Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επιστήμη Υπολογιστών

Ανάλυση της Επίδοσης των Αλγορίθμων Χρωματισμού Γράφων σε Πολυπύρηνα Συστήματα

Διπλωματικη Εργασια

του

ΑΘΑΝΑΣΙΟΥ Β. ΠΕΠΠΑ

Επιβλέπων: Γεώργιος Γκούμας
Αναπληρωτής Καθηγητής

Αθήνα, Σεπτέμβριος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επιστήμη Υπολογιστών

Ανάλυση της Επίδοσης των Αλγορίθμων Χρωματισμού Γράφων σε Πολυπύρηνα Συστήματα

Διπλωματικη Εργασια

του

ΑΘΑΝΑΣΙΟΥ Β. ΠΕΠΠΑ

Επιβλέπων: Γεώργιος Γκούμας
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Νοεμβρίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Γκούμας	Νεκτάριος Κοζύρης	Διονύσιος Πνευματικάτος
Αναπληρωτής Καθηγητής	Καθηγητής	Καθηγητής

Αθήνα, Σεπτέμβριος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επιστήμη Υπολογιστών

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Αθανάσιος Πέππας

4 Νοεμβρίου 2022

Περίληψη

Ο χρωματισμός γράφων απόστασης-κ είναι ένα πρόβλημα που χρωματίζει τις κορυφές ενός γράφου έτσι ώστε να μην υπάρχουν κορυφές με το ίδιο χρώμα σε απόσταση κ βημάτων. Κατά συνέπεια, ο χρωματισμός απόστασης 1, χρωματίζει τις κορυφές ενός γράφου έτσι ώστε καμία γειτονική κορυφή να μην έχει το ίδιο χρώμα. Ο χρωματισμός γράφων χρησιμοποιείται σε πολλές εφαρμογές λόγω της ικανότητάς του να αντιμετωπίζει εξαρτήσεις δεδομένων, επεξεργάζοντας παράλληλα κορυφές ίδιου χρώματος. Η επεξεργασία των δεδομένων με αυτόν τον τρόπο, οδηγεί μας αποδεσμεύει από την ανάγκη συγχρονισμού, γεγονός που μπορεί να οδηγήσει σε μικρότερους χρόνους εκτέλεσης.

Η παρούσα διπλωματική εργασία έχει ως στόχο την ανάλυση και βελτιστοποίηση των διαφόρων αλγορίθμων που προτείνονται στη βιβλιογραφία για το πρόβλημα του χρωματισμού γράφων, καθώς και τη χρήση τους σε πραγματικές εφαρμογές, όπως το PageRank και το Community Detection.

Λέξεις Κλειδιά

Γράφος, Χρωματισμός Γραφου, Ισορροπημένος Χρωματισμός, Παραλληλισμός, Συγχρονισμός, Πολυνηματισμός, PageRank, Community Detection

Abstract

Distance-k graph coloring is a problem that colors the vertices of a graph such that there is no vertices with the same color in k-distance step. As a consequence, distance-1 coloring, colors the vertices of a graph such that no adjacent vertices has the same color. Graph coloring is used in many parallel applications due to its capability on addressing dependency issues by processing vertices of the same color in parallel. Processing data in that way, leads to no need of synchronization between depended data, which as well can lead to smaller execution times.

This diploma thesis aims to analyzing and optimizing the various algorithms proposed in literature for distance-1 graph coloring problem as well as using it on real life applications namely PageRank and Community Detection.

Keywords

Graph, Coloring, Balancing, Parallel Algorithms, Multithreading, Synchronization, PageRank, Community Detection

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Γεώργιο Γκούμα για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Υπολογιστικών Συστημάτων. Επίσης ευχαριστώ ιδιαίτερα την Δρ. Γιαννούλα Χριστίνα για την καθοδήγησή της και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου και τους φίλους μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Σεπτέμβριος 2022

Αθανάσιος Πέππας

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Ο χρωματισμός γράφων, στο πεδίο της θεωρίας γράφων, είναι ένας αλγόριθμος που δίνει στις κορυφές ετικέτες (χρώματα) έτσι ώστε καμία γειτονική κορυφή να μην έχει την ίδια ετικέτα (χρώμα). Η προσέγγιση αυτή χρησιμοποιείται σε πολλές πραγματικές εφαρμογές του, όπως το Chromatic Scheduling [1], η κατανομή καταχωρητών [2], η βελτιστοποίηση [3] και οι υπολογισμοί αραιών πινάκων [4], η δρομολόγηση συγκρουόμενων εργασιών [5] και ο έλεγχος κυκλωμάτων [6]. Η σημασία του χρωματισμού γραφημάτων είναι ότι βοηθάει όλες αυτές τις εφαρμογές να παραλληλοποιήσουν τα δεδομένα τους και δεν υπάρχει πιο αντιπροσωπευτική εφαρμογή από το Chromatic Scheduling, όπου αρχικά χρωματίζεται ο γράφος και στη συνέχεια επεξεργάζεται παράλληλα τις κορυφές με το ίδιο χρώμα. Δύο εφαρμογές που εμπίπτουν στις παραπάνω κατηγορίες και χρησιμοποιούν τον αλγόριθμο χρωματισμού γράφων είναι το PageRank [7] και το Community Detection [8], όπου στην πρώτη τα δεδομένα ενημερώνονται παράλληλα και στη δεύτερη χρησιμοποιείται η χρήση της κλάσης χρώματος για την ταχύτερη σύγκλιση σε ένα αποτέλεσμα. Επομένως, έχει μεγάλη σημασία για τον χρωματισμό γράφων να παράγονται όσο το δυνατόν λιγότερα χρώματα έτσι ώστε να αυξάνεται παραλληλισμός. Σε πολλές περιπτώσεις, ο χρωματισμός ενός γράφου δεν είναι αρκετός καθώς μπορεί να οδηγήσει σε υποαξιοποίηση των πόρων διότι πολλά χρώματα δεν έχουν τον ίδιο αριθμό κορυφών με αποτέλεσμα να υπάρχει ανισοροπία στην δουλειά που κάνουν τα διάφορα νήματα. Έτσι, το επόμενο βήμα του χρωματισμού, είναι η εξισορρόπηση η οποία αποσκοπεί στην ισομερή κατανομή των κορυφών μεταξύ των χρωμάτων. Από τις παραπάνω πληροφορίες είναι προφανές ότι για να επιταχύνουμε οποιαδήποτε εφαρμογή χρησιμοποιώντας τους διάφορους αλγορίθμους χρωματισμού πρέπει αφενός να παράγουμε όσο το δυνατόν λιγότερα χρώματα που να είναι ισορροπημένα και αφετέρου να το κάνουμε αυτό σε βέλτιστο χρόνο εκτέλεσης.

Ωστόσο, η ελαχιστοποίηση του αριθμού των χρωμάτων είναι ένα NP-πλήρες πρόβλημα [9], επομένως η χρήση ευρηστικών μεθόδων είναι αναγκαία. Το κίνητρο αυτής της διατριβής ήταν η διερεύνηση και η σύγκριση των ευριστικών που έχουν προταθεί για τον χρωματισμό γραφημάτων: SS [10], IS [11], IIS [12], HTM-RCU [13], FineGrain και η εξισορρόπηση γράφων: CLU, VFF, VFF-HTM, Scheduled, Recoloring [14] και να εξαχθούν τα χαρακτηριστικά των εισόδων που επηρεάζουν την απόδοση όσον αφορά το χρόνο εκτέλεσης καθώς και τον αριθμό των χρωμάτων. Ο στόχος μας με αυτή την ανάλυση είναι να ταξινομήσουμε αυτούς τους αλγορίθμους και να βρούμε το καταλληλότερο σχήμα ανάλογα με τα χαρακτηριστικά της εισόδου, στην περίπτωσή μας ενός γράφου. Επιπλέον προσπαθήσαμε να βελτιστοποιήσουμε

την ευρηστική HTM-RCU κατανέμοντας τα δεδομένα στα νήματα βέλτιστα. Τέλος, χρησιμοποιήσαμε το PageRank και το Community Detection για να συγκρίνουμε τα ευρήματά μας σε πραγματικές εφαρμογές.

Κεφάλαιο 2

Αλγόριθμοι Χρωματισμού Γράφων

Σε αυτό το κεφάλαιο παρουσιάζουμε τους αλγορίθμους χρωματισμού γράφων στους οποίους βασίσαμε την ανάλυσή μας. Εξετάζουμε 6 αλγορίθμους. Ο πρώτος είναι ο άπληστος αλγόριθμος (2.1) ο οποίος αποτελεί και τον πυρήνα για όλους τους άλλους αλγορίθμους.Ο απευθείας παραλληλισμός του άπληστου αλγόριθμου μπορεί να δημιουργήσει πολλά προβλήματα εξάρτησης, καθώς δύο νήματα μπορεί να αναθέσουν το ίδιο χρώμα σε γειτονικές κορυφές. Οι Sequential-Solve (2.2), Iterative-Solve (2.3) εισάγουν δύο πρόσθετες φάσεις για την αντιμετώπιση αυτών των προβλημάτων προσπαθώντας να τα επιλύσουν διαδοχικά και επαναληπτικά αντίστοιχα. Το μειονέκτημα των αλγορίθμων πολλαπλών φάσεων είναι ότι διασχίζουμε τον γράφο τουλάχιστον δύο φορές. Αυτό το ζήτημα προσπαθεί να λύσει ο Improved-Iterative-Solve (2.4) εισάγοντας την έννοια των γύρων. Σε κάθε γύρο προσπαθεί να επιθεωρήσει μόνο ένα υποσύνολο κορυφών. Ο επόμενος αλγόριθμος χρησιμοποιεί το Hardware Transactional Memory (2.5) για να εξαλείψει τη λογική των πολλαπλών φάσεων καθώς επίσης και να διευθετήσει τα διάφορα πρόβληματα εξάρτησης ενώ ο τελευταίος είναι μια παραλλαγή του HTM με τη διαφορά πώς για τον συγχρονισμό χρησιμοποιεί την έννοια των κλειδωμάτων (2.6).

