή λιγότερα features για την μία πρόβλεψη τότε οι δύο τύποι προβλέψεων είναι πανομοιότυποι. Όσο αυξάνεται ο αριθμός των features που χρησιμοποιούνται στην μία πρόβλεψη τόσο περισσότερο αποκλίνει από την άλλη και τις πραγματικές τιμές.

LassoLars(): Φαίνεται ότι ακολουθεί πολύ τη λογική του Lars, τόσο το απλό μοντέλο όσο και το μοντέλο cross validation.

OrthogonalMatchingPursuit(): Η πρόβλεψη με επιπλέον features δε φαίνεται να διαφέρει από την πρόβλεψη με μόνο feature τις εντολές.

Η περίπτωση του cross validation δεν μπορεί να πραγματοποιήσει προβλέψεις με ένα μόνο feature, οπότε δεν μπορούν να γίνουν συγκρίσεις.

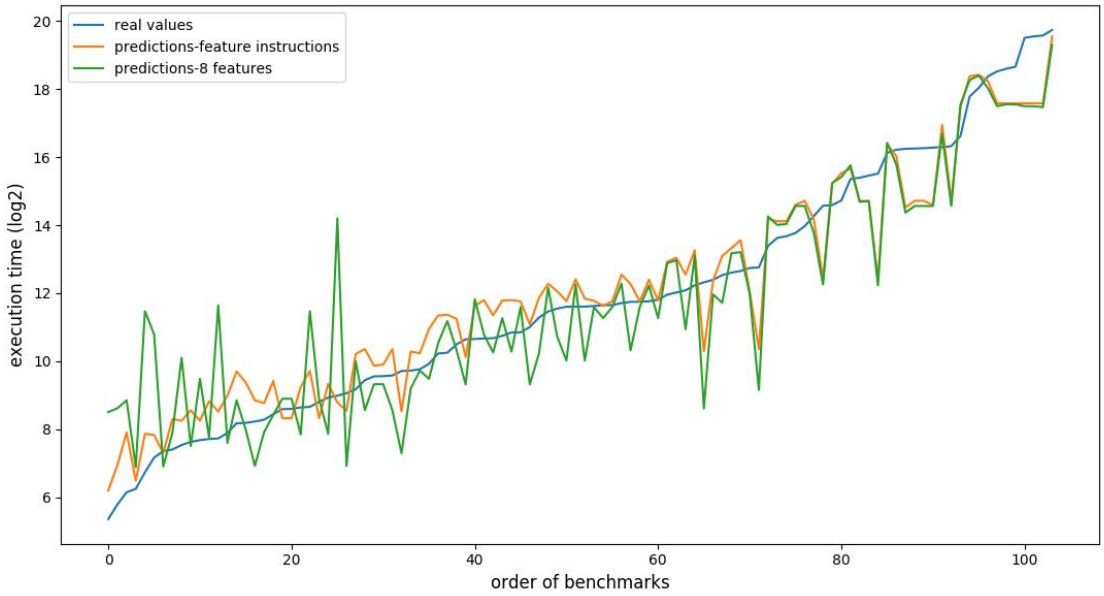
BayesianRidge(): Παρόμοια αποτελέσματα με το LinearRegression

HuberRegressor(): Τα αποτελέσματα για τις δύο προβλέψεις δε φαίνεται να έχουν κάποια μεγάλη διαφορά και αποκλίνουν και τα δύο από τις μικρότερες πραγματικές τιμές.

RANSACRegressor(): Παρόμοια αποτελέσματα με τον HuberRegressor, αν και οι αποκλίσεις είναι μικρότερες.

TheilSenRegressor(): Από όλες τις μεθόδους που έχουν χρησιμοποιηθεί είναι η πιο αργή. Όσο μεγαλώνει ο αριθμός των features τόσο μεγαλώνει και η διασπορά που γίνεται κατά την πρόβλεψη.

KneighborsRegressor(): Είναι από τις μεθόδους που οι αλλαγές στον αριθμό των features έχουν επίδραση στις προβλέψεις και για τις υψηλότερες τιμές, όπου σε όλα τα υπόλοιπα μοντέλα οι προβλέψεις για αυτές τις τιμές δεν είχαν διαφορές. Ωστόσο, η επίδραση των επιπλέον χαρακτηριστικών δε φαίνεται να ευνοεί την πρόβλεψη, καθώς μεγαλώνει τη διασπορά.



Σχήμα 5.2 : Προβλέψεις σε σχέση με πραγματικές τιμές του χρόνου εκτέλεσης με χρήση του KneighborsRegressor με 10 γείτονες

MLPRegressor(): Είναι πολύ αργή μέθοδος και το αποτέλεσμα θυμίζει την περίπτωση του ElasticNet. Μπορεί να αναγνωρίσει την επίδραση των επιπλέον χαρακτηριστικών, αλλά αδυνατεί να κάνει πρόβλεψη με μόνο feature τις εντολές , οπότε αποτυγχάνει συνολικά.

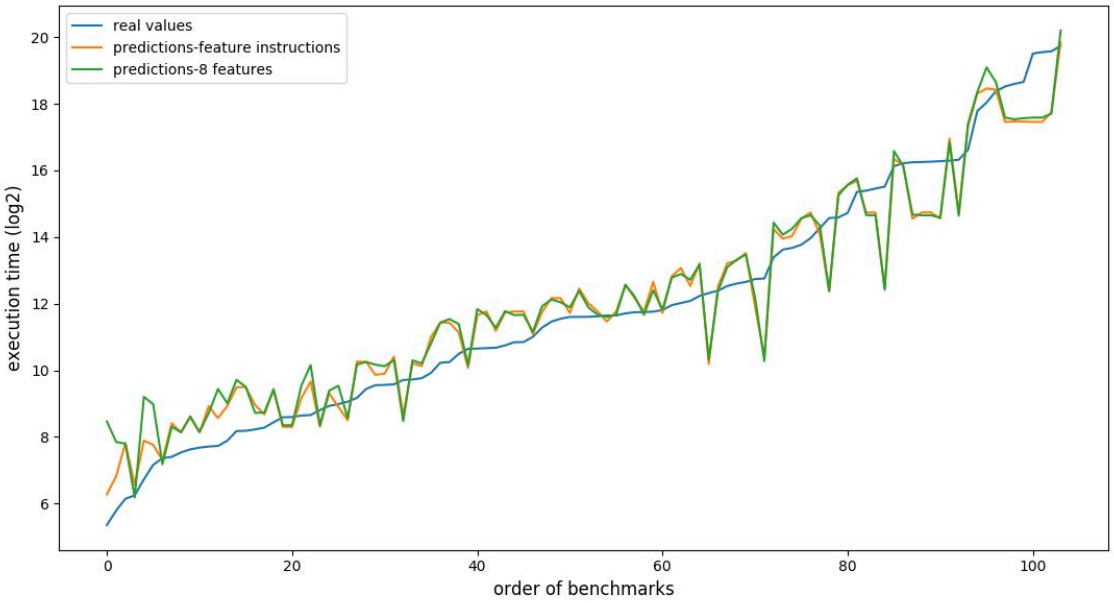
LinearSVR(): Όπως και το ElasticNet, η μέθοδος αποτυγχάνει να κάνει πρόβλεψη. Αυτό το πρόβλημα παρατηρείται σε όλες τις μεθόδους regression της μονάδας sklearn.svm και αυτό περιλαμβάνει τα SVR και NuSVR.

DecisionTreeRegressor(): Είναι από τις λίγες μεθόδους που οι προβλέψεις επηρεάζονται από τα επιπλέον χαρακτηριστικά για κάθε μέγεθος τιμών και δεν επηρεάζονται σε βαθμό που οδηγεί σε μεγάλες αποκλίσεις ή διασπορές.

Η μέθοδος ExtraTreeRegressor δεν δίνει σημαντικά διαφορετικά αποτελέσματα, αλλά έχει λίγο μικρότερες διακυμάνσεις σε κάποια σημεία.

Για να αντιμετωπιστούν τα μειονεκτήματα των δέντρων απόφασης χρησιμοποιείται το BaggingRegressor, που κάνει έναν αριθμό εκτιμήσεων χρησιμοποιώντας ως εκτιμητή το δέντρο και βγάζει τον μέσο όρο τους.

Χρησιμοποιώντας οχτώ features, τη μέθοδο ExtraTreeRegressor και 100 εκτιμήσεις, το BaggingRegressor δίνει το ακόλουθο διάγραμμα:



Σχήμα 5.3 : Προβλέψεις σε σχέση με τις πραγματικές τιμές του χρόνου εκτέλεσης με χρήση του BaggingRegressor με 100 εκτιμήσεις πάνω στο ExtraTreeRegressor.

Όπως φαίνεται οι δύο περιπτώσεις προβλέψεων δεν διαφέρουν πολύ.

3.1.1 Overfitting

Ένα κοινό χαρακτηριστικό όλων των διαγραμμάτων ήταν ότι για υψηλότερες τιμές οι προβλέψεις δεν διαφέρουν σημαντικά και σε κάποιες περιπτώσεις ταυτίζονται. Ένας λόγος που μπορεί να συμβαίνει αυτό είναι το overfitting, δηλαδή υπάρχουν πολλές κοντινές μετρήσεις με αποτέλεσμα να αλληλοκαλύπτονται και έτσι να μην φαίνεται διαφορά στο διάγραμμα. Μία απλή μέθοδος να μειωθεί το overfitting είναι να μειωθεί ο αριθμός των εντολών και μία τεχνική που εφαρμόζεται είναι η επιλογή του ενός δέκατου όλων των δεδομένων. Τα συνθετικά προγράμματα έχουν δημιουργηθεί από διαδοχικές εφαρμογές ενός script με διαφορετικά εύρη τιμών και για αυτό υπάρχουν υποσύνολα από αυτές που έχουν κοινά χαρακτηριστικά. Με την τεχνική μείωσης των δεδομένων επιλέγεται το ένα δέκατο των δεδομένων από κάθε τέτοιο σύνολο, οπότε δε μειώνεται το εύρος που καλύπτει η πρόβλεψη.

Η μείωση αυτή δεν έχει καλά αποτελέσματα για τις περισσότερες μεθόδους και οι δύο περιπτώσεις προβλέψεων αποκλίνουν περισσότερο από τις πραγματικές τιμές.

Το ExtraTreeRegressor για την πρόβλεψη βάσει των 3 features βελτιώνεται λίγο από την πρόβλεψη με βάση τις εντολές, ενώ όταν υπάρχουν περισσότερα features οι προβλέψεις είναι περίπου ίδιες μεταξύ τους.

Το DecisionTreeRegressor είναι το μόνο διάγραμμα που υπάρχει κάποια βελτίωση με την προσθήκη επιπλέον features. Ωστόσο, η μόνη μεγάλη βελτίωση παρατηρείται μόνο σε μερικές υψηλές τιμές, ενώ σε όλες τις υπόλοιπες η πρόβλεψη με επιπλέον features προσεγγίζει λίγο καλύτερα από την πρόβλεψη με μόνο εντολές.

3.1.2 Συμπεράσματα

Οι προβλέψεις που γίνονται με τα χαρακτηριστικά που έχουν συγκεντρωθεί δεν είναι επαρκείς. Σε όλα τα μοντέλα ο αριθμός των εντολών παίζει τον κυρίαρχο ρόλο και επικαλύπτει τον ρόλο των άλλων χαρακτηριστικών για την πρόβλεψη. Αυτό σημαίνει ότι οι δύο προβλέψεις που γίνονται δεν διαφέρουν σημαντικά μεταξύ τους και όταν διαφέρουν σημαίνει ότι δε γίνεται καλή πρόβλεψη.

Το συμπέρασμα είναι ότι η κλίμακα των μεγεθών που χρησιμοποιούνται είναι πολύ μεγάλη με βασικό παράδειγμα τις εντολές που φθάνουν σε αριθμό πάνω από δεκάδες εκατομμύρια σε σχέση με τις μετρήσεις από το IACA που το περισσότερο που φθάνουν είναι η χιλιάδα. Το ίδιο συμβαίνει και με τις εξόδους με αποτέλεσμα οι μετρικές regression να μην δίνουν πληροφορία για τις προβλέψεις. Αυτό το πρόβλημα δε λύνεται με κανονικοποίηση. Για να υπάρξουν καλύτερες προβλέψεις χρειάζεται η μείωση της κλίμακας που φτάνουν τα δεδομένα και καλύτερη αξιοποίηση των μεγεθών που παράγει το IACA.