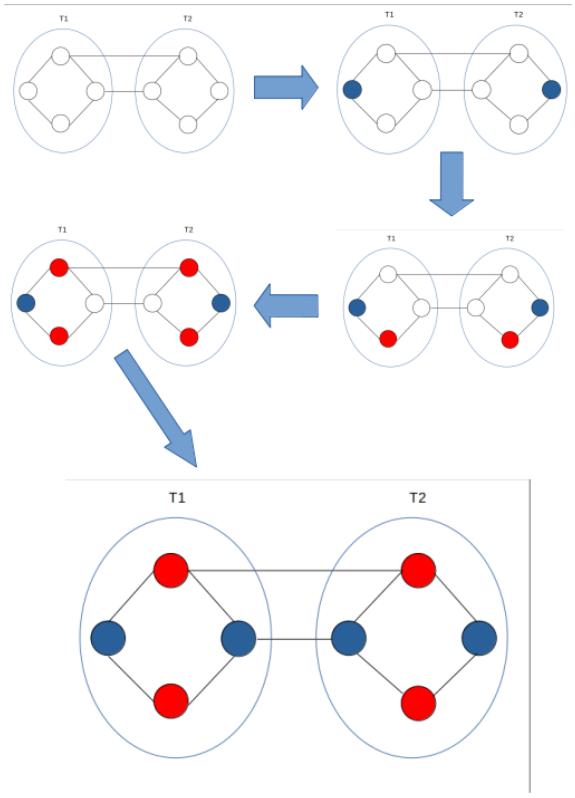
2.1 Άπληστος Αλγόριθμος

Ο άπληστος αλγόριθμος που προτείνεται στο [5] (2.1) είναι ο πυρήνας όλων των άλλων αλγορίθμων και για το λόγο αυτό είναι απαραίτητο να περιγραφεί με όσο το δυνατόν περισσότερη λεπτομέρεια. Σε αυτόν τον αλγόριθμο καθώς και σε όλους τους άλλους αλγορίθμους, το $adj(v)$ δηλώνει τους γείτονες του v. Το *forbiddenColors* είναι ο πίνακας που περιέχει τα χρώματα των γειτόνων μιας κορυφής. Ο πίνακας *color* κρατάει στη θέση v το χρώμα της κορυφής v και αρχικοποιείται με -1 που δηλώνει ότι μια κορυφή δεν έχει χρωματιστεί ακόμα. Η στρατηγική έχει ως εξής, διατρέχει διαδοχικά τον γράφο (γραμμή 2) και για κάθε κορυφή (γραμμή 3), καταγράφει τα χρώματα της γειτονιάς του στο *forbiddenColors* (γραμμή 4). Μόλις ολοκληρωθεί η καταγραφή αυτή, βρίσκει το μικρότερο χρώμα που δεν βρίσκεται στο *forbiddenColors* και το αναθέτει στην κορυφή v (γραμμή 5).

Αλγοριθμος 2.1: *Άπληστος*

```
1: Input : G(V, E)
2: for υ ∈ V do
3:     for u ∈ adj(υ) do
4:         forbiddenColors ← {colors of colored neighbors of υ}
5:     end for
6:     color[υ] ← {smallest color ∉ forbiddenColors}
7: end for
```

Ο άπληστος αλγόριθμος σε πολλές περιπτώσεις είναι ο καταλληλότερος διότι, όπως θα δούμε στο επόμενο κεφάλαιο, δημιουργεί τον μικρότερο αριθμό χρωμάτων, όμως από άποψη εκτέλεσης χρόνου έχει αρκετά περιθώρια βελτίωσης. Το πρώτο βήμα προς την κατεύθυνση αυτή είναι η παραλληλοποίησή του, όμως όπως έχουμε ήδη πει, μια απλή παραλληλοποίηση δημιουργεί προβλήματα και αυτό φαίνεται γραφικά παρακάτω.



Εικόνα 2.1: *Σύγκουρη χρωμάτων κατα την παραλληλοποίηση του άπληστου αλγόριθμου*

Στις επόμενες ενότητες αυτού του κεφαλαίου θα διερευνήσουμε τους διάφορους αλγορίθμους και στρατηγικές που έχουν προταθεί στη βιβλιογραφία για την αντιμετώπιση των προβλημάτων αυτών.

2.2 Sequential-Solve (SS)

Αυτός ο αλγόριθμος που προτείνεται στο [10] αποτελείται από τρεις φάσεις. Στην πρώτη φάση, οι κορυφές του γράφου διαμερίζονται και χρωματίζονται από n νήματα παράλληλα. Μέσα σε κάθε νήμα οι κορυφές χρωματίζονται διαδοχικά με το ελάχιστο νόμιμο χρώμα. Στο τέλος κάθε νήματος υπάρχει ένα φράγμα συγχρονισμού. Λόγω της κατάτμησης, γειτονικές κορυφές μπορεί να καταλήξουν σε διαφορετικά νήματα.Αυτό, παρόλο που λαμβάνονται υπόψη οι ήδη χρωματισμένοι γείτονες εντός του ίδιου νήματος καθώς και οι γείτονες άλλων νημάτων, είναι δυνατόν να οδηγήσει δύο νήματα στο να χρωματίσουν ταυτόχρονα γειτονικές κορυφές με το ίδιο χρώμα. Επειδή δεν υπάρχει τρόπος να αποφευχθούν τέτοιες περιπτώσεις σε αυτή τη φάση, ο χρωματισμός της φάσης αυτής ονομάζεται υποθετικός όπου δε μας ενδιαφέρουν τυχόν ασυνέπειες. Για την επίλυση των ασυνεπειών αυτών δημιουργείται μια δεύτερη φάση κατά την οποία, και πάλι οι κορυφές χωρίζονται σε n νήματα και κάθε νήμα ελέγχει για τυχόν ασυνέπειες. Εάν υπάρχει ασυνέπεια, η κορυφή με τον ελάχιστο δείκτη αποθηκεύεται για μελλοντικό επαναχρωματισμό. Στο τρίτο και τελευταίο βήμα, οι κορυφές που έχουν βρεθεί στη δεύτερη φάση επαναχρωματίζονται κατά διαδοχικό τρόπο. Οι λεπτομέρειες αυτού του αλγορίθμου παρουσιάζονται στο 2.2.

Αλγοριθμος 2.2: *Sequential-Solve*

1: *Input* : $G(V, E)$
2: **for** $v \in V$ **ιν παραλλελ do** ▷ Begin of speculative coloring (Phase 1)
3: **for** $u \in adj(v)$ **do**
4: $forbiddenColors \leftarrow \{colors\ of\ colored\ neighbors\ of\ v\}$
5: **end for**
6: $color[v] \leftarrow \{smallest\ color \notin forbiddenColors\}$
7: **end for**
8: **for** $v \in V$ **ιν παραλλελ do** ▷ Begin of detect conflicts (Phase 2)
9: **for** $u \in adj(v)$ **do**
10: **if** $color[v] == color[u]$ *and* $v < u$ **then**
11: $store(v)$
12: **end if**
13: **end for**
14: **end for**
15: *Resolve Conflicts Sequentially* ▷ Phase 3

2.3 Iterative-Solve (IS)

Αυτός ο αλγόριθμος που προτείνεται στις [15] και [11], προχωράει σε γύρους και κάθε γύρος αποτελείται από 2 φάσεις: τη φάση του υποθετικού χρωματισμού (γραμμές 4-7) και τη φάση ανίχνευσης συγκρούσεων (γραμμές 8-12). Σε κάθε γύρο ο αλγόριθμος εξετάζει παράλληλα ένα υποσύνολο U κορυφών. Ο υποθετικός χρωματισμός είναι ουσιαστικά ο άπληστος αλγόριθμος και σε αυτή τη φάση δεν μας ενδιαφέρουν τυχόν ασυνέπειες. Οι ασυνέπειες που μπορεί να προκύψουν θα εντοπιστούν (γραμμή 11) στη δεύτερη φάση όπου όταν εντοπιστεί σύγκρουση αποθηκεύεται η κορυφή με τον υψηλότερο δείκτη για επαναχρωματισμό στον

επόμενο γύρο (γραμμές 12,13). Ο αλγόριθμος ολοκληρώνεται όταν δεν υπάρχουν κορυφές προς επαναχρωματισμό. Σε αυτόν τον αλγόριθμο, υπάρχουν δύο σημαντικά σημεία συγχρονισμού, ένα στο τέλος της πρώτης φάσης μετά τη γραμμή 7 και ένα στο τέλος της δεύτερης φάσης μετά τη γραμμή 12. Σε κάθε φράγμα συγχρονισμού υπάρχουν n νήματα που πρέπει να συγχρονιστούν, επομένως, συνολικά υπάρχουν 2n νήματα που πρέπει να συγχρονιστούν σε κάθε γύρο. Οι λεπτομέρειες αυτού του αλγορίθμου παρουσιάζονται στο 2.3.

Αλγοριθμος 2.3: *Iterative-Solve*

```
1:  Input : G(V, E)
2:  U = V
3:  while U ≠ 0 do
4:      for υ ∈ U ιν παραλλελ do                ▷ Begin of speculative coloring (Phase 1)
5:          for u ∈ adj(υ) do
6:              forbiddenColors ← {colors of colored neighbors of υ}
7:          end for
8:          color[υ] ← {smallest color ∉ forbiddenColors}
9:      end for
10:     R = ∅                                   ▷ set of vertices to be re-colored
11:     for υ ∈ U ιν παραλλελ do                ▷ Begin of detect conflicts (Phase 2)
12:         for u ∈ adj(υ) do
13:             if color[υ] == color[u] and υ > u then
14:                 R = R ⋃ υ
15:             end if
16:         end for
17:     end for
18:     U = R
19: end while
```

2.4 Improved Iterative-Solve (IIS)

Αυτός ο αλγόριθμος που προτείνεται στο [12] βασίζεται στην επαναληπτική επίλυση (2.3) και αποτελεί μια προσπάθεια να αφαιρεθεί όσο το δυνατόν περισσότερος συγχρονισμός νημάτων. Για να επιτευχθεί αυτό οι φάσεις της ανίχνευσης και της επίλυσης συνδυάζονται σε μία φάση detect-and-resolve. Αυτό είναι εφικτό διότι μόλις διαπιστωθεί ότι μια κορυφή είναι αντικρουόμενη μπορεί να επαναχρωματιστεί αμέσως αντί να περιμένει να επαναχρωματιστεί στον επόμενο γύρο. Τελικά, με αυτή τη στρατηγική τα κύρια σημεία συγχρονισμού μειώνονται από δύο σε ένα μόνο. Μια διαφορά μεταξύ του Iterative-Solve και του Improved Iterative-Solved είναι ότι στον IS ο υποθετικός χρωματισμός πραγματοποιείται μέσα στον βρόχο while συνεπώς εκτελείται πολλές φορές, ενώ στο IIS γίνεται αρχικά ο υποθετικός χρωματισμός και στη συνέχεια γίνεται η επαναληπτική ανίχνευση και επίλυση. Έτσι η IIS εξελίσσεται ως εξής, στην αρχή γίνεται ο υποθετικός χρωματισμός (γραμμές 2-5) όπου επιτρέπονται οι συγκρούσεις και μετά ακολουθεί η φάση ανίχνευσης και επίλυσης συγκρούσεων (γραμμές 8-17) όπου ελέγχονται μόνο οι επαναχρωματισμένες κορυφές του τελευταίου γύρου με σκοπό την ελαχιστοποίηση του υποσυνόλου των κορυφών U^{i-1}.

Αλγοριθμος 2.4: *Improved Iterative-Solve*

```
1: Input : G(V, E)
2: for v ∈ V ιν παραλλελ do                          ▷ speculative coloring (Round 0)
3:     for u ∈ adj(v) do
4:         forbiddenColors ← {colors of colored neighbors of v}
5:     end for
6:     color[v] ← {smallest color ∉ forbiddenColors}
7: end for
8: U^0 ← V                                                    ▷ Inspect all vertices
9: i ← 1                                                      ▷ round counter
10: while U^(i−1) ≠ 0 do                          ▷ Vertices (re)colored in the last round
11:     L ← ∅                                      ▷ List of defectively colored vertices
12:     for v ∈ U^(i−1) ιν παραλλελ do
13:         for u ∈ adj(v) do
14:             if color[v] == color[u] and v < u then
15:                 forbiddenColors ← {colors of colored neighbors}
16:                 color(v) ← {smallest color ∉ forbiddenColors}
17:                 L ← L ⋃ v                          ▷ v was recolored in this round
18:             end if
19:         end for
20:     end for
21:     U_i ← L                          ▷ Vertices to be inspected in the next round
22:     i ← i + 1
23: end while
```

2.5 Hardware Transactional Memory - Read Copy Update

Αυτός ο αλγόριθμος που προτείνεται στο [13] είναι μια διαφορετική προσέγγιση για την παραλληλοποίηση του άπληστου αλγορίθμου. Συγκεκριμένα, χρησιμοποιεί δύο διαφορετικές τεχνικές, το HTM που χρησιμοποιείται στη φάση ανίχνευσης και επίλυσης και το RCU, η οποία είναι μια τεχνική για ταυτόχρονες δομές δεδομένων, με στόχο την ορθότητα και τη μείωση του αποτυπώματος της συναλλαγής.

2.5.1 Hardware Transactional Memory (HTM)

Το Hardware Transactional Memory είναι ένας μηχανισμός που επιτρέπει την ατομική εκτέλεση ενός συνόλου εντολών τα οποία τα λέμε συναλλαγές. Το HTM επιτυγχάνει συντονισμό μεταξύ των νημάτων χάρη στα σύνολα ανάγνωσης/εγγραφής που παρακολουθούν τις θέσεις μνήμης ανάγνωσης/εγγραφής αντίστοιχα. Εάν δεν υπάρχει σύγκρουση μεταξύ των συνόλων οποιουδήποτε νήματος, η συναλλαγή ολοκληρώνεται, διαφορετικά η συναλλαγή αποτυγχάνει και καμία αλλαγή δεν είναι ορατή σε άλλα νήματα.

Ο λόγος για τον οποίο το HTM χρησιμοποιείται στη φάση ανίχνευσης και επίλυσης είναι επειδή από τη φύση του μπορεί να αντιμετωπίσει αυτά τα ζητήματα χωρίς να δηλωθεί ρητά από τον προγραμματιστή. Συγκεκριμένα,

- Η ανίχνευση συγκρούσεων χρωματισμού επιτυγχάνεται με τα σύνολα ανάγνωσης/εγγραφής. Ας υποθέσουμε ότι δύο νήματα T1, T2 δημιουργούν δύο συναλλαγές για δύο

γειτονικές κορυφές u και v αντίστοιχα. Αυτό σημαίνει ότι η συναλλαγή του T1 κατέχει τις αναγνώσεις για τα χρώματα της γειτονιάς της u και την εγγραφή για το χρώμα της u και η συναλλαγή του T2 κατέχει τις αντίστοιχες πληροφορίες για την v. Εάν ο T1 προσπαθήσει να ενημερώσει το χρώμα της u ή ο T2 το χρώμα της v, το HTM θα ανιχνεύσει μια σύγκρουση ανάγνωσης/εγγραφής, επομένως μία θα διακοπεί και μία θα συνεχίσει.

- Η επίλυση των συγκρούσεων επιτυγχάνεται με την πολιτική επίλυσης συγκρούσεων του HTM. Όταν το HTM εντοπίσει σύγκρουση μεταξύ δύο συναλλαγών, αναλόγως την πολιτική του, επιλέγει να διακόψει τη μια συνναλαγή και την άλλη να την συνεχίσει. Αυτή είναι μια τεράστια διαφορά μεταξύ του αλγορίθμου HTM-RCU και των προηγούμενων αλγορίθμων όπου ο προγραμματιστής πρέπει να επιλέξει ρητά την κορυφή που θα ξαναχρωματιστεί ώστε να υπάρχει πρόοδος.

2.5.2 Read Copy Update (RCU)

Read Copy Update (RCU) [16] είναι μια τεχνική ταυτόχρων δομών δεδομένων που επιτρέπει σε πολλαπλά νήματα να επεξεργάζονται δεδομένα ταυτόχρονα. Η ιδέα πίσω από την RCU είναι μάλλον απλή, όταν ένα νήμα θέλει να επεξεργαστεί δεδομένα, τα αντιγράφει σε ένα τοπικά, τα επεξεργάζεται και τέλος ενημερώνει την αρχική δομή δεδομένων. Αυτές οι ενημερώσεις συμβαίνουν σε ένα ατομικό βήμα, έτσι ώστε τα άλλα νήματα να βλέπουν είτε την παλιά έκδοση είτε τη νέα. Για λόγους ορθότητας και συνέπειας, το νήμα ελέγχει πρώτα αν τα δεδομένα στην αρχική δομή δεδομένων είναι αμετάβλητα και αν ναι τα ενημερώνει. Ο έλεγχος και η ενημέρωση πραγματοποιούνται σε ένα μόνο ατομικό βήμα.

Το RCU βοηθά επίσης στη μείωση του αποτυπώματος της συναλλαγής, αντιγράφοντας μόνο τα δεδομένα που πρέπει να υποβληθούν σε επεξεργασία και όχι ολόκληρη τη δομή δεδομένων, επομένως, μέσα στο σύνολο ανάγνωσης της συναλλαγής υπάρχει, σχετικά, μια μικρή ποσότητα δεδομένων.

2.5.3 Αλγόριθμος

Ο αλγόριθμος HTM-RCU εξετάζει παράλληλα τις κορυφές μέσα σε έναν βρόχο φορ και για κάθε κορυφή υπάρχουν δύο φάσεις:

- Φάση 1: Εδώ παρακολουθούμε τα απαγορευμένα χρώματα μιας κορυφής v (τα χρώματα των γειτόνων της) (γραμμή 6) και υπολογίζουμε ένα υποθετικό χρώμα (γραμμή 11). Επίσης σε αυτή τη φάση κρατάμε μόνο τις γειτονικές κορυφές που θα ελεγχθούν στην επόμενη φάση όπως περιγράφεται παραπάνω (γραμμή 8). Εάν, δεν υπάρχουν κορυφές προς επιθεώρηση, αναθέτουμε το υποθετικό χρώμα στο v.

- Φάση 2: Σε αυτή τη φάση, εκτελούμε τη συναλλαγή (γραμμές 15-25/27) και αναθέτουμε ένα χρώμα στην κορυφή (γραμμή 24). Στη γραμμή 17 επιθεωρούμε τις κορυφές της check_list και αν υπάρχει κορυφή με το ίδιο χρώμα (γραμμή 18) διακόπτουμε,

τερματίζουμε τη συναλλαγή και ξεκινάμε από τη φάση 1 (γραμμή 28). Διαφορετικά, αναθέτουμε υποθετικό χρώμα στην κορυφή και τερματίζουμε τη συναλλαγή.

Αλγοριθμος 2.5: *HTM-RCU*

```
1:  Input : G(V, E)
2:  for v ∈ V ιν παραλλελ do
3:      RETRY :
4:      forbidden ← 0
5:      for u ∈ adj(v) do
6:          forbidden ← forbidden ⋃ u.color
7:          if isUncolored(u) and belongs(u) ! = thread_id then
8:              check_list ← check_lists ⋃ u                ▷ uncolored neighbors
9:          end if                                          ▷ assigned to another thread
10:     end for
11:     spec_color ← {smallest color ∉ forbidden}
12:     if check_list == 0 then
13:         color(u) ← spec_color
14:     else
15:         BEGIN_TM
16:         check ← 1
17:         for u ∈ check_list do
18:             if color(u) == spec_color then
19:                 check ← 0
20:                 break
21:             end if
22:         end for
23:         if check == 1 then
24:             color(u) ← spec_color
25:             END_TM
26:         else
27:             END_TM
28:             goTo RETRY
29:         end if
30:     end if
31: end for
```

Ένα άλλο σημαντικό σημείο-κλειδί του αλγορίθμου είναι ότι στην check_list δεν χρειάζεται να συμπεριληφθούν οι κορυφές που αντιστοιχούν στο ίδιο νήμα. Μέσα σε ένα νήμα, οι κορυφές επεξεργάζονται διαδοχικά, επομένως δεν υπάρχει περίπτωση σύγκρουσης. Επίσης, χάρη στους μηχανισμούς HTM και RCU, εάν μια κορυφή καταλήξει σε χρώμα αυτό το χρώμα είναι σωστό, οπότε εκτός από τις κορυφές του ίδιου νήματος, παραλείπουμε από τη check_list και τις χρωματισμένες.

2.6 FineGrain

Ο αλγόριθμος FineGrain είναι μια εναλλακτική λύση του HTM-RCU με τη διαφορά ότι χρησιμοποιεί κλειδώματα για το συγχρονισμό μεταξύ των νημάτων.