3.2 Πρόβλεψη regression ανά επανάληψη

Μία μέθοδος για την αποφυγή των μεγάλων τιμών στα μεγέθη είναι αντί για συνολικό αριθμό εντολών να χρησιμοποιείται ο αριθμός των εντολών σε μία επανάληψη. Αυτός προκύπτει από τη στατική ανάλυση του basic block. Με αυτόν τον τρόπο το μοντέλο δουλεύει καλύτερα για στατική ανάλυση, καθώς δεν περιλαμβάνει ως feature τις συνολικές εντολές που αποτελούν δυναμική πληροφορία.

Κατά την κατασκευή του μοντέλου πρόβλεψης ο αριθμός των εντολών χρησιμοποιείται κατευθείαν όπως προκύπτει από το IACA και για αυτό το λόγο είναι πολύ πιο συνδεδεμένος με τα υπόλοιπα χαρακτηριστικά. Μετά την μετατροπή ο χρόνος και η ενέργεια έχουν πολύ μικρές δεκαδικές τιμές, οπότε μπορεί να χρησιμοποιηθεί ως μετρική σφάλματος και το μέσο απόλυτο σφάλμα.

Για την αποφυγή του overfitting χρησιμοποιείται η ιδέα του clustering [48]. Σύμφωνα με αυτήν την ιδέα, τα δεδομένα μπορούν να σπάσουν βασιζόμενα σε κάποια χαρακτηριστικά σε πολύ μικρά σύνολα που ονομάζονται clusters. Στο script κατασκευής έχουν γίνει δοκιμές για διαχωρισμό σε clusters με βάση όλα τα features ή όλα εκτός από τις εντολές και ο διαχωρισμός γίνεται με τον αλγόριθμο kmeans. Ο αλγόριθμος kmeans χωρίζει τα δεδομένα σε clusters έτσι ώστε να υπάρχει ίση διασπορά μεταξύ τους και με στόχο την ελαχιστοποίηση της απόστασης των εσωτερικών τους σημείων από ένα φαινομενικό κέντρο. Το φαινομενικό κέντρο επιλέγεται από τα διάφορα δεδομένα και ο αλγόριθμος τοποθετεί τα σημεία στο πιο κοντινό κέντρο.

Η μέθοδος είναι να τοποθετηθούν όλα τα δεδομένα σε clusters με βάση τα features του training συνόλου και από κάθε cluster θα επιλεχθεί ένα στοιχείο που θα χρησιμοποιηθεί σε ένα καινούριο training σύνολο.

Όπως και στις προηγούμενες προβλέψεις σκοπός των παρακάτω δοκιμών είναι να εντοπιστούν τα μοντέλα που μπορούν να πραγματοποιήσουν προβλέψεις με τη χρήση των επιπλέον features. Οι πραγματικές τιμές είναι ταξινομημένες και αναπαρίστανται με μία μπλε καμπύλη, οι προβλέψεις με χρήση μόνο εντολών με μία πορτοκαλί καμπύλη και οι προβλέψεις με περισσότερα features με μία πράσινη.

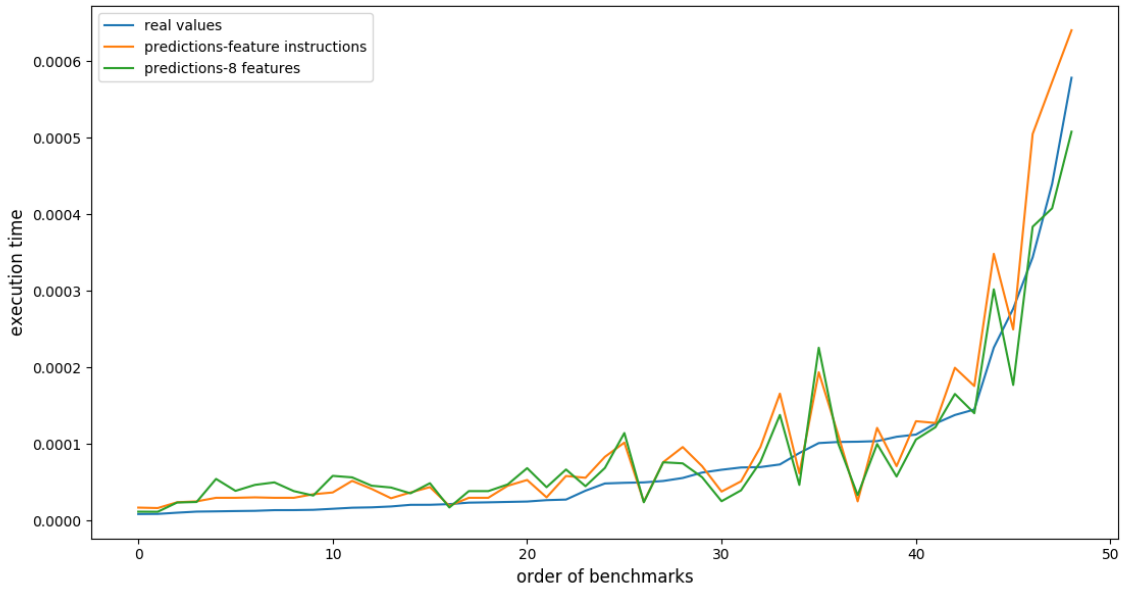
Από τα προγράμματα των benchmarks έχει απορριφθεί μόνο ένα επειδή ο αριθμός εντολών του στο σώμα επαναλήψεων είναι πολύ μεγάλος και δεν κλιμακώνεται καλά με τις υπόλοιπες τιμές. Ο μικρότερος αριθμός εντολών σε πρόγραμμα είναι 50 φορές μικρότερος από το μεγαλύτερο και αυτό είναι πολύ καλύτερο από την προηγούμενη περίπτωση που ο μικρότερος αριθμός μπορεί να ήταν και δεκάδες χιλιάδες φορές μικρότερος από τον μεγαλύτερο.

Η έξοδος που αναζητείται είναι ο χρόνος εκτέλεσης. Ο χρόνος που χρειάζεται για να γίνει η διαδικασία του clustering είναι σημαντικά μεγαλύτερος από την απλή διαδικασία πρόβλεψης.

LinearRegression() : Παρατηρείται ότι η προσθήκη features οδηγεί τις προβλέψεις να παίρνουν αρκετά χαμηλότερες τιμές από τις πραγματικές, οπότε δεν προσφέρουν στην πρόβλεψη.

Ridge() : Αντίθετη περίπτωση είναι το Ridge που παρατηρείται τα χαρακτηριστικά να επηρεάζουν την πρόβλεψη χωρίς να μειώνουν τις τιμές που μπορεί να φτάσει.

Αυτό φαίνεται καλύτερα στο παρακάτω διάγραμμα που έχουν χρησιμοποιηθεί 8 features και 500 clusters:



Σχήμα 5.4 : Προβλέψεις σε σχέση με τις πραγματικές τιμές του χρόνου εκτέλεσης με χρήση του Ridge

Τα αποτελέσματα δεν διαφέρουν στην περίπτωση με cross validation.

SGDRegressor(): Αυτή η μέθοδος δεν μπορεί να κάνει προσέγγιση με feature μόνο τις εντολές, αλλά επηρεάζεται θετικά από επιπλέον features και προσεγγίζει περισσότερο τις πραγματικές τιμές. Ωστόσο, η προσέγγιση αυτή είναι πολύ μικρότερη από προσεγγίσεις σε άλλες περιστάσεις.

ElasticNet(): Όπως και στην περίπτωση που χρησιμοποιούνται όλες οι εντολές στο πρόγραμμα, το μοντέλο αυτό αδυνατεί να κάνει πρόβλεψη και εμφανίζει όλες τις προβλέψεις σε μία γραμμή. Ενδιαφέροντα είναι τα αποτελέσματα για την περίπτωση που χρησιμοποιείται cross validation, καθώς φαίνεται να πλησιάζει σε τιμή τις προβλέψεις που γίνονται με το Ridge regression.

Lars(): Αυτή η μέθοδος μπορεί να κάνει προβλέψεις για το χρόνο αλλά όχι για την ενέργεια και αυτό πιθανόν οφείλεται στο ότι η ενέργεια παίρνει μικρότερες τιμές από ότι ο χρόνος. Παρόμοια αποτελέσματα εμφανίζονται στην περίπτωση cross validation.

Lasso(): Ακολουθεί τη λογική του ElasticNet, δηλαδή δεν μπορεί να κάνει προβλέψεις με απλές παραμέτρους, αλλά μπορεί να κάνει όταν χρησιμοποιείται cross validation.

LassoLars(): Είναι συνδυασμός των χαρακτηριστικών των μεθόδων Lasso και Lars. Πιο συγκεκριμένα, όταν χρησιμοποιούνται απλές παράμετροι δεν πραγματοποιεί προβλέψεις για το χρόνο και την ενέργεια. Στην περίπτωση με παράμετρο το cross validation εμφανίζεται αποτέλεσμα μόνο για το χρόνο. Παρόμοια αποτελέσματα δίνει και η μέθοδος LassoLarsIC.

OrthogonalMatchingPursuit(): Αυτή η μέθοδος μπορεί να πραγματοποιήσει προβλέψεις που φαίνεται η επιρροή της προσθήκης των features, τόσο για το χρόνο όσο και για την ενέργεια. Στην περίπτωση του cross validation, δεν μπορεί να κάνει υπολογισμό για την περίπτωση με feature μόνο τις εντολές, οπότε δε μπορεί να γίνει σύγκριση.

BayesianRidge(): Μέθοδος που πραγματοποιεί προβλέψεις με παρόμοια μορφή όπως το Ridge regression.
Στις robust μεθόδους regression, δηλαδή το HuberRegressor, το RANSACRegressor και το TheilSenRegressor παρατηρείται μία μεγάλη διακύμανση στα αποτελέσματα των προβλέψεων χρησιμοποιώντας μερικά features. Από αυτές τις μεθόδους μόνο το HuberRegressor μπορεί να κάνει προβλέψεις που προσεγγίζουν κάπως στις πραγματικές τιμές. Στις άλλες δύο μεθόδους κάθε ξεχωριστή πρόβλεψη μπορεί να παίρνει τελείως διαφορετικές τιμές από κάποια άλλη, οπότε δε μπορεί να δώσουν ακριβές μοντέλο.

Αντίθετη είναι η περίπτωση του LinearSVR, που στην περίπτωση με feature όλες τις εντολές δεν μπορούσε να κάνει πρόβλεψη. Τα αποτελέσματα με χρήση μόνο των εντολών ανά επανάληψη αποκλίνουν περισσότερο σε σχέση με άλλες μεθόδους και όταν συμπεριλαμβάνονται τα υπόλοιπα χαρακτηριστικά η πρόβλεψη προσεγγίζει περισσότερο τις πραγματικές τιμές. Ωστόσο οι υπόλοιπες μέθοδοι που ανήκουν στη μονάδα sklearn.svm αδυνατούν να πραγματοποιήσουν προβλέψεις.

KneighborsRegressor(): Σε αυτήν την περίπτωση οι προβλέψεις φτάνουν μέχρι μία τιμή και σταματούν να προσεγγίζουν τις υψηλότερες τιμές.

MLPRegressor(): Αυτή η μέθοδος κάνει προβλέψεις αλλά τα αποτελέσματα είναι πολλές τάξεις μεγέθους μεγαλύτερα από τις πραγματικές τιμές και δεν ακολουθούν κάποια διάταξη,

DecisionTrees: Με το συγκεκριμένο σύνολο features δεν μπορούν να πραγματοποιήσουν προβλέψεις.

3.2.1 Συμπεράσματα

Όπως φαίνεται υπάρχουν αρκετές μέθοδοι που μπορούν να δώσουν προβλέψεις για τις εξόδους ανά επανάληψη και στις οποίες η επιρροή των επιπλέον features είναι σημαντική. Μπορεί να επιλεχθούν διαφορετικές μέθοδοι για την πρόβλεψη του χρόνου εκτέλεσης και της ενεργειακής κατανάλωσης , ανάλογα με το ποια μπορεί να δώσει καλύτερα αποτελέσματα. Όταν επιλεχθεί η κατάλληλη μέθοδος, το αποτέλεσμα της πρόβλεψης που αντιστοιχεί σε μία επανάληψη μπορεί να πολλαπλασιαστεί με τον αριθμό των επαναλήψεων για να υπολογιστεί η συνολική απόδοση στο συγκεκριμένο κομμάτι κώδικα. Αν και αυτή η προσέγγιση δεν είναι απόλυτα ακριβής μπορεί να δώσει πολύ καλύτερα αποτελέσματα από την μέθοδο προβλέψεων με feature τις συνολικές εντολές.