2.6.1 FineGrain Technique

Ας υποθέσουμε ότι αποθηκεύουμε τον γράφο σε μια συνδεδεμένη λίστα έτσι ώστε ένας κόμβος να περιέχει τον δείκτη μιας κορυφής και το χρώμα της. Ο πιο προφανής τρόπος συγχρονισμού των νημάτων, είναι να κλειδώσουμε ολόκληρη τη λίστα, να κάνουμε υπολογισμούς και τέλος να την ξεκλειδώσουμε. Είναι σαφές, όμως, ότι με αυτή την προσέγγιση, θα καταλήξουμε σε μεγάλη επιβάρυνση, καθώς διαφορετικά νήματα μπορεί να θέλουν να επεξεργαστούν διαφορετικά μέρη της λίστας που δεν είναι εξαρτημένα, οπότε δεν υπάρχει καθόλου ανάγκη για συγχρονισμό. Για να ξεπεραστεί αυτό, η προσέγγιση FineGrain, όπως περιγράφεται στο [17], χρησιμοποιεί πολλαπλές κλειδαριές για το κλείδωμα μεμονωμένων κόμβων και όχι ολόκληρης της λίστας. Επιπλέον, για να εξασφαλιστεί η πρόοδος προς τα εμπρός, τα κλειδώματα αποκτώνται με την ίδια σειρά από όλα τα νήματα και με την ίδια ακριβώς σειρά απελευθερώνονται στο τέλος.

2.6.2 Αλγόριθμος

Ο αλγόριθμος εξετάζει τον γράφο παράλληλα και για κάθε κορυφή εκτελούνται 2 φάσεις. Η πρώτη φάση (γραμμές 4-13) είναι ακριβώς η ίδια με τον προηγούμενο αλγόριθμο, όπου υπολογίζονται τα απαγορευμένα χρώματα και η check_list. Στη δεύτερη φάση, στη γραμμή 15 πραγματοποιείται η διαδικασία κλειδώματος. Η σειρά κλειδώματος για μια κορυφή v, έχει ως εξής:

- Εάν ο δείκτης της v είναι μικρότερος από την check_list[0], πρώτα κλειδώνουμε την κορυφή v και μετά κλειδώνουμε μία προς μία τις κορυφές της check_list[0].
- Εάν ο δείκτης της v είναι μεταξύ των δεικτών των κορυφών της check_list[0] τότε διατρέχουμε την check_list[0] κλειδώνοντας τις κορυφές, στη συνέχεια κλειδώνουμε και την v και μετά τις υπόλοιπες κορυφές.
- Εάν ο δείκτης του v είναι μεγαλύτερος από την check_list[length-1] τότε κλειδώνουμε πρώτα τις κορυφές της check_list και μετά την κορυφή v.

Μετά από αυτό, πραγματοποιείται η ανίχνευση ασυνέπειας (γραμμές 17-22) και αν δεν υπάρχουν ασυνέπειες, αναθέτουμε το υποθετικό χρώμα στην κορυφή v (γραμμή 24), απελευθερώνουμε τα κλειδώματα με την ίδια ακριβώς σειρά (γραμμή 25) και συνεχίζουμε στην επόμενη κορυφή. Αν υπάρχουν ασυνέπειες, πρώτα απελευθερώνουμε τα κλειδώματα (γραμμή 27) και μετά ξαναπροσπαθούμε από την αρχή (γραμμή 28).

Αλγοριθμος 2.6: *FineGrain Locking*

```
1: Input : G(V, E)
2: for v ∈ V ιν παραλλελ do
3:     RETRY :
4:     forbidden ← 0
5:     for u ∈ adj(v) do
6:         forbidden ← forbidden ⋃ u.color
7:         if isUncolored(u) and belongs(u) ! = thread_id then
8:             check_list ← check_lists ⋃ u                ▷ uncolored neighbors
9:         end if                                          ▷ assigned to another thread
10:    end for
11:    spec_color ← {smallest color ∉ forbidden}
12:    if check_list == 0 then
13:        color(u) ← spec_color
14:    else
15:        lock vertices with some order based on order between u and vertices of check_list
16:        check ← 1
17:        for u ∈ check_list do
18:            if color(u) == spec_color then
19:                check ← 0
20:                break
21:            end if
22:        end for
23:        if check == 1 then
24:            color(u) ← spec_color
25:            unlock vertices with same order as locking
26:        else
27:            unlock vertices with same order as locking
28:            goTo RETRY
29:        end if
30:    end if
31: end for
```

Κεφάλαιο 3

Αλγόριθμοι Χρωματισμού Γράφων : Ανάλυση και Αξιολόγηση

Σε αυτό το κεφάλαιο παρουσιάζουμε τα αποτελέσματα των πειραμάτων μας για τους αλγορίθμους χρωματισμού. Δείχνουμε ότι πράγματι χρησιμοποιοντας πολυνηματικότητα μπορούμε να επιτύχουμε μικρότερους χρόνους εκτέλεσης από άπληστο τον αλγόριθμο [2.1]. Επίσης, συγκρίνουμε τους αλγορίθμους, δείχνοντας ότι στις περισσότερες περιπτώσεις οι αλγόριθμοι SS [2.2] και HTM-RCU [2.5] είναι οι καταλληλότεροι αλγόριθμοι ανάλογα με τα χαρακτηριστικά του κάθε γράφου. Επιπροσθέτως, καταφέραμε να βελτιώσουμε το σχήμα HTM-RCU διαμοιράζοντας τις κορυφές του γράφου στα νήματα με βάση τον αριθμό των ακμών δείχνοντας ότι σε πολλές περιπτώσεις ο καλύτερος αλγόριθμος είναι ο HTM-RCU και οι παραλλαγές του. Τέλος, συγκρίνουμε τον αριθμό χρωμάτων που χρησιμοποιούνται, δείχνοντας ότι παρόλο που ο άπληστος, καθώς και οι σειριακές εκδόσεις, παρέχουν λιγότερα χρώματα, οι παράλληλοι αλγόριθμοι μπορούν να επιτύχουν εξίσου λίγα. Οι παραπάνω παρατηρήσεις δείχνουν ότι υπάρχει ένα trade off μεταξύ των λίγων χρωμάτων και του μικρού χρόνου εκτέλεσης και τελικά, το τι θα επιλέξουμε να χρησιμοποιήσουμε εξαρτάται από την εφαρμογή που εκτελούμε.

3.1 Μεθοδολογία

Για τα πειράματά μας χρησιμοποιήσαμε ένα σύνολο 26 γράφων 5.1. 12 παραχθέντες (Rmat−××), που στο εξής θα αναφέρονται ως συνθετικοί, και 14 πραγματικούς (Real−××).Για τους συνθετικούς γράφους έχουμε την εξής παραμετροποίηση:

- Rmat(1|2|3|4)-1: A = 0.25 B = 0.25 C = 0.25 D = 0.25
- Rmat(1|2|3|4)-2: A = 0.45 B = 0.15 C = 0.15 D = 0.25
- Rmat(1|2|3|4)-3: A = 0.55 B = 0.15 C = 0.15 D = 0.15

Συνθετικοί γράφοι με ίσα A,B,C και D είναι πιο κανονικοποιημένοι και κατ' επέκταση πιο εύκολοι στους υπολογισμούς ενώ όσο διαφοροποιούμε τις παραμέτρους αυτές γίνονται όλο και πιο ακανόνιστοι δηλαδή δύσκολοι. Όσον αφορά τους πραγματικούς γράφους, είναι όλοι ακανόνιστοι.

Graph	Source	GraphID	V	E	MaxDeg	AvgDeg	STD
rmat-10Mx100M	Generated	Rmat1-1	10M	100M	103	19,9	10,3
rmat-10Mx100M	Generated	Rmat1-2	10M	100M	2193	19,9	27,8
rmat-10Mx100M	Generated	Rmat1-3	10M	100M	46303	19,9	107,7
rmat-10Mx200M	Generated	Rmat2-1	10M	200M	183	39,9	19,6
rmat-10Mx200M	Generated	Rmat2-2	10M	200M	4381	39,9	55,2
rmat-10Mx200M	Generated	Rmat2-3	10M	200M	83878	39,7	211,1
rmat-10Mx400M	Generated	Rmat3-1	10M	400M	339	79,9	38,2
rmat-10Mx400M	Generated	Rmat3-2	10M	400M	8819	79,9	110
rmat-10Mx400M	Generated	Rmat3-3	10M	400M	147359	79,1	410
rmat-10Mx50M	Generated	Rmat4-1	10M	50M	54	9,9	5,6
rmat-10Mx50M	Generated	Rmat4-2	10M	50M	1127	9,9	14
rmat-10Mx50M	Generated	Rmat4-3	10M	50M	24861	9,9	54,7
arabic-2005	[18]	Real-1	22M	639M	9905	28,1	78,8
com-Orkut	[18]	Real-2	3M	117M	33313	76,2	154,8
FullChip	[18]	Real-3	2.9M	26M	2312481	8,9	1806,8
indochina-2004	[18]	Real-4	7M	194M	6985	26,1	215,8
kmer_A2a	[18]	Real-5	170M	180M	40	2,1	0,6
kmer_V1r	[18]	Real-6	214M	232M	8	2,1	0,6
mawi_2015	[18]	Real-7	226M	240M	2107954	2,1	14016
mycielskian19	[18]	Real-8	0.3M	451M	196607	2296,9	4530
nlpkkt200	[18]	Real-9	27M	401M	28	27,6	2,4
nlpkkt240	[18]	Real-10	16M	232M	28	27,6	2,2
Queen_4147	[18]	Real-11	4M	166M	81	79,4	6,3
stokes	[18]	Real-12	11M	349M	720	30,5	41,4
sx-stackoverflow	[18]	Real-13	2M	36M	38148	13,9	137,8
webbase-2001	[18]	Real-14	118M	1019M	3841	8,6	26

3.1: *Graph Suite*

Για την αξιολόγηση των αλγορίθμων, καθώς και για τη σύγκρισή τους, χρησιμοποιούμε τη μετρική επιτάχυνσης (speedup) η οποία δίνεται από την ακόλουθη έκφραση

$$Speedup = \frac{T_s}{T_p}$$

όπου T_s είναι ο χρόνος εκτέλεσης της σειριακής έκδοσης και T_p της παράλληλης. Η επιτάχυνση δείχνει πόσες φορές η παράλληλη έκδοση ενός αλγορίθμου είναι ταχύτερη από τη σειριακή. Μια άλλη μετρική για σύγκριση είναι ο αριθμός των χρωμάτων που παράγει κάθε σχήμα. Πέρα από τις μετρικές που προσανατολίζονται στο χρόνο για να συγκρίνουμε τα γραφήματα με βάση τα χαρακτηριστικά τους, χρησιμοποιούμε τη μετρική τυπική απόκλιση (std) η οποία μας δείνει πόσο διαφέρει ο βαθμός των κορυφών από το μέσο βαθμό του γράφου.

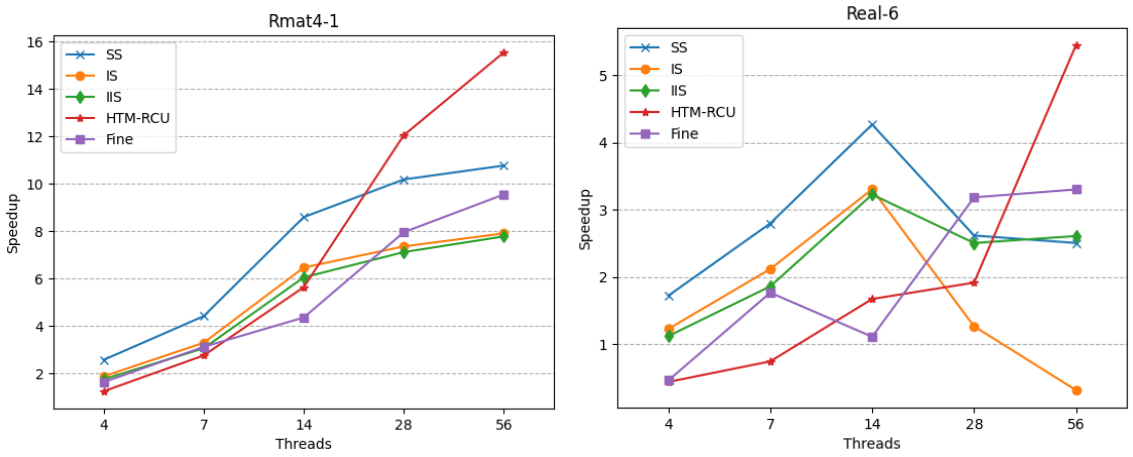
Τα πειράματα πραγματοποιήθηκαν στη μηχανή Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz η οποία έχει 2 υποδοχές με 1 cpu η κάθε μία και σε κάθε cpu υπάρχουν 28 νήματα, οπότε έχουμε στη διάθεσή μας 56 νήματα υλικού. Τέλος, πήραμε αποτελέσματα για 1, 4, 7, 14, 28 και 56 νήματα.

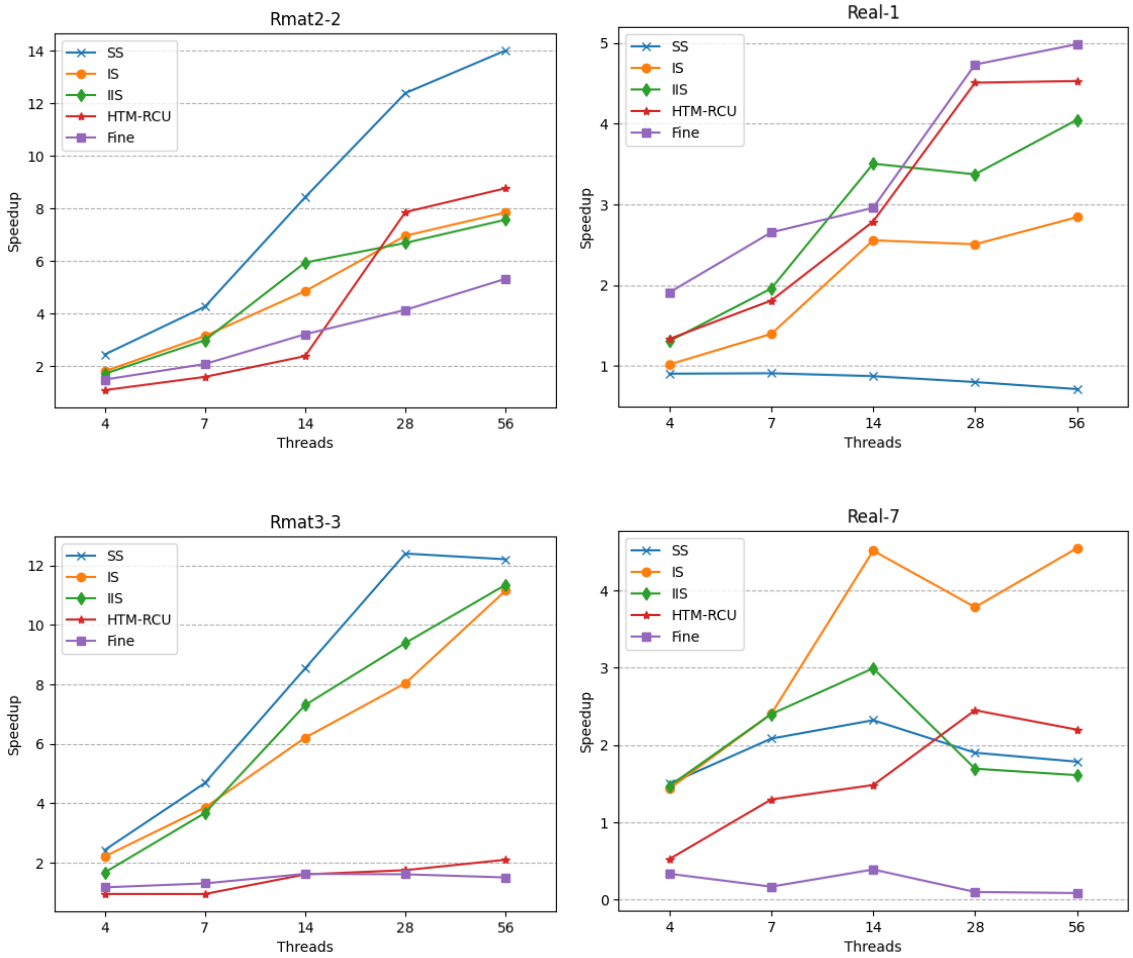
3.2 Κλιμακωσιμότητα

Αρχικά πρέπει να δούμε, εάν έχουμε επιτάχυνση σε σχέση με τον άπληστο που είναι και ο απώτερος στόχος μας.

Στο σχήμα 3.1 έχουμε ένα δείγμα 6 γράφων (3 συνθετικοί (αριστερά) και 3 πραγματικοί (δεξιά» με διάφορες τιμές τυπικής απόκλισης (std). Οι γράφοι στην πρώτη σειρά έχουν χαμηλό std της τάξης του 0.5, οι μεσαίοι, έχουν μεσαίες τιμές std της τάξης 10× ενώ στην τελευταία σειρά έχουν τιμές std της τάξης του 1000×. Η επιτάχυνση που υπολογίζουμε είναι σε σχέση με τον άπληστο αλγόριθμο. Αυτό σημαίνει ότι δείχνουμε πόσες φορές ένας αλγόριθμος είναι ταχύτερος ή βραδύτερος από τον άπληστο αλγόριθμο. Η πρώτη διαπίστωση που μπορούμε να αντλήσουμε είναι ότι οι συνθετικοί γράφοι επιταχύνουν καλύτερα από τους πραγματικούς, επιτυγχάνοντας επιτάχυνση πάνω από 10×. Αυτό οφείλεται στην αυξημένη ακανόνιστη μορφή των πραγματικών γραφημάτων. Δεύτερον, όλοι ανεξαιρέτως οι γράφοι πετυχαίνουν κάποια στιγμή επιτάχυνση. Για παράδειγμα, ο Rmat4-1, είναι ένα παράδειγμα καλής περίπτωσης, καθώς επιτυγχάνεται επιτάχυνση για όλους τους αλγορίθμους και για αυξανόμενο αριθμό νημάτων. Από την άλλη πλευρά, μπορούμε να δούμε ότι παρόλο που ο Real-7 είναι μια δύσκολη περίπτωση, υπάρχει επιτάχυνση για κάποια ζεύγη αλγορίθμων,νημάτων όπως για παράδειγμα ο IS πέτυχε επιτάχυνση 4.54× σε 56 νήματα.

Όπως μπορούμε να δούμε από τους πίνακες 3.2 και 3.3 αυτές οι παρατηρήεσις μπορούν να επεκταθούν σε ολόκληρη τη σουίτα γραφημάτων, καθώς πετυχάινουμε επιτάχυνση για όλους τους γράφους, αποδεικνύοντας ότι πράγματι υπάρχουν οφέλη από τη χρήση νημάτων όσον αφορά το χρόνο εκτέλεσης. Επίσης, μπορούμε να δούμε ότι για διαφορετικούς γράφους υπάρχει ένας αλγόριθμος που ταιριάζει καλύτερα, όπως θα διερευνήσουμε στην επόμενη ενότητα.