3.3 Προβλέψεις χωρίς εκτέλεση

Για τον υπολογισμό του συνολικού χρόνου και της ενέργειας που καταναλώθηκε χρειάζεται να είναι γνωστός ο αριθμός των εντολών του προγράμματος. Εκτός από κάποιες απλές περιπτώσεις που είναι εύκολο να υπολογιστεί πόσες φορές εκτελείται ένα σώμα επαναλήψεων for, ο υπολογισμός των εντολών είναι μία διαδικασία που απαιτεί την εκτέλεση του προγράμματος. Ωστόσο, σε πολλές περιστάσεις η εκτέλεση των προγραμμάτων βασίζεται σε συγκεκριμένα στοιχεία που θέτει ο προγραμματιστής οπότε οι επαναλήψεις που

διαρκεί μία εντολή for μπορεί να πάρουν συγκεκριμένες τιμές κατά τη διάρκεια εκτέλεσης. Αν οι επαναλήψεις καλύπτουν κάποιο συγκεκριμένο εύρος μπορεί ο χρήστης αξιοποιώντας ένα από τα μοντέλα πρόβλεψης να κάνει μία εκτίμηση των εξόδων χωρίς να εκτελέσει το πρόγραμμα.

Για αυτό μπορεί να χρησιμοποιηθεί ένα script που δίνει μία εκτίμηση όλων των εντολών σε κάθε block επαναλήψεων ή συναρτήσεων του προγράμματος. Σε ένα πρόγραμμα τον περισσότερο κώδικα τον καταλαμβάνουν συναρτήσεις και τους περισσότερους πόρους τους καταναλώνουν οι πολλές επαναλήψεις κώδικα. Με προηγούμενες διαδικασίες υπολογίζονται τα χαρακτηριστικά των συναρτήσεων και επαναλήψεων με τη βοήθεια του IACA. Μέσα σε κάθε σώμα επαναλήψεων μπορεί να υπάρχουν άλλα εμφωλευμένα σώματα και κλήσεις συναρτήσεων, ενώ μέσα σε συναρτήσεις μπορεί να υπάρχουν επαναλήψεις.

Στην περίπτωση των συναρτήσεων στις εντολές του κυρίου σώματός τους συμπεριλαμβάνονται οι εντολές σε block επαναλήψεων και οι εντολές από κλήσεις άλλων συναρτήσεων. Στην περίπτωση των block επαναλήψεων δίνεται ένας συγκεκριμένος αριθμός επαναλήψεων από τον χρήστη για να υπολογιστούν οι εντολές.

Οι επαναλήψεις είναι ένα μέγεθος που δεν είναι σταθερό και αλλάζει ανάλογα με το πρόγραμμα και τον προγραμματιστή. Για παράδειγμα μία εντολή while μπορεί να εκτελείται για διαφορετικό αριθμό επαναλήψεων ανάλογα με κάποια συνθήκη ή από κάποια μεταβλητή που χρησιμοποιείται για έλεγχο και προκύπτει τυχαία. Ένας προγραμματιστής μπορεί, όμως, να εκτιμήσει το εύρος των επαναλήψεων που μπορεί να εκτελεστεί το σώμα, και παίρνοντας τα άκρα να παρατηρήσει πόσο μπορεί να διαρκέσει. Για να ρυθμίζονται οι επαναλήψεις διαφόρων τύπων αντικαθίστανται με τύπους for στους οποίους μπορούν εύκολα να οριστεί ο αριθμός επαναλήψεων.

Έχοντας αναλυτικά το πώς κατανέμονται οι εντολές στον κώδικα , ο χρήστης μπορεί να κάνει πρόβλεψη σε κάθε κομμάτι χρησιμοποιώντας το μοντέλο πρόβλεψης. Χρησιμοποιεί τις εντολές ανά επανάληψη και στο αποτέλεσμα που προκύπτει πολλαπλασιάζει με τον αριθμό εντολών. Αυτή η διαδικασία γίνεται για κάθε σώμα επαναλήψεων και μετά αθροίζοντάς τα προκύπτουν οι συνολικές έξοδοι.

3.4 Επιλογή τελικού μοντέλου εκτίμησης χρόνου και ενέργειας

Για την κατασκευή μοντέλου πρόβλεψης θα πρέπει να επιλεχθούν κάποιες μέθοδοι οι οποίες μπορούν να πραγματοποιήσουν προβλέψεις και υπάρχει σαφής επιρροή σε αυτές ανάλογα με τα features που χρησιμοποιούνται. Οι μέθοδοι που ακολουθούν τα παραπάνω χαρακτηριστικά είναι οι ακόλουθες:

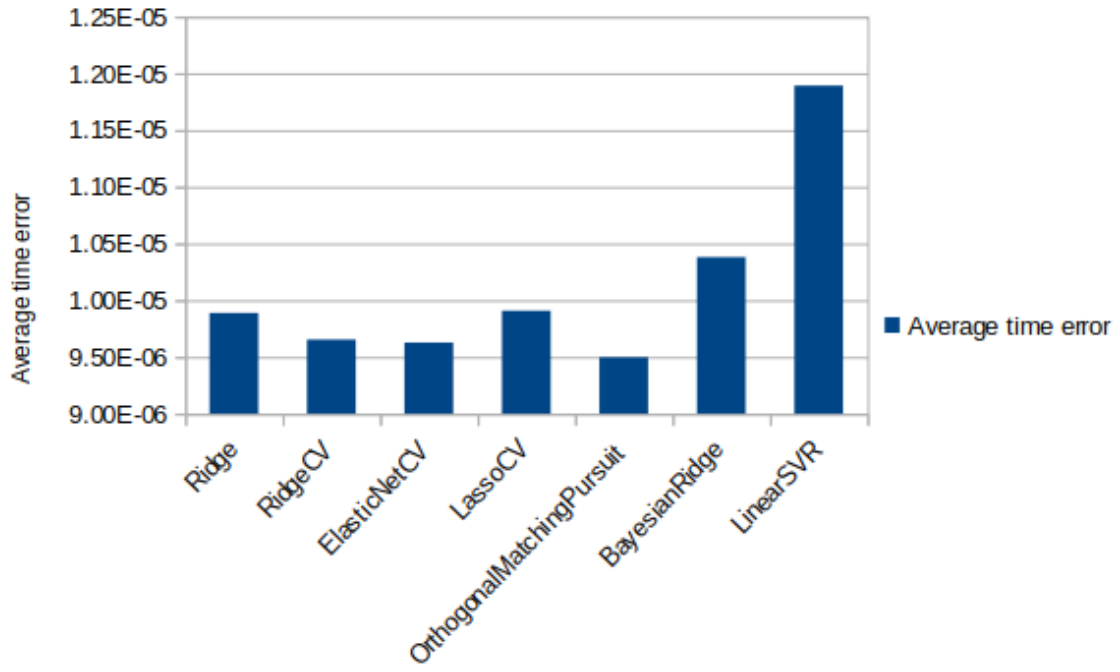
- **Ridge**
- **RidgeCV**
- **ElasticNetCV**
- **LassoCV**
- **OrthogonalMatchingPursuit**
- **BayesianRidge**
- **LinearSVR**

Η επιλογή θα γίνει με βάση το πόσο ακριβείς είναι οι προβλέψεις σύμφωνα με κάποιες μετρικές. Αυτές οι μετρικές είναι το μέσο απόλυτο σφάλμα και το r2_score. Ως test σύνολο θα χρησιμοποιηθεί ένα υποσύνολο από τα συνθετικά προγράμματα. Μπορεί να επιλεχθούν

διαφορετικές μέθοδοι για τον χρόνο εκτέλεσης και την ενεργειακή κατανάλωση. Τέλος, στο επόμενο κεφάλαιο θα δειχθεί η ακρίβεια των μοντέλων που χτίστηκαν και επιλέχθηκαν με βάση το σύνολο συνθετικών προγραμμάτων, όταν αυτά χρησιμοποιηθούν για την εκτίμηση του χρόνου και της ενέργειας σε ευρέως χρησιμοποιούμενα benchmarks.

Με τις εναλλακτικές μεθόδους θα γίνουν προβλέψεις πάνω στο test σύνολο για ένα σύνολο από clusters. Η τιμή των clusters ξεκινάει από 100 και συνεχίζει να παίρνει τιμές ανά 100 μέχρι να φτάσουν τα 1000 clusters. Σε κάθε μέθοδο για κάθε τιμή clusters υπολογίζεται το μέσο απόλυτο σφάλμα και το r2_score. Μετά βρίσκεται το μέσο απόλυτο σφάλμα της μεθόδου αθροίζοντας τα μέσα απόλυτα σφάλματα για κάθε τιμή clusters και διαιρώντας με τον αριθμό των διαφορετικών τιμών. Από τις μεθόδους επιλέγεται εκείνη με το ελάχιστο μέσο απόλυτο σφάλμα. Προφανώς πρέπει οι τιμές του r2_score να είναι υψηλές για να γίνει δεκτή μία μέθοδος.

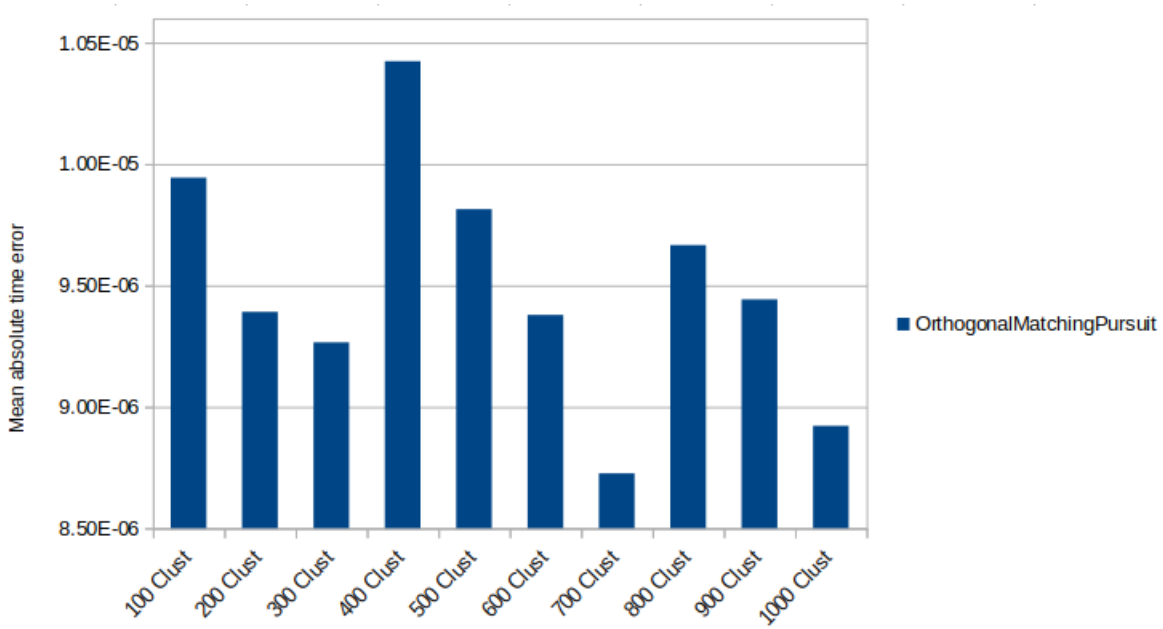
Εξετάζοντας τις μεθόδους και παίρνοντας το μέσο απόλυτο σφάλμα του χρόνου εκτέλεσης από όλες τιμές clusters προκύπτει το ακόλουθο σχήμα:



Σχήμα 5.5 : Το μέσο απόλυτο σφάλμα πρόβλεψης του χρόνου εκτέλεσης για τις διαφορετικές μεθόδους

Πολλά από τα σφάλματα έχουν κοντινές τιμές, αλλά το πιο μικρό είναι το σφάλμα της μεθόδου OrthogonalMatchingPursuit, οπότε αυτή επιλέγεται.