Εικόνα 3.1: *Speedup over Greedy*

Graph	SS					IS					IIS				
Rmat1-1	3,88	7,10	13,57	17,67	19,52	3,94	6,82	13,42	17,02	18,64	2,83	4,95	9,66	12,92	14,29
Rmat1-2	3,99	7,12	13,74	18,83	21,41	3,92	6,91	13,72	18,80	21,10	2,84	5,02	10,17	13,97	15,98
Rmat1-3	3,97	6,94	13,53	18,66	20,16	3,12	6,96	11,05	12,57	14,30	2,89	5,18	8,17	11,85	9,37
Rmat2-1	3,99	6,91	13,69	18,03	19,85	3,92	6,82	13,38	17,37	19,09	2,77	4,81	9,46	12,55	13,66
Rmat2-2	4,01	7,02	13,90	20,45	23,13	3,93	6,89	10,64	15,28	17,23	2,83	4,98	9,92	11,17	12,65
Rmat2-3	4,01	6,92	13,39	19,43	20,35	3,98	6,96	13,95	16,68	15,70	2,95	5,15	8,22	10,41	12,01
Rmat3-1	3,99	7,07	13,78	17,88	19,37	3,87	7,03	13,70	17,62	19,13	2,68	4,86	9,61	12,14	13,29
Rmat3-2	4,01	7,07	13,93	21,04	23,60	3,89	6,97	13,76	15,49	16,73	2,78	3,91	9,63	14,44	17,18
Rmat3-3	3,46	6,68	12,17	17,68	17,40	3,99	6,95	11,18	14,48	20,09	2,44	5,36	10,65	13,70	16,55
Rmat4-1	4,03	6,94	13,54	16,05	16,98	3,90	6,88	13,55	15,43	16,58	2,86	5,02	9,96	11,72	12,82
Rmat4-2	4,04	7,05	13,55	17,50	18,86	4,04	7,35	14,22	17,75	19,71	2,85	5,12	9,97	13,10	14,63
Rmat4-3	3,97	6,80	13,15	17,74	19,06	3,96	6,95	13,53	11,40	13,00	2,89	5,08	10,09	10,82	10,20
Real-1	1,74	1,75	1,68	1,54	1,37	1,72	2,37	4,33	4,25	4,82	1,74	2,61	4,66	4,49	5,39
Real-2	3,93	6,57	12,46	17,42	19,55	2,45	3,73	8,58	10,33	12,09	2,16	3,74	7,38	11,08	10,61
Real-3	1,70	1,97	2,22	2,10	1,92	2,18	3,60	6,21	6,08	4,73	2,20	3,65	6,63	6,59	4,74
Real-4	1,44	1,76	1,87	1,83	1,88	2,66	3,45	4,78	4,59	4,08	2,90	4,57	6,64	6,87	5,78
Real-5	2,93	4,80	7,69	6,38	6,70	2,74	4,91	7,61	6,32	6,56	1,88	3,48	6,09	4,81	4,78
Real-6	2,75	4,48	6,83	4,19	4,01	2,55	4,40	6,88	2,63	0,65	1,88	3,12	5,42	4,21	4,38
Real-7	2,47	3,43	3,82	3,13	2,94	2,45	4,11	7,72	6,47	7,77	2,54	4,15	5,17	2,92	2,78
Real-8	3,99	6,90	13,11	15,26	13,51	3,99	6,94	13,24	13,15	9,99	2,83	4,94	9,59	9,94	9,94
Real-9	1,73	1,95	2,12	1,99	1,97	5,04	8,26	13,22	8,23	8,29	3,66	6,11	10,69	7,47	7,44
Real-10	1,73	1,95	2,12	1,98	1,97	5,00	8,37	13,18	8,20	8,11	3,71	6,31	10,60	7,22	7,24
Real-11	1,64	1,78	1,96	1,93	2,02	2,72	4,13	6,10	5,85	5,57	1,99	3,08	5,52	4,95	5,82
Real-12	1,74	1,99	2,21	2,05	2,10	3,50	4,55	8,51	8,90	10,41	2,45	4,11	7,72	6,47	7,77
Real-13	1,87	2,12	2,33	2,28	2,29	2,89	3,93	7,60	8,65	9,91	2,89	5,08	7,75	11,36	10,71
Real-14	1,78	2,25	2,63	2,27	1,22	1,27	1,98	2,61	1,24	0,78	1,34	2,06	3,87	1,44	3,66

3.2: *Speedup: SS, IS, IIS*

Graph	HTM-RCU					FineGrain				
Rmat1-1	2,08	3,39	9,62	20,60	26,63	2,43	4,39	5,14	9,33	12,99
Rmat1-2	2,15	2,88	7,24	15,85	16,66	2,46	4,01	5,53	7,87	7,23
Rmat1-3	1,83	2,14	3,70	8,13	8,71	1,67	2,03	2,71	3,21	3,64
Rmat2-1	3,49	3,77	14,27	23,66	29,43	2,99	4,44	6,50	11,41	12,02
Rmat2-2	2,12	3,12	4,68	15,52	17,31	2,20	3,08	4,77	6,16	7,92
Rmat2-3	1,37	1,50	3,01	4,24	5,46	1,67	2,07	2,41	2,52	2,51
Rmat3-1	3,51	7,09	14,18	20,75	33,60	2,83	4,24	7,84	10,48	11,05
Rmat3-2	2,03	3,01	4,54	13,30	14,43	2,19	3,13	4,97	5,72	6,70
Rmat3-3	1,71	1,71	2,89	3,15	3,78	1,61	1,80	2,24	2,22	2,07
Rmat4-1	1,76	3,92	8,04	17,21	22,21	2,35	4,53	6,34	11,59	13,92
Rmat4-2	1,59	3,80	5,43	11,43	14,62	2,67	3,39	6,16	9,01	9,74
Rmat4-3	2,59	2,66	4,64	8,06	9,17	1,79	2,77	3,13	4,42	3,85
Real-1	2,99	4,06	6,27	10,13	10,18	2,77	3,85	4,29	6,87	7,23
Real-2	3,00	5,81	8,34	11,76	15,36	2,02	2,35	4,06	3,10	4,33
Real-3	1,80	2,70	3,08	4,95	6,79	1,69	0,96	1,43	2,44	1,97
Real-4	1,11	1,15	1,16	1,46	1,53	1,10	1,16	1,16	1,41	1,48
Real-5	1,55	2,07	2,89	16,11	20,71	1,95	1,94	2,56	9,93	11,49
Real-6	1,43	2,44	5,49	6,29	17,92	1,38	5,30	3,33	9,57	9,92
Real-7	1,51	3,73	4,27	7,06	6,34	2,19	1,10	2,56	0,65	0,55
Real-8	2,00	3,40	5,22	6,62	5,78	0,58	0,59	0,44	0,30	0,27
Real-9	0,42	4,13	4,58	2,89	12,47	1,05	1,07	3,83	3,81	13,32
Real-10	1,20	6,28	4,51	3,17	19,04	0,84	2,40	1,89	2,33	12,90
Real-11	4,40	7,51	14,57	27,02	29,51	4,29	2,80	17,42	26,81	28,19
Real-12	2,98	10,00	13,76	26,29	35,36	1,76	2,60	5,58	5,56	8,05
Real-13	1,66	2,32	3,39	4,69	5,66	1,55	2,22	2,17	2,51	2,61
Real-14	2,42	4,58	18,36	32,82	39,53	2,47	3,56	9,41	11,66	11,94

3.3: *Speedup:HTM-RCU, FineGrain*

3.3 Σύγκριση Αλγορίθμων

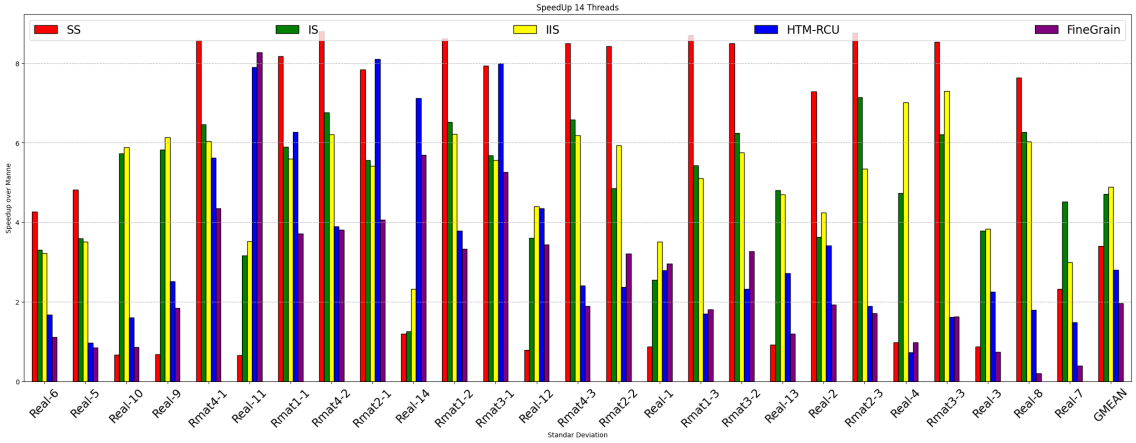
Καθώς εμβαθύνουμε στους πίνακες 3.2 και 3.3 με περισσότερες λεπτομέρειες, μπορούμε να δούμε ότι υπάρχει διαφορά μεταξύ των επιταχύνσεων. Έτσι, σε αυτή την ενότητα θα διερευνήσουμε τα χαρακτηριστικά ενός γράφου που επηρεάζουν την απόδοση. Για τη σύγκρισή μας θα χρησιμοποιήσουμε και πάλι τον άπληστο αλγόριθμο (speedup over Greedy) και θα ταξινομήσουμε τους γράφους με βάση την τιμή STD.

Στα σχήματα 3.2-3.4, έχουμε ταξινομήσει τους γράφους με βάση τις τιμές STD από αριστερά προς τα δεξιά κατά αύξουσα σειρά. Βλέπουμε καθαρά ότι, στα 14 νήματα ο κυρίαρχος αλγόριθμος είναι ο SS (κόκκινο χρώμα), καθώς πετυχαίνει την καλύτερη επιτάχυνση σε 14 από τους 26 γράφους. Ωστόσο, καθώς αυξάνουμε τον αριθμό των νημάτων βλέπουμε ότι ο HTM-RCU (μπλε χρώμα) κερδίζει έδαφος επιτυγχάνοντας την καλύτερη απόδοση σε 8 από τους 26 γράγους στα 28 νήματα και στους 12 στα 56 νήματα, ενώ σε 14 νήματα ήταν ο καλύτερος αλγόριθμος για μόνο 3 εισόδους. Αυτό οφείλεται στην πολιτική ανίχνευσης και επίλυσης κάθε αλγορίθμου. Καθώς αυξάνεται ο αριθμός των νημάτων, υπάρχουν περισσότερες συγκρούσεις και επομένως είναι κρίσιμος ο τρόπος με τον οποίο τις αντιμετωπίζουμε. Στον HTM-RCU, οι συναλλαγές αντιμετωπίζουν αυτές τις συγκρούσεις αυτόματα, ενώ ο SS τις ανιχνεύει πρώτα παράλληλα και μετά τις επιλύει διαδοχικά. Η σειριακή φάση της SS προσθέτει μεγάλο φώρτο εργασίας που δεν έχει ο HTM-RCU, επομένως σε γράφους με χαμηλό std ο HTM-RCU αποδίδει καλύτερα. Ωστόσο, ενώ ο HTM-RCU αποδίδει καλά για γράφους με χαμηλό STD όταν έχουμε πολλά νήματα, αυτή η ικανότητα έχει ένα άνω όριο. Αυτό το όριο οφείλεται στη διαθέσιμη μνήμη που έχει μια συναλλαγή όπου κρατάει τα σύνολα ανάγνωσης/εγγραφής και είναι θέμα υλικού. Αυτό σημαίνει ότι μια συναλλαγή μπορεί να