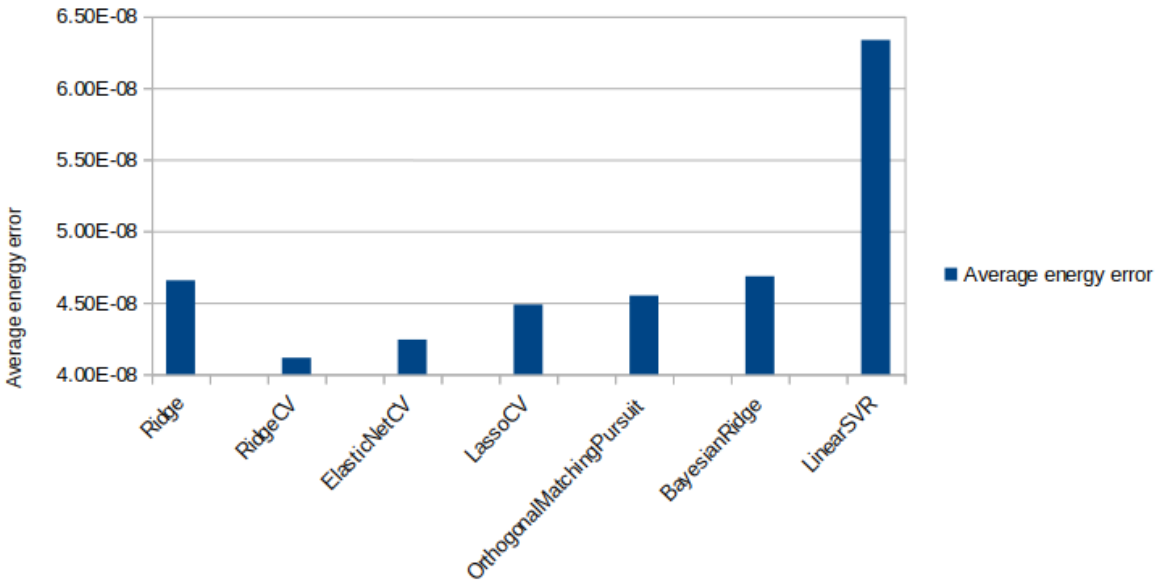
Από αυτή τη μέθοδο εξετάζεται για ποιο αριθμό clusters το σφάλμα είναι μικρότερο.



Σχήμα 5.6 : Τα μέσα απόλυτα σφάλματα των προβλέψεων του χρόνου εκτέλεσης για διαφορετικές τιμές clusters

Όπως φαίνεται για αριθμό 700 clusters εντοπίζεται το μικρότερο σφάλμα. Επομένως, για τις προβλέψεις του χρόνου εκτέλεσης θα χρησιμοποιηθεί η μέθοδος OrthogonalMatchingPursuit με 700 clusters.

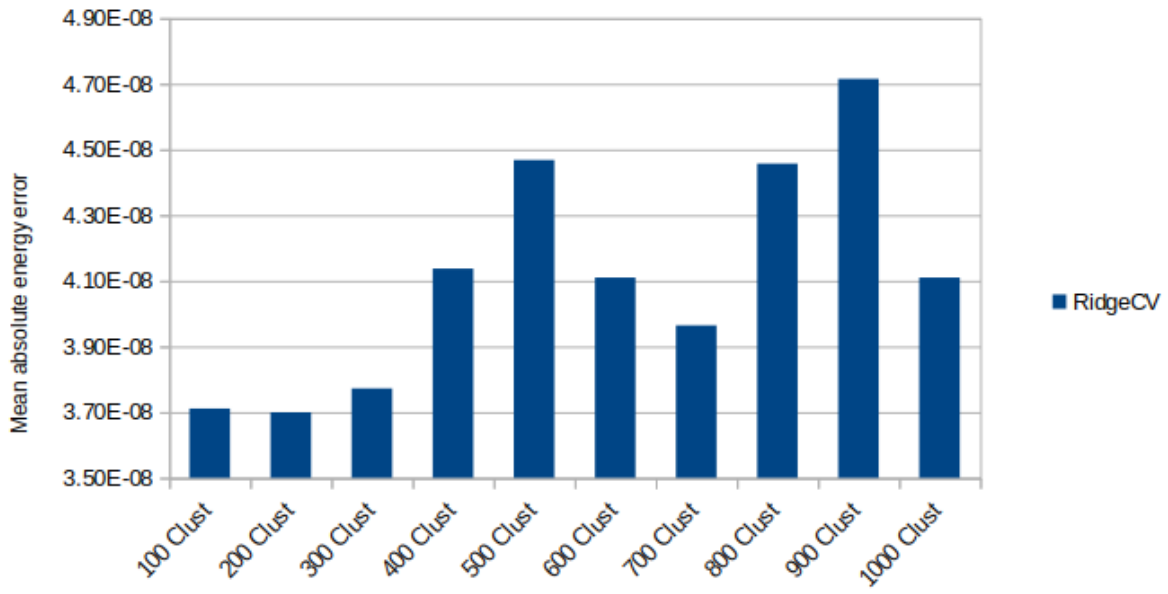
Η ίδια τεχνική χρησιμοποιείται για την εύρεση της καλύτερης μεθόδου για την πρόβλεψη ενέργειας.



Σχήμα 5.7 : Το μέσο απόλυτο σφάλμα πρόβλεψης της ενεργειακής κατανάλωσης για τις διαφορετικές μεθόδους

Το πιο μικρό σφάλμα παρατηρείται στην περίπτωση του RidgeCV, οπότε αυτή επιλέγεται.

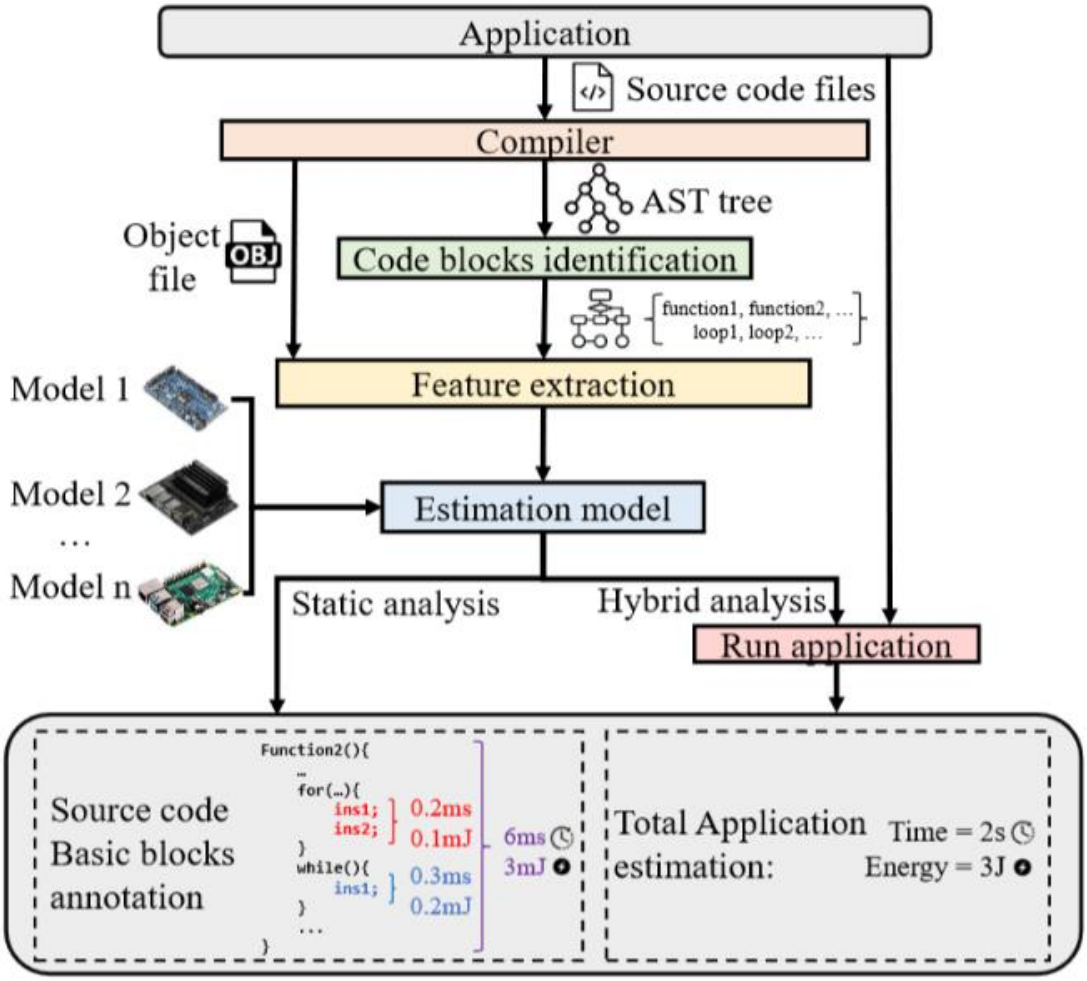
Από αυτή τη μέθοδο εξετάζεται για ποιο αριθμό clusters, το σφάλμα παίρνει τη μικρότερη τιμή.



Σχήμα 5.8: Τα μέσα απόλυτα σφάλματα προβλέψεων της ενεργειακής κατανάλωσης για διαφορετικές τιμές clusters

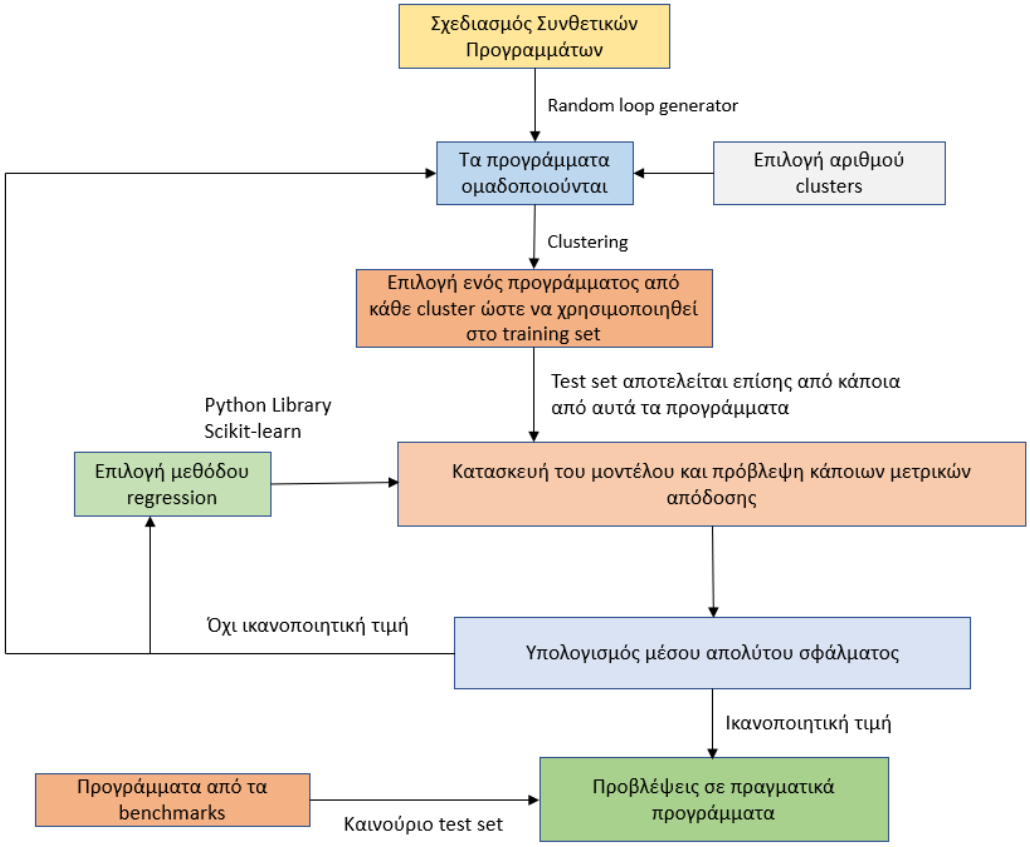
Το μικρότερο σφάλμα εντοπίζεται για 200 clusters.
Επομένως, για τις προβλέψεις της ενεργειακής κατανάλωσης χρησιμοποιείται η μέθοδος RidgeCV με 200 clusters.

Συνολικά η μεθοδολογία που πραγματοποιήθηκε σε αυτό το κεφάλαιο συνοψίζεται με το παρακάτω σχήμα.



Σχήμα 5.9 : Μεθοδολογία εκτίμησης της απόδοσης με στατική ανάλυση

Πιο συγκεκριμένα, η δημιουργία του μοντέλου εκτίμησης συνοψίζεται με το ακόλουθο σχήμα.