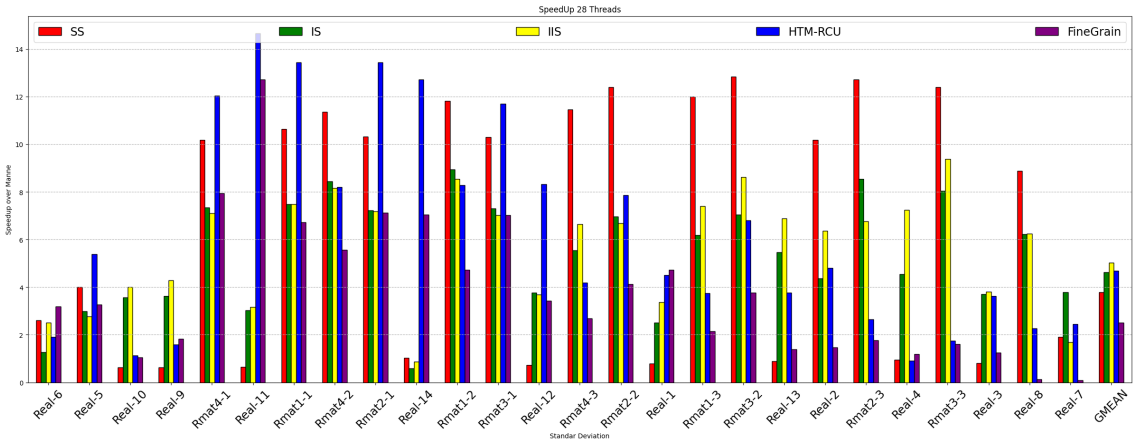
υποφέρει από έλλειψη μνήμης, γεγονός που μπορεί να οδηγήσει σε συνεχείς ματαιώσεις έως ότου τελικά μια συναλλαγή δεσμευτεί. Αυτό το σενάριο είναι σύνηθες σε γράφους υψηλού STD και σε πολλά νήματα και τα πειράματά μας έδειξαν ότι ο καλύτερος τρόπος για την επίλυσή του, είναι η διαδοχική επίλυση συνεπώς ο SS. Αυτό μπορεί να φανεί στα διαγράμματα 3.2-3.4 όπου καθώς αυξάνεται ο αριθμός των νημάτων τείνουν να χωρίζονται στη μέση με το HTM-RCU να αποδίδει καλά για γράφους χαμηλού STD και το SS με γραφήματα υψηλού STD. Συμπερασματικά, το HTM-RCU ταιριάζει καλύτερα για γραφήματα με χαμηλό STD σε 28 και 56 νήματα, ενώ το SS ταιριάζει καλύτερα όταν τρέχουμε με λίγα νήματα ή για εισόδους με υψηλό STD.

3.4: *Best Algorithm per Thread*

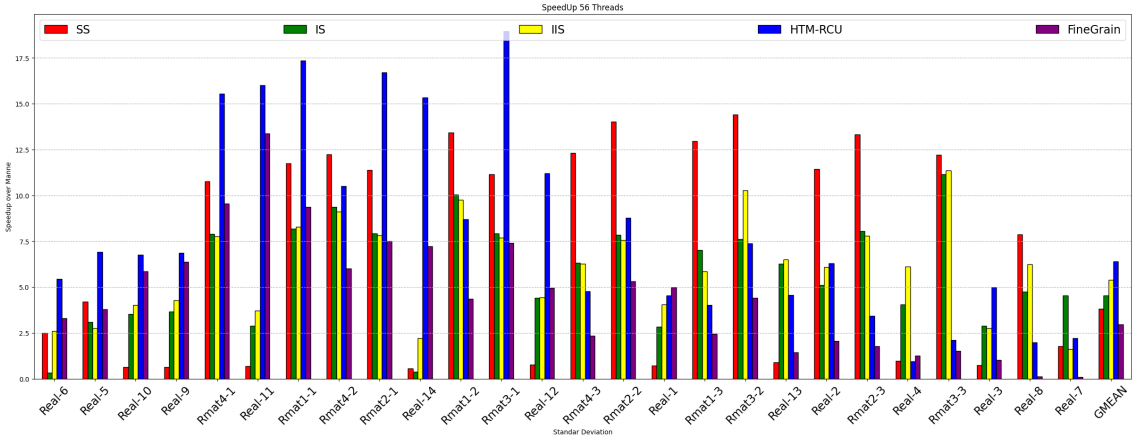
Graph	**4**	**7**	**14**	**28**	**56**
Rmat1-1	SS	SS	SS	HTM	HTM
Rmat1-2	SS	SS	SS	SS	SS
Rmat1-3	SS	SS	SS	SS	SS
Rmat2-1	SS	SS	HTM	HTM	HTM
Rmat2-2	SS	SS	SS	SS	SS
Rmat2-3	SS	SS	SS	SS	SS
Rmat3-1	SS	SS	HTM	HTM	HTM
Rmat3-2	SS	SS	SS	SS	SS
Rmat3-3	SS	SS	SS	SS	SS
Rmat4-1	SS	SS	SS	HTM	HTM
Rmat4-2	SS	SS	SS	SS	SS
Rmat4-3	SS	SS	SS	SS	SS
Real-1	FINE	FINE	IIS	FINE	FINE
Real-2	SS	SS	SS	SS	SS
Real-3	IS	IS	IIS	IIS	HTM
Real-4	IIS	IIS	IIS	IIS	IIS
Real-5	SS	SS	SS	HTM	HTM
Real-6	SS	SS	SS	FINE	HTM
Real-7	SS	IS	IS	IS	IS
Real-8	SS	SS	SS	SS	SS
Real-9	IS	IS	IIS	IIS	HTM
Real-10	IS	IS	IIS	IIS	HTM
Real-11	HTM	HTM	FINE	HTM	HTM
Real-12	IIS	HTM	IIS	HTM	HTM
Real-13	IS	IIS	IS	IIS	IIS
Real-14	FINE	FINE	HTM	HTM	HTM



Εικόνα 3.2: *Speedup over Greedy 14 Threads*



Εικόνα 3.3: *Speedup over Greedy 28 Threads*



Εικόνα 3.4: *Speedup over Greedy 56 Threads*

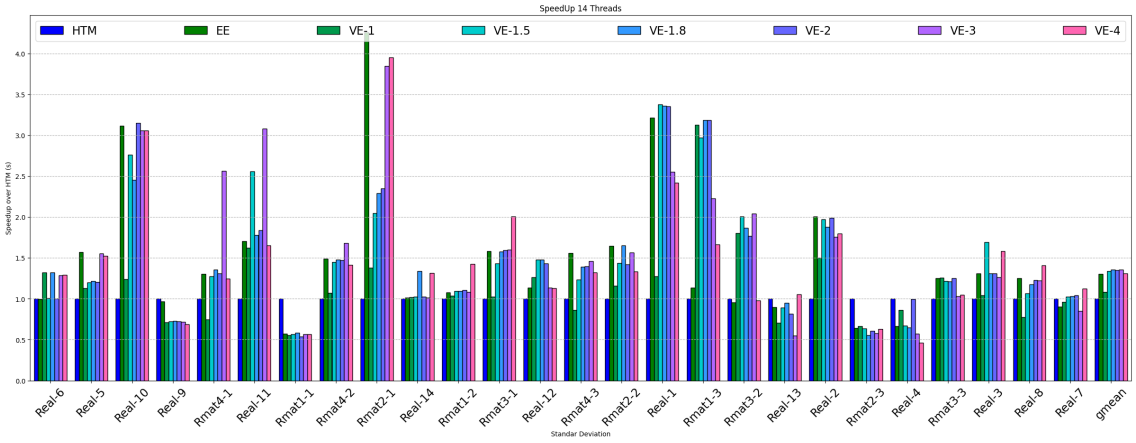
3.4 HTM-RCU σε βάθος

Ο παραλληλισμός των 3 πρώτων αλγορίθμων (SS,IS,IIS) έγινε με τον μηχανισμό OpenMP, ο οποίος κατανέμει αυτόματα τα δεδομένα σε νήματα. Από την άλλη πλευρά, για τον HTM-RCU, τα δεδομένα πρέπει να κατανεμηθούν ρητά από τον προγραμματιστή. Η κατανομή που ακολουθήθηκε στα προηγούμενα πειράματα ήταν η ισοκατανομή των κορυφών μεταξύ των νημάτων. Ωστόσο, αυτό μπορεί να οδηγήσει σε άνιση εργασία από τα νήματα λόγω της διαφοράς των βαθμών των κορυφών. Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα γράφημα με 200 κορυφές και ότι τρέχουμε με 2 νήματα, T1, T2. Αυτό σημαίνει ότι 100 κορυφές θα ανατεθούν στο T1 και 100 στο T2. Εάν οι συνολικοί γείτονες του T2 είναι διπλάσιοι από αυτούς του T1, ο T2 πρέπει να κάνει διπλάσια δουλειά, οδηγώντας σε άνιση κατανομή του χρόνου, καθώς ο T1 θα περιμένει και δεν θα κάνει τίποτα μέχρι να τελειώσει ο T2. Αυτή είναι μια μεγάλη απώλεια πόρων. Η απλή λύση είναι η κατανομή των κορυφών μεταξύ των νημάτων με βάση τον αριθμό των ακμών, ώστε κάθε νήμα να εργάζεται με τον ίδιο αριθμό γειτόνων. Επίσης, δοκιμάσαμε μια εκδοχή όπου κάναμε μια υβριδική κατανομή που σημαίνει ότι δοκιμάσαμε μια ισοκατανομή όσον αφορά τις ακμές ενώ προσπαθούμε να έχουμε όσο το δυνατόν ίσες κορυφές μεταξύ των νημάτων. Για την τελευταία έκδοση δοκιμάσαμε διαφορετικές τιμές του αριθμού των κορυφών που ανατίθενται σε ένα νήμα πριν ικανοποιηθεί η συνθήκη ίσων ακμών, όπως φαίνεται παρακάτω. Έτσι, έχουμε τις ακόλουθες εκδόσεις:

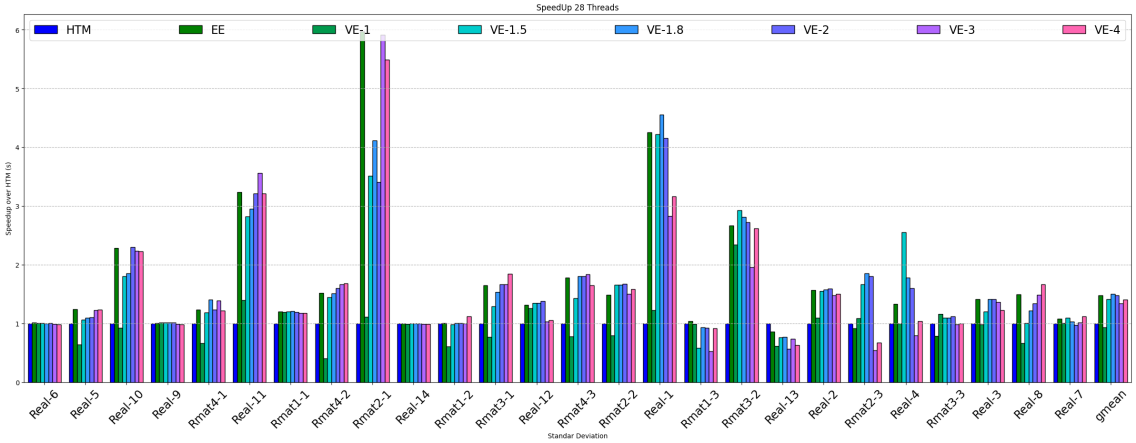
- **HTM-RCU**: The original version with equal number of vertices among threads.
- **EE**: Equal number of edges among threads.
- **VE**: Equal number of edges among threads or equal number of vertices.
  - 1, 1.5, 1.8, 2, 3, 4

Στα σχήματα 3.5-3.7 βλέπουμε ότι πράγματι η ίση κατανομή των ακμών στις περισσότερες περιπτώσεις βελτίωνει τον χρόνο εκτέλεσης, επιτυγχάνοντας για παράδειγμα στο Rmat2-1 επιτάχυνση 5,96 και αν δώσουμε προσοχή στο γεωμετρικό μέσο θα δούμε ότι για 14 νήματα όλες οι παραλλαγές αποτελούν βελτίωση και για 28/56 νήματα όλες εκτός από το ΄Ε-1, γεγονός που μας δείχνει ότι η ίση κατανομή των ακμών μπορεί να οδηγήσει σε καλύτερους χρόνους εκτέλεσης.