Σχήμα 5.10 : Μεθοδολογία κατασκευής μοντέλου πρόβλεψης

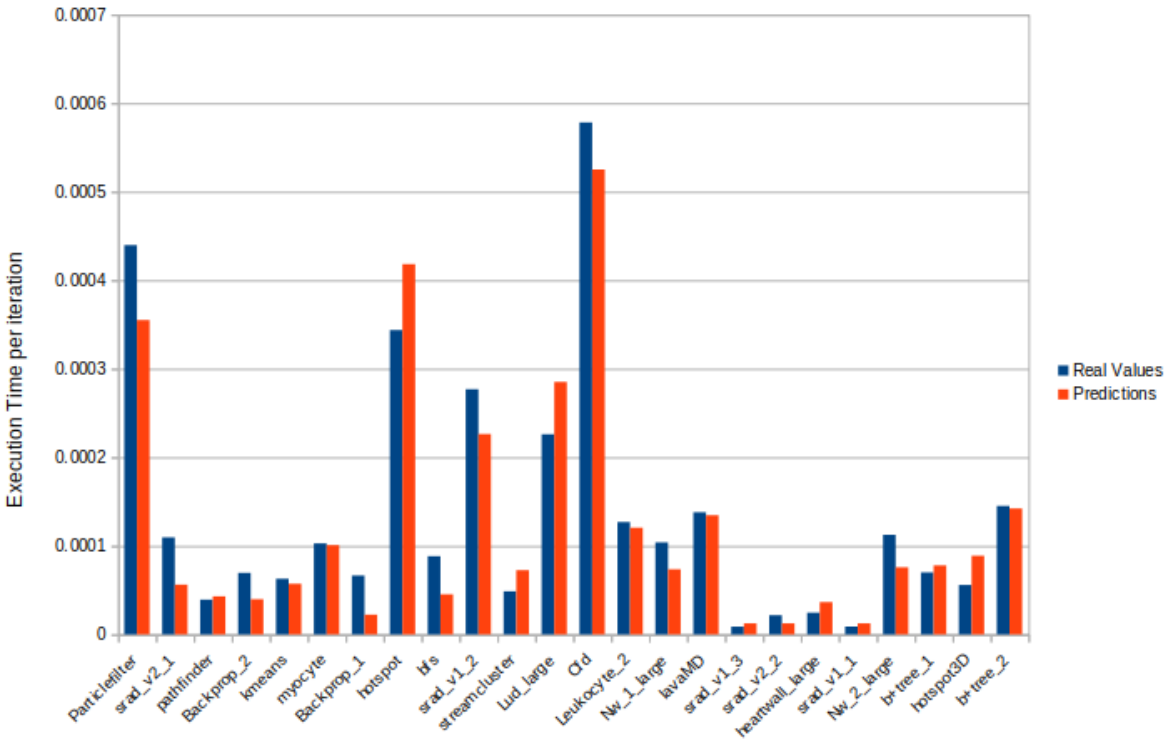
Κεφάλαιο 6

Πειραματικά Αποτελέσματα

1. Μελέτη σε περιβάλλον Nvidia Tegra

Με τη χρήση των μεθόδων που επιλέχθηκαν στην προηγούμενη ενότητα θα γίνουν προβλέψεις πάνω στα προγράμματα των benchmarks. Από τα benchmarks θα επιλεχθούν τα προγράμματα από το rodinia καθώς έχουν μεγαλύτερες διαφορές μεταξύ τους. Οι προβλέψεις θα πραγματοποιηθούν τόσο για το χρόνο όσο και για την ενέργεια πάνω στην πλακέτα Nvidia Tegra TX1. Οι πρώτες προβλέψεις πραγματοποιούνται όταν στα features περιλαμβάνονται οι εντολές ανά επανάληψη και οι έξοδοι αφορούν συγκεκριμένα κομμάτια κώδικα. Οι δεύτερες προβλέψεις προκύπτουν από τις πρώτες πολλαπλασιάζοντας με τον αριθμό επαναλήψεων για να βρεθεί η συνολική απόδοση σε αυτό το κομμάτι κώδικα. Οι τρίτες προβλέψεις προκύπτουν από συνδυασμό άλλων προβλέψεων για να βρεθεί η συνολική απόδοση σε όλο το πρόγραμμα.

Στο παρακάτω ραβδόγραμμα φαίνονται οι διαφορές μεταξύ των πραγματικών τιμών του χρόνου εκτέλεσης και των προβλεπόμενων τιμών.

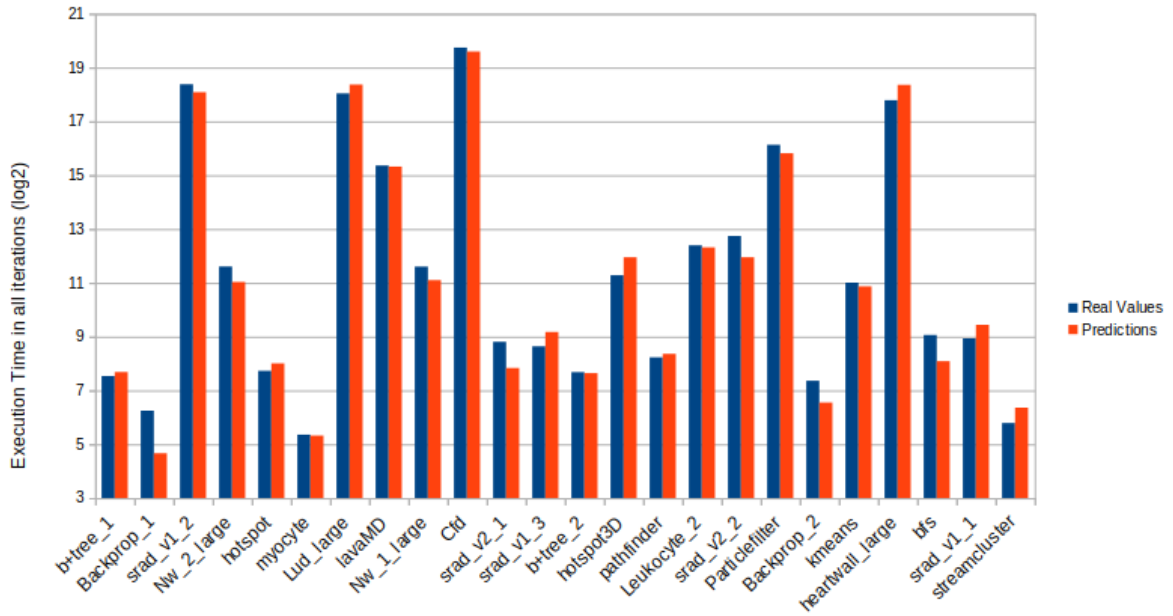


Σχήμα 6.1 : Ραβδογράμματα των προβλέψεων για τον χρόνο εκτέλεσης στα βασικά blocks των προγραμμάτων πάνω στην πλακέτα Nvidia Tegra TX1.

Στα περισσότερα προγράμματα οι προβλέψεις φαίνονται ικανοποιητικές. Εξαίρεση αποτελούν κάποια προγράμματα όπως το backprop, στα οποία παρατηρούνται πολλά cache και memory misses, οπότε το μοντέλο δεν δουλεύει ιδανικά. Αυτό γίνεται επειδή το μοντέλο χτίστηκε με βάση την ιδανική περίπτωση μηδενικών cache misses καθώς βασίζεται σε στατική ανάλυση.

Ωστόσο το r2_score διατηρεί υψηλές τιμές που φτάνουν το μέγεθος του 0.926.

Για να γίνουν προβλέψεις πάνω στον πραγματικό κώδικα πρέπει να πολλαπλασιαστούν οι παραπάνω προβλέψεις με τον αριθμό επαναλήψεων που αντιστοιχούν στο συγκεκριμένο πρόγραμμα. Επειδή οι τιμές που μπορούν να πάρουν οι έξοδοι έχουν μεγάλη ποικιλία και μπορεί να διαφέρουν για πολλές τάξεις μεγέθους, εμφανίζεται ο λογάριθμος τους με βάση το 2. Το διάγραμμα που προκύπτει είναι το ακόλουθο.



Σχήμα 6.2 : Ραβδογράμματα των προβλέψεων για τον χρόνο εκτέλεσης στα σώματα επαναλήψεων των προγραμμάτων πάνω στην πλακέτα Nvidia Tegra TX1.

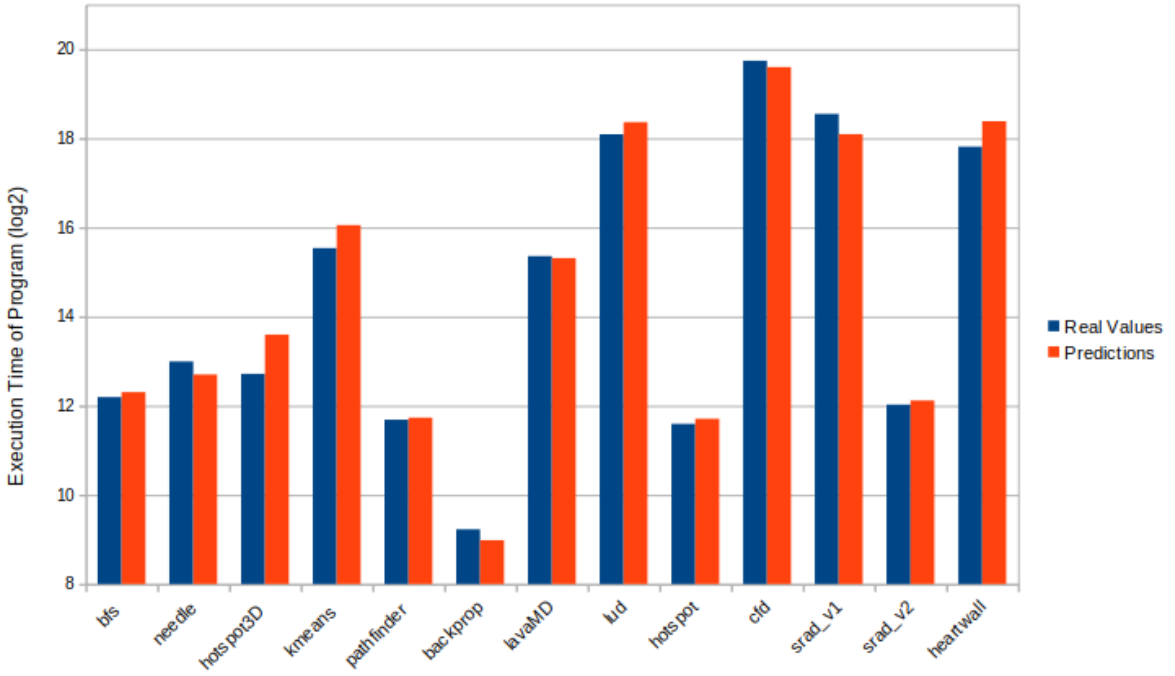
Όπως παρατηρείται στα διαγράμματα δεν υπάρχει κάποιο πρόγραμμα που η πρόβλεψη απέχει πολύ σε τιμή από την πραγματική έξοδο.

Αυτό γίνεται ιδιαίτερα αισθητό στην περίπτωση του r2_score που φτάνει την τιμή 0.9705.

Τα κομμάτια κώδικα που γίνονται προβλέψεις αποτελούν σώματα επαναλήψεων , γιατί οι επαναλήψεις καταναλώνουν το σημαντικότερο χρόνο σε ένα πρόγραμμα. Αν εκτιμηθούν οι χρόνοι εκτέλεσης για κάθε σώμα επαναλήψεων και για τις συναρτήσεις που τα καλούν, μπορεί να γίνει εκτίμηση για τον συνολικό χρόνο που καταναλώνει το πρόγραμμα.

Με τα εργαλεία που έχουν χρησιμοποιηθεί και τα script που έχουν κατασκευασθεί μπορεί να βρεθούν όλες οι επαναλήψεις και οι συναρτήσεις σε ένα πρόγραμμα και με τον υπολογισμό των εντολών τους να γίνει πρόβλεψη του χρόνου για την καθεμία. Για το συνολικό πρόγραμμα υπολογίζεται ο χρόνος και συγκρίνεται με το άθροισμα των παραπάνω προβλέψεων.

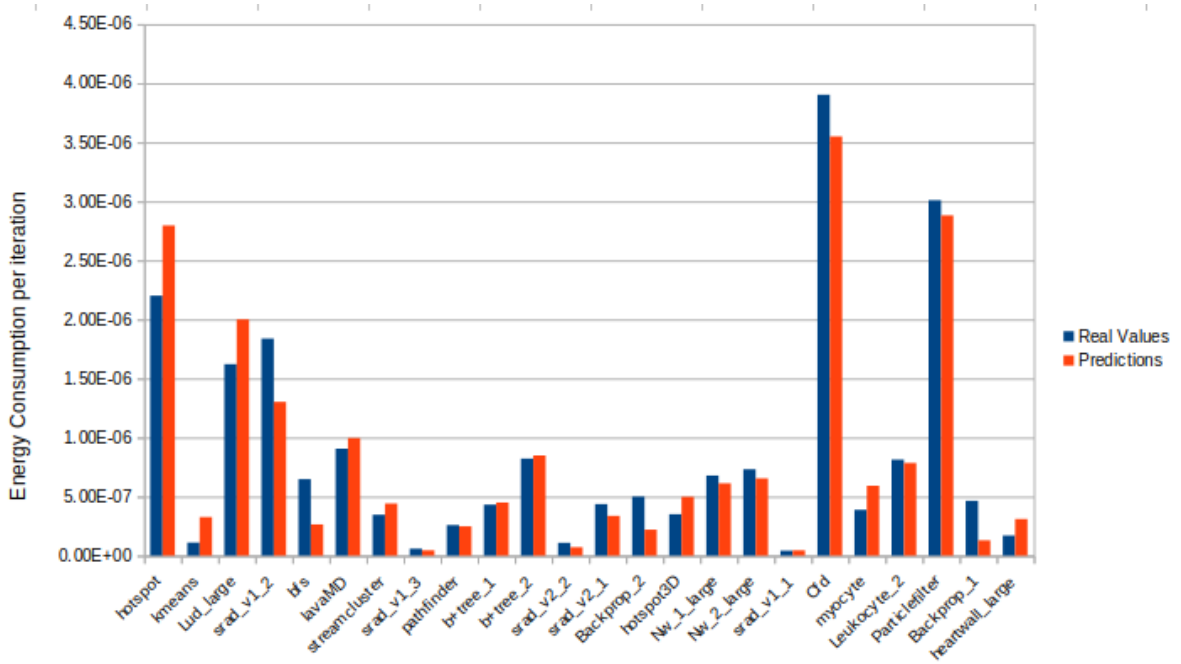
Τα αποτελέσματα εμφανίζονται με μορφή λογαρίθμου για να φαίνονται στο διάγραμμα:



Σχήμα 6.3 : Ραβδογράμματα που αντιστοιχούν στις προβλέψεις για τον χρόνο εκτέλεσης σε ολόκληρα τα προγράμματα πάνω στην πλακέτα Nvidia Tegra TX1.

Όπως φαίνεται από τα διαγράμματα οι προβλέψεις προσεγγίζουν αρκετά τις πραγματικές τιμές.

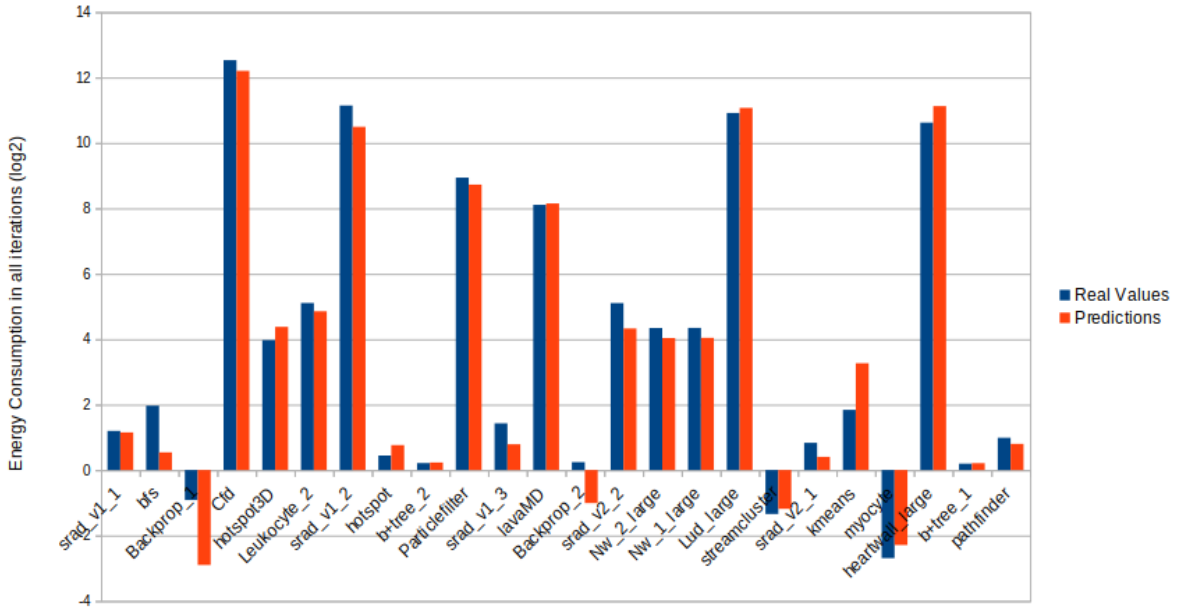
Η ίδια διαδικασία μπορεί να χρησιμοποιηθεί για την περίπτωση της ενέργειας. Τότε θα χρησιμοποιηθεί ένα διαφορετικό μοντέλο, το RidgeCV.



Σχήμα 6.4 : Ραβδογράμματα των προβλέψεων για την ενεργειακή κατανάλωση στα βασικά blocks των προγραμμάτων πάνω στην πλακέτα Nvidia Tegra TX1.

Πέρα από τις περιπτώσεις που μπορεί να υπάρχουν πολλά memory misses, όπως στην περίπτωση του backprop, παρατηρείται να γίνονται αρκετά καλές προσεγγίσεις.

Ωστόσο η μικρότερη ακρίβεια στις προσεγγίσεις φαίνεται στο r2_score που μπορεί να φτάσει χαμηλότερες τιμές της τάξης 0.9064 σε σχέση με την περίπτωση της πρόβλεψης του χρόνου. Για την περίπτωση που περιλαμβάνονται οι συνολικές επαναλήψεις τα ραβδογράμματα παίρνουν την παρακάτω μορφή, όπου οι τιμές αποτελούν λογάριθμο με βάση το 2.



Σχήμα 6.5 : Ραβδογράμματα των προβλέψεων για την ενεργειακή κατανάλωση στα σώματα επαναλήψεων των προγραμμάτων πάνω στην πλακέτα Nvidia Tegra TX1.

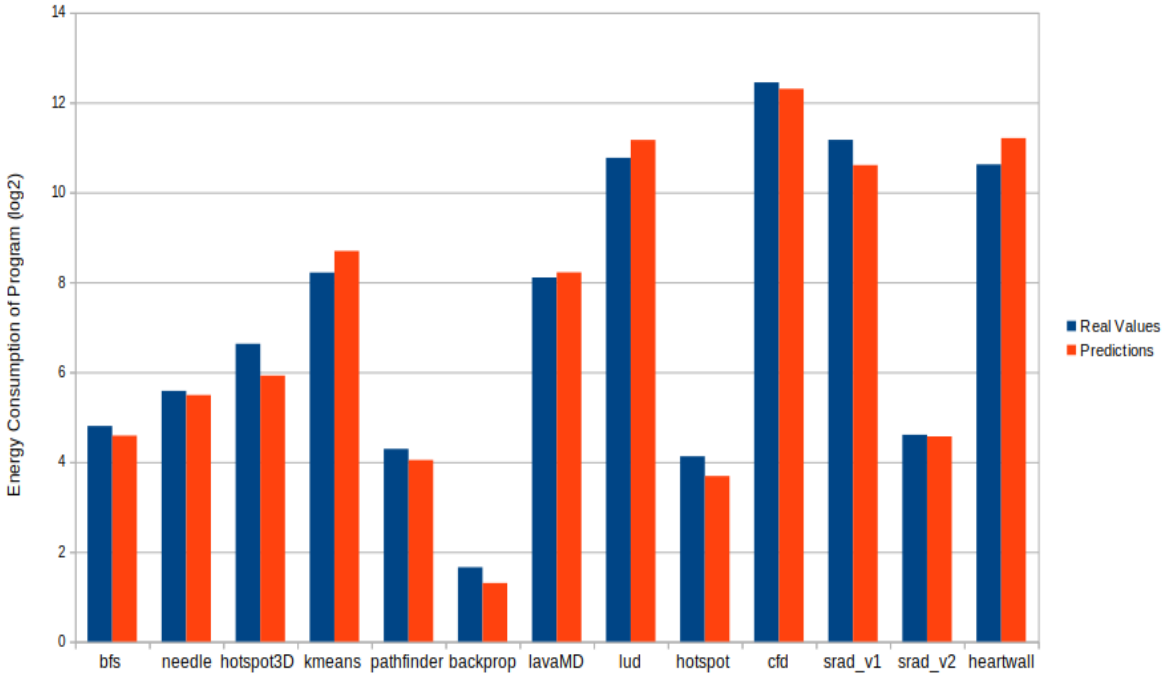
Σε κάποιες περιπτώσεις τα διαγράμματα παίρνουν αρνητικές τιμές και αυτό γίνεται επειδή οι τιμές της ενέργειας είναι μικρότερες του 1.

Αυτό που παρατηρείται είναι ότι σε μερικά προγράμματα η πρόβλεψη δεν προσεγγίζει ιδιαίτερα τις πραγματικές τιμές και αυτό συμβαίνει για περισσότερα προγράμματα από την περίπτωση που γινόταν εκτίμηση του χρόνου εκτέλεσης. Τα περισσότερα από αυτά τα προγράμματα έχουν πολύ μικρή ενεργειακή κατανάλωση, όπως για παράδειγμα το backprop, οπότε βγαίνει το συμπέρασμα ότι η ενέργεια δεν μπορεί να προβλεφθεί με μεγάλη ακρίβεια για μικρές τιμές της.

Η ακρίβεια αυτής της πρόβλεψης είναι μικρότερη από την πρόβλεψη του χρόνου και αυτό φαίνεται από το r2_score που φτάνει τιμές της τάξης 0.9438. Αυτό πιθανόν οφείλεται στο ότι ο αισθητήρας δεν είναι τόσο ακριβής και ότι η ενέργεια ως μέγεθος εξαρτάται ακόμα περισσότερο από τα δεδομένα σε σχέση με το χρόνο.

Για την καλύτερη κατανόηση των αποτελεσμάτων γίνεται και πρόβλεψη πάνω στην ενεργειακή κατανάλωση ολόκληρων προγραμμάτων.

Τα αποτελέσματα εμφανίζονται με λογαριθμική μορφή:



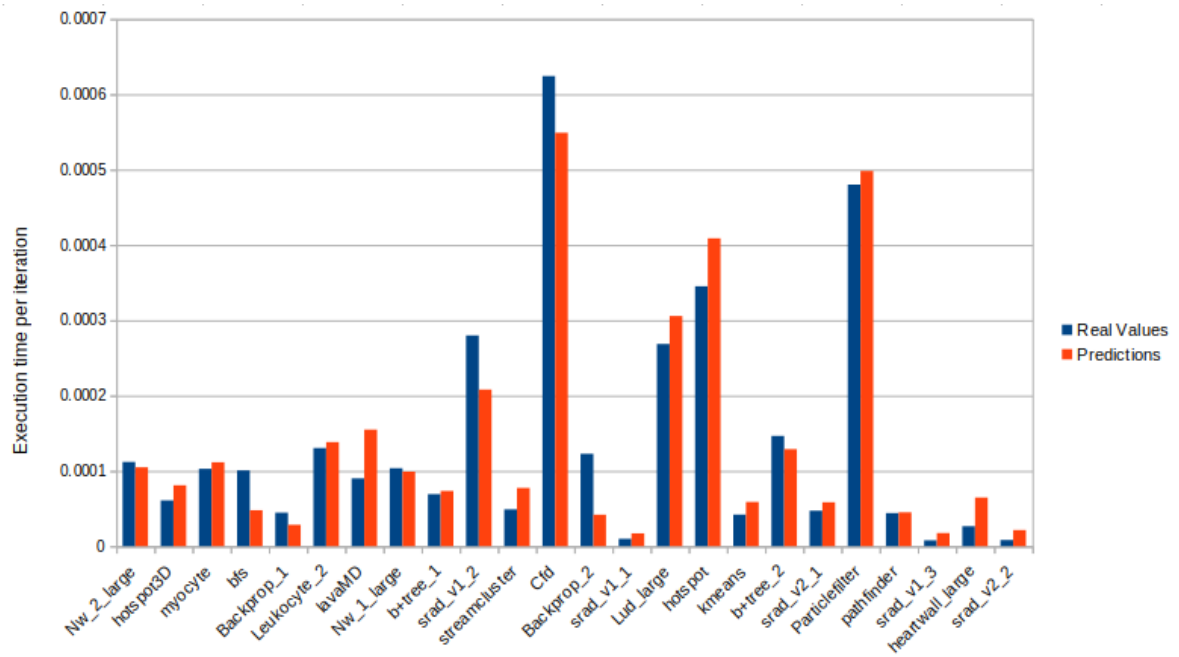
Σχήμα 6.6 : Ραβδογράμματα που αντιστοιχούν στις προβλέψεις για την ενεργειακή κατανάλωση σε ολόκληρα τα προγράμματα πάνω στην πλακέτα Nvidia Tegra TX1.

Συγκρίνοντας τις εικόνες 6.3 και 6.6 φαίνεται ότι η πρόβλεψη του χρόνου εκτέλεσης μπορεί να γίνει με μεγαλύτερη ακρίβεια από την πρόβλεψη της ενέργειας. Ωστόσο, η πρόβλεψη ενέργειας δεν αποκλίνει σημαντικά και σε αρκετές περιπτώσεις προσεγγίζει σε μεγάλο βαθμό τις πραγματικές τιμές.

2. Μελέτη σε περιβάλλον Raspberry Pi

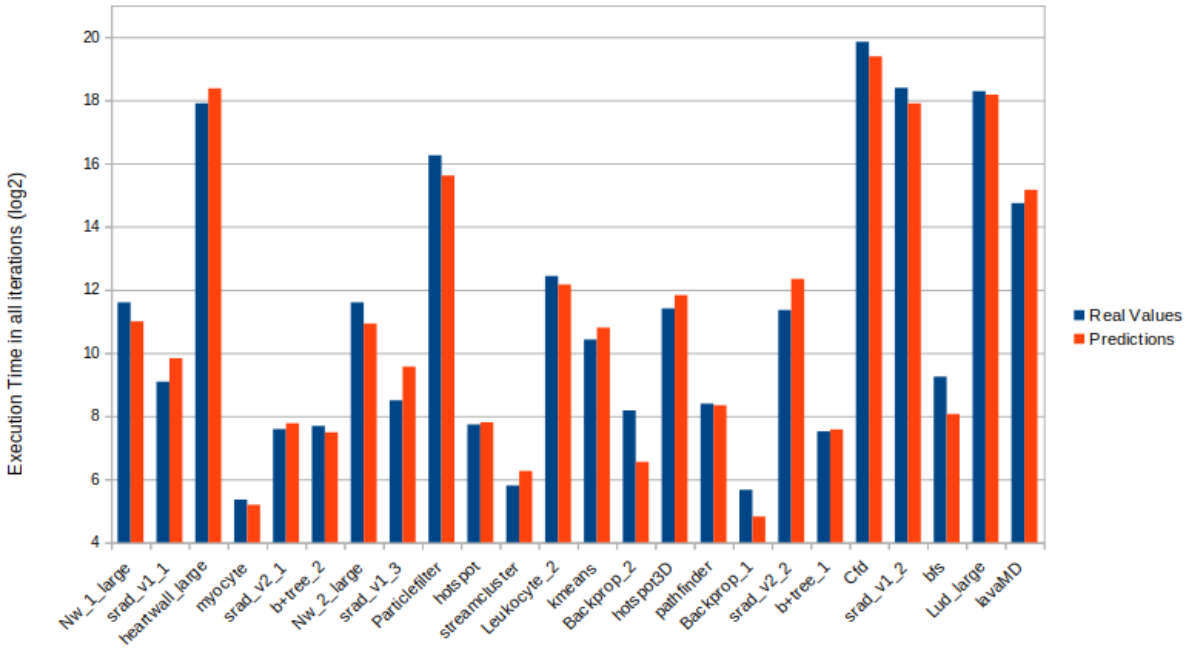
Οι προηγούμενες προβλέψεις έγιναν σε περιβάλλον συγκεκριμένης αρχιτεκτονικής και πλατφόρμας. Ωστόσο, το συγκεκριμένο μοντέλο πρόβλεψης μπορεί να κάνει εκτιμήσεις για την απόδοση των προγραμμάτων και σε διαφορετικά περιβάλλοντα. Η πλατφόρμα που επιλέχθηκε είναι μία πλακέτα Raspberry Pi τύπου 4 και οι μετρήσεις αφορούν τον επεξεργαστή ARM Cortex A72. Η μόνη μετρική απόδοσης που μπορεί να υπολογιστεί είναι ο χρόνος.

Ακολουθείται η ίδια μεθοδολογία που χρησιμοποιήθηκε και πριν. Τα συνθετικά προγράμματα εκτελούνται στο περιβάλλον της πλακέτας και συγκεντρώνονται οι χρόνοι εκτέλεσής τους. Τα προγράμματα των benchmarks , επίσης εκτελούνται στην πλακέτα και συγκεντρώνονται οι χρόνοι τους. Χρησιμοποιείται η μέθοδος πρόβλεψης και γίνεται διαχωρισμός των δεδομένων σε clusters. Οι προβλέψεις με feature τις εντολές ανά επανάληψη παρουσιάζονται στο παρακάτω διάγραμμα:



Σχήμα 6.7 : Ραβδογράμματα των προβλέψεων για τον χρόνο εκτέλεσης στα βασικά blocks των προγραμμάτων πάνω στην πλακέτα Raspberry Pi 4.

Σε γενικές γραμμές οι προβλέψεις για τους χρόνους εκτέλεσης δεν διαφέρουν πολύ σε σχέση με τις προβλέψεις στην άλλη πλακέτα. Οι περισσότερες εκτιμήσεις είναι αρκετά κοντά στις πραγματικές τιμές και μόνο σε λίγες περιπτώσεις δε γίνεται καλή προσέγγιση. Η τιμή του r2_score μπορεί να φτάσει τιμές μεγέθους 0.93 αλλά η τιμή του κυμαίνεται κυρίως σε 0.89.
Για την περίπτωση που γίνεται εκτίμηση πάνω στις συνολικές επαναλήψεις των blocks κώδικα το διάγραμμα προβλέψεων έχει την παρακάτω μορφή , όπου τα μεγέθη αναπαρίστανται με λογαριθμικές τιμές:

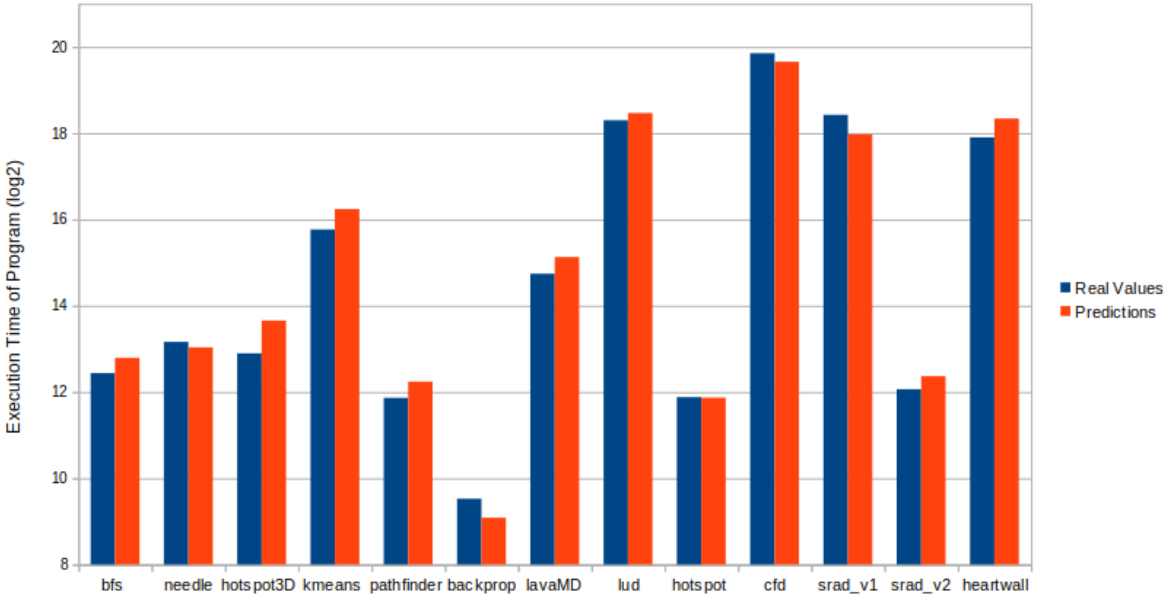


Σχήμα 6.8 : Ραβδογράμματα των προβλέψεων για τον χρόνο εκτέλεσης στα σώματα επαναλήψεων των προγραμμάτων πάνω στην πλακέτα Raspberry Pi 4.

Οι προβλέψεις είναι αρκετά καλές σε σχέση με τις πραγματικές τιμές , αλλά παρατηρείται μία μεγαλύτερη αύξηση στα σφάλματα σε σχέση με τις προβλέψεις στην άλλη πλακέτα. Αυτό γίνεται καλύτερα κατανοητό υπολογίζοντας το r2_score για αυτές τις προβλέψεις, το οποίο φτάνει στην καλύτερη περίπτωση σε τιμές 0.91 αλλά κυρίως κυμαίνεται σε τιμή 0.88.

Ένας πιθανός λόγος για αυτήν την μείωση ίσως είναι η υπερθέρμανση της πλακέτας από την εκτέλεση των συνθετικών προγραμμάτων. Αυτό θα οδηγούσε να υπολογίζονται σε μικρό βαθμό υψηλότεροι χρόνοι εκτέλεσης και να γίνεται μία υπερεκτίμηση των εξόδων. Ένα πιο γενικό συμπέρασμα είναι ότι το μοντέλο δεν λειτουργεί με την ίδια ακριβώς ακρίβεια σε διαφορετικές πλατφόρμες. Ωστόσο, αυτό παρατηρείται μόνο σε μερικά προγράμματα , ενώ στα υπόλοιπα η επίδραση είναι μηδαμινή.

Η επίδραση αυτού του παράγοντα δεν είναι αισθητή στην περίπτωση που υπολογίζεται ο χρόνος εκτέλεσης ολόκληρων των προγραμμάτων, όπως φαίνεται στο παρακάτω διάγραμμα, όπου οι έξοδοι παίρνουν λογαριθμικές τιμές:



Σχήμα 6.9 : Ραβδογράμματα που αντιστοιχούν στις προβλέψεις για τον χρόνο εκτέλεσης σε ολόκληρα τα προγράμματα πάνω στην πλακέτα Raspberry Pi 4

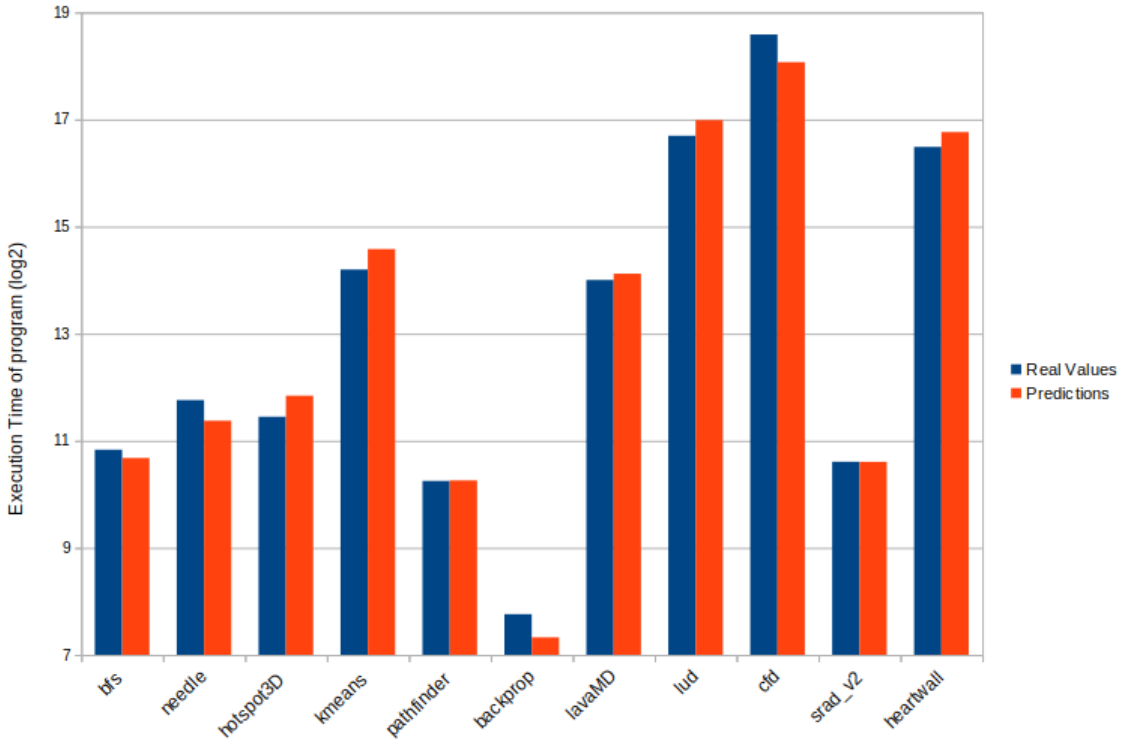
Στη συγκεκριμένη περίπτωση τα σφάλματα παίρνουν αισθητά πιο χαμηλές τιμές.

Είναι φανερό ότι το μοντέλο μπορεί να πραγματοποιήσει προβλέψεις με αρκετά καλή ακρίβεια και σε άλλες πλακέτες, ωστόσο δεν είναι ιδανικό καθώς οι τιμές των εξόδων διαφέρουν από πλακέτα σε πλακέτα και δεν προσαρμόζεται ιδανικά ώστε οι προβλέψεις να γίνονται με την ίδια ακρίβεια.

3. Μελέτη στο περιβάλλον του υπολογιστή

Ένας απλός τρόπος να γίνουν προβλέψεις σε περιβάλλον Intel είναι να γίνουν μετρήσεις στο περιβάλλον του υπολογιστή που κατασκευάστηκαν τα προγράμματα. Φυσικά η μέτρηση των χρόνων εκτέλεσης στο περιβάλλον δεν είναι απόλυτα ακριβής καθώς υπάρχει ένα σύνολο δραστηριοτήτων που αλλάζουν διαρκώς την απόδοσή του. Ωστόσο, δίνεται μία αρκετά καλή εικόνα για το αν γίνονται προβλέψεις σε περιβάλλον Intel. Το μοντέλο του υπολογιστή είναι Lenovo G50-70 20351 και διαθέτει επεξεργαστή Intel(R) Core(TM) i5-4210U.

Όπως και στις πλακέτες, πραγματοποιήθηκαν προβλέψεις πάνω στα βασικά block των εφαρμογών και προβλέψεις που έλαβαν υπόψη τους τον αριθμό επαναλήψεων που εκτελείται κάθε block. Αυτές οι προβλέψεις συνδυάστηκαν και έδωσαν ως αποτέλεσμα την πρόβλεψη για το συνολικό χρόνο εκτέλεσης ενός συνόλου προγραμμάτων. Οι προβλέψεις για ολόκληρα τα προγράμματα φαίνονται στο παρακάτω διάγραμμα:



Σχήμα 6.10 : Ραβδογράμματα που αντιστοιχούν στις προβλέψεις για τον χρόνο εκτέλεσης σε ολόκληρα τα προγράμματα στο περιβάλλον του υπολογιστή.

Από αυτό το διάγραμμα φαίνεται ότι το μοντέλο μπορεί να χρησιμοποιηθεί για προβλέψεις σε διαφορετικές αρχιτεκτονικές.

Κεφάλαιο 7

Επίλογος - Συμπεράσματα

Η απόδοση των προγραμμάτων είναι ένα ζήτημα που απασχολεί για δεκαετίες τους προγραμματιστές και τους κατασκευαστές επεξεργαστών. Ο υπολογισμός της απόδοσης με δυναμικές μεθόδους σπαταλά πολλούς υπολογιστικούς πόρους , οπότε γίνονται διαρκώς προσπάθειες για την βελτιστοποίησή του.

Στην συγκεκριμένη διπλωματική κατασκευάστηκε ένα μοντέλο πρόβλεψης της χρονικής και ενεργειακής κατανάλωσης σε συγκεκριμένα block κώδικα και σε διαφορετικές πλακέτες χρησιμοποιώντας στατική ανάλυση. Για την δημιουργία του κατασκευάστηκε ένα σύνολο συνθετικών προγραμμάτων και η αξιολόγηση του έγινε από μία ποικιλία από πραγματικά προγράμματα. Επιλέχθηκαν συγκεκριμένες μέθοδοι regression για την πρόβλεψη και έγινε ομαδοποίηση των συνθετικών προγραμμάτων με την μέθοδο του clustering. Τα αποτελέσματα που προέκυψαν είχαν ικανοποιητικές τιμές για τον χρόνο και την ενέργεια εκτέλεσης στα block κώδικα και χρησιμοποιήθηκαν για την εκτίμηση της συνολικής απόδοσης σε ολόκληρα τα προγράμματα της αξιολόγησης.

Σε αυτήν την διπλωματική προτείνεται μία γενικευμένη μεθοδολογία για την εκτίμηση του χρόνου εκτέλεσης και της ενεργειακής κατανάλωσης προγραμμάτων μέσω στατικής ανάλυσης σε διαφορετικές συσκευές και αρχιτεκτονικές. Σε σύγκριση με τη σχετική βιβλιογραφία δεν λαμβάνονται ως είσοδος πληροφορίες δυναμικής ανάλυσης του κώδικα και η προτεινόμενη τεχνική δεν περιορίζεται σε ένα συγκεκριμένο μικροελεγκτή. Αφού δίνονται τα πρώτα δείγματα ότι μια τέτοια προσέγγιση είναι εφικτή σε επόμενο βήμα στόχος είναι η εξέλιξη της τεχνικής ώστε να λαμβάνεται υπόψη πιο σύνθετη πληροφορία(όπως για παράδειγμα τη σειρά εκτέλεσης των εντολών), καθώς επίσης η αξιολόγηση σε πολύ μεγαλύτερο βαθμό συσκευών με πολυπλοκότερες/ετερογενείς αρχιτεκτονικές.

Βιβλιογραφία

[1] Valgrind Documentation
https://valgrind.org/docs/

[2] Wichmann B.A. et al. (March 1995). "*Industrial Perspective on Static Analysis*". **Software Engineering Journal: 69-75**

[3] Difference between Static and Dynamic Analysis
https://software.intel.com/en-us/inspector-user-guide-windows-dynamic-analysis-vs-static-analysis

[4] Vijay D'Silva; et al. (2008). "*A Survey of Automated Techniques for Formal Software Verification*". **Transactions On CAD**

[5] FDA (2010-09-08). "*Infusion Pump Software Safety Research at FDA*". **Food and Drug Administration**

[6]
Computer based safety systems - technical guidance for assessing software aspects of digital computer based protection systems, "*Computer based safety systems*"

[7] VDC Research (2012-02-01). "*Automated Defect Prevention for Embedded Software Quality*". **VDC Research**

[8] Bazzaz, Mostafa, Mohammad Salehi, and Alireza Ejlali. "*An accurate instruction-level energy estimation model and tool for embedded systems."* **"IEEE transactions on instrumentation and measurement"**

[9] Zheng, Xinnian, Lizy K. John, and Andreas Gerstlauer. "*Accurate phaselevel cross-platform power and performance estimation*." **Proceedings of the 53rd Annual Design Automation Conference. 2016.**

[10] Zheng, Xinnian, et al. "*Sampling-based binary-level cross-platform performance estimation*." **Design, Automation & Test in Europe Conference & Exhibition (2017)**

[11] Brandolese, Carlo.
"*Source-level estimation of energy consumption and execution time of embedded software*." **11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools. IEEE, 2008**

[12] Meng, Kewen, and Boyana Norris.
"*Mira: A framework for static performance analysis.*" **IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2017**

[13] Callou, Gustavo, et al. "*Energy consumption and execution time estimation of embedded system applications*." **Microprocessors and Microsystems 35.4 (2011): 426440.**

[14] Ferdinand, Christian, and Reinhold Heckmann. "*ait: Worst-case execution time prediction by static program analysis*."
Building the Information Society. Springer, Boston, MA, 2004. 377383.

[15] Li, Xiafeng, et al. "*Chronos: A timing analyzer for embedded software.*"
Science of Computer Programming 69.13 (2007): 5667.

[16] Intel architecture code analyzer Documentation,
https://software.intel.com/sites/default/files/managed/3d/23/intel-architecture-code-analyzer-3.0-users-guide.pdf

[17] LLVM Machine Code Analyzer Documentation
https://llvm.org/docs/CommandGuide/llvm-mca.html

[18] Clang 11 Documentation
https://clang.llvm.org/docs/index.html

[19] Profiling with clang
https://clang.llvm.org/docs/UsersManual.html

[20] Clang Static Analyzer
https://clang.llvm.org/docs/ClangStaticAnalyzer.html

[21] Interfaces to Clang
https://clang.llvm.org/docs/Tooling.html

[22] Clang AST
https://clang.llvm.org/docs/IntroductionToTheClangAST.html

[23] clang.cindex – Python library
https://github.com/llvm-mirror/clang/blob/master/bindings/python/clang/cindex.py

[24] Ordinary least squares published by A.M. Legendre
A.M. Legendre. "*Nouvelles méthodes pour la détermination des orbites des* comètes".
Firmin Ditot, Paris, 1805

[25] Linear Models
Hilary L. Seal (1967). "*The historical development of the Gauss linear model*".
Biometrika. 54 (1/2): 1–24

[26] Probability distributions
Lange, Kenneth L.; Little, Roderick J.A.; Taylor, Jeremy M.G. (1989). "Robust Statistical Modeling Using the t distribution".
Journal of the American Statistical Association. 84 (408): 881-896

[27] Ridge Regression
Swindel, Benee F. (1981). "*Geometry of Ridge Regression Illustrated*"
The American Statistician. 35 (1): 12-15

[28] Lasso Regression
Tibshirani, Robert (1996). "*Regression Shrinkage and Selection via the Lasso*".
Journal of the Royal Statistical Society, Series B. 58(1): 267-288

[29] Least absolute deviation regression
Narula, Subhash C.; Wellington, John F. (1982). "*The Minimum Sum of Absolute Erros Regression: A State of the Art Survey*".
International Statistical Review. 50 (3): 317-326

[30] Principal Component Regression
Hawkins, Douglas M. (1973). "*On the investigation of Alternative Regressions by Principal Component Analysis*".
Journal of the Royal Statistical Society, Series C. 22 (3): 275-286

[31] R^2 score
Glantz, Stanton A.; Slinker, B. K. (1990). *Primer of Applied Regression and Analysis of Variance.*
McGraw-Hill

[32] F-distribution
Mood, Alexander; Franklin A. Graybill; Dyane C. Boes (1974). *Introduction to the Theory of Statistics (Third Edition, pp. 246-249).*
McGraw-Hill

[33] Null hypothesis
Everitt, Brian (1998). *The Cambridge Dictionary of Statistics*.
Cambridge, UK New York: Cambridge University Press

[34] Student-t distribution
Helmert FR (1875). "*Über die Berechnung des wahrscheinlichen Fehlers aus einer endlichen Anzahl wahrer Beobachtungsfehler*".
Z. Math. U. Physik. 20: 300–3

[35] API Reference – class and function reference of scikit-learn
https://scikit-learn.org/stable/modules/classes.html

[36] scikit-learn Linear Models
https://scikit-learn.org/stable/modules/linear_model.html#linear-model

[37] scikit-learn Nearest Neighbors
https://scikit-learn.org/stable/modules/neighbors.html#neighbors

[38] scikit-learn Neural network models (supervised)
https://scikit-learn.org/stable/modules/neural_networks_supervised.html#neural-networks-supervised

[39] scikit-learn Decision Trees
https://scikit-learn.org/stable/modules/tree.html#tree

[40] scikit-learn Support Vector Machines
https://scikit-learn.org/stable/modules/svm.html#svm

[41] scikit-learn Ensemble methods
https://scikit-learn.org/stable/modules/ensemble.html#ensemble

[42] scikit-learn Metrics and scoring
https://scikit-learn.org/stable/modules/model_evaluation.html#model-evaluation

[43] Polybench benchmark
https://web.cse.ohio-state.edu/~pouchet.2/software/polybench/

[44] rodinia_3.1 benchmark
http://lava.cs.virginia.edu/Rodinia/download_links.htm

[45] gcc -o/-O option flags
https://www.rapidtables.com/code/linux/gcc/gcc-o.html

[46] Pin documentation
https://software.intel.com/sites/landingpage/pintool/docs/71313/Pin/html/

[47] Pinplay logger -Program Record/Replay Toolkit
https://software.intel.com/en-us/articles/program-recordreplay-toolkit

[48] scikit-learn Clustering
https://scikit-learn.org/stable/modules/clustering.html#clustering