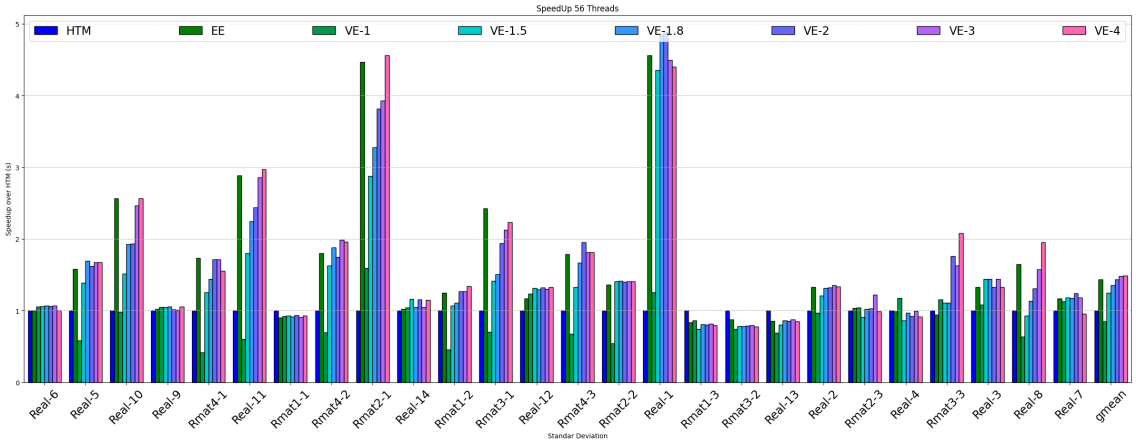
Τέλος, παρουσιάζουμε στον πίνακα 3.5 τα αποτελέσματα από αυτές τις βελτιώσεις παρόμοια με τον πίνακα 3.4 για να δείξουμε ότι σε πολλές περιπτώσεις οι παραλλαγές HTM-RCU ξεπέρασαν όλους τους άλλους αλγορίθμους σε 17 από τους 26 γράφους, καθιστώντας τους τον κυρίαρχο αλγόριθμο στη σουίτα μας.



Εικόνα 3.5: *Speedup over HTM-RCU 14 Threads*



Εικόνα 3.6: *Speedup over HTM-RCU 28 Threads*



Εικόνα 3.7: *Speedup over HTM-RCU 56 Threads*

3.5: *Best Algorithm per Input*

Graph	**Best Algorithm**	**Thread**
Rmat1-1	VE-2	56
Rmat1-2	VE-1.8	56
Rmat1-3	SS	56
Rmat2-1	VE-1.8	56
Rmat2-2	EE	56
Rmat2-3	SS	56
Rmat3-1	HTM-RCU	56
Rmat3-2	SS	56
Rmat3-3	SS	28
Rmat4-1	VE-1.5	56
Rmat4-2	VE-2	56
Rmat4-3	SS	56
Real-1	VE-2	28
Real-2	SS	56
Real-3	VE-1.8	56
Real-4	IIS	28
Real-5	HTM	28
Real-6	VE-1.5	28
Real-7	IS	56
Real-8	SS	56
Real-9	EE	56
Real-10	EE	56
Real-11	VE-2	56
Real-12	VE-1.5	56
Real-13	EE	56
Real-14	VE-2	56

3.5 Αριθμός Παραγώμενων Χρωμάτων

Για την ποιότητα του χρωματισμού και τον αριθμό των χρωμάτων που παράγουν τα σχήματα θα συγκρίνουμε αρχικά τις σειριακές παραλλαγές με τον άπληστο αλγόριθμο. Ο άπληστος αλγόριθμος λόγω της διαδοχικής του φύσης παράγει χρωματισμό με χρώματα κοντά στο βέλτιστο. Στον πίνακα 3.6 μπορούμε να δούμε την σειριακή έκδοση των σχημάτων μας- όλα τα σχήματα και για όλες τις εισόδους παράγουν τον ίδιο αριθμό χρωμάτων με τον άπληστο. Φυσικά αυτό θα έπρεπε να συμβαίνει, διότι στη σειριακή λειτουργία, όλα τα σχήματά μας είναι ουσιαστικά ο άπληστος αλγόριθμος.

Graph	Greedy	SS	IS	IIS	HTM	FineGrain
Rmat1-1	14	14	14	14	14	14
Rmat1-2	30	30	30	30	30	30
Rmat1-3	123	123	123	123	123	123
Rmat2-1	21	21	21	21	21	21
Rmat2-2	45	45	45	45	45	45
Rmat2-3	201	201	201	201	201	201
Rmat3-1	32	32	32	32	32	32
Rmat3-2	72	72	72	72	72	72
Rmat3-3	334	334	334	334	334	334
Rmat4-1	10	10	10	10	10	10
Rmat4-2	20	20	20	20	20	20
Rmat4-3	77	77	77	77	77	77
Real-1	3248	3248	3248	3248	3248	3248
Real-2	116	116	116	116	116	116
Real-3	8	8	8	8	8	8
Real-4	6849	6849	123	6849	6849	6849
Real-5	11	11	11	11	11	11
Real-6	6	6	6	6	6	6
Real-7	10	10	10	10	10	10
Real-8	19	19	19	19	19	19
Real-9	2	2	2	2	2	2
Real-10	2	2	2	2	2	2
Real-11	39	39	39	39	39	39
Real-12	31	31	31	31	31	31
Real-13	136	136	136	136	136	136
Real-14	1507	1507	1507	1507	1507	1507

3.6: *Number of Colors: Serial*

Έτσι, από εδώ και στο εξής για να αξιολογήσουμε και να καταλήξουμε σε ένα συμπέρασμα, θα χρησιμοποιήσουμε ως βάση σύγκρισης τις σειριακές εκδόσεις κάθε σχήματος. Στους πίνακες 3.7-3.10 βλέπουμε τους αριθμούς που έχουν παραχθεί σε όλες τις εκδόσεις (4, 7, 14, 28 και 56 νήματα). Η πρώτη διαπίστωση που μπορούμε να αντλήσουμε είναι ότι όσο αυξάνεται ο αριθμός των νημάτων αυξάνεται και ο αριθμός των χρωμάτων. Μια τέτοια συμπεριφορά είναι αναμενόμενη, διότι για να επιλυθούν οι συγκρούσεις των παράλληλων εκδόσεων πρέπει να δημιουργηθούν νέα χρώματα. Ωστόσο, για να μπορέσουμε να συγκρίνουμε τα συστήματα πρέπει να ποσοτικοποιήσουμε αυτή την τάση. Για να το κάνουμε αυτό χρησιμοποιούμε την τυπική απόκλιση των χρωμάτων σε όλες τις εκδόσεις και τα αποτελέσματα φαίνονται στον πίνακα 3.11. Ο SS έχει STD 0 ή το χαμηλότερο STD μεταξύ των άλλων σε 18 από τις 26 εισόδους, ο IS σε 9, ο IIS σε 9, ο HTM-RCU σε 6, ο FineGrain σε 7, ο EE σε 8, ο VE-1 σε 8, ο VE-1.5 σε 7, ο VE-1.8 σε 9, ο VE-2 σε 8, ο VE-3 σε 7 και ο VE-4 σε 7, που σημαίνει ότι ο SS είναι το καλύτερο σχήμα όσον αφορά τον αριθμό των χρωμάτων που παράγουν. Επιστρέφοντας στους πίνακες 3.7-3.10, βλέπουμε ότι σε ορισμένες περιπτώσεις τα σχήματα κατάφεραν να παράγουν λιγότερα χρώματα από το σειριακό. Στον πίνακα 3.12 μπορούμε να δούμε σε πόσες εισόδους κάθε σχήμα κατάφερε να διατηρήσει τον ίδιο αριθμό χρωμάτων με το σειριακό ή τον μείωσε στην πραγματικότητα. Τα αποτελέσματα έχουν την ίδια αναλογία με τα προηγούμενα, καθώς σε κάθε λειτουργία το SS ξεπερνάει όλα τα άλλα σχήματα. Ένα ενδιαφέρον σημείο όμως αφορά το HTM-RCU και τις βελτιώσεις του όπου βλέπουμε ότι όλες οι παραλλαγές ξεπερνούν το αρχικό με καλύτερο το σχήμα VE-1 το οποίο είναι πολύ κοντά στο SS.

Graph	**SS**					**IS**					**IIS**				
Rmat1-1	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Rmat1-2	29	29	30	30	29	29	30	30	30	29	29	29	29	30	29
Rmat1-3	131	133	136	143	148	130	132	133	148	153	128	136	138	147	152
Rmat2-1	21	21	21	21	21	8	8	8	8	8	21	21	21	21	21
Rmat2-2	46	45	46	46	46	45	46	46	45	46	45	46	46	47	47
Rmat2-3	205	218	220	233	250	210	213	216	232	256	208	219	227	236	248
Rmat3-1	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
Rmat3-2	72	72	73	73	74	73	73	73	74	74	73	73	73	74	74
Rmat3-3	336	349	367	373	400	340	348	366	385	410	345	354	367	385	405
Rmat4-1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Rmat4-2	19	20	20	20	20	19	19	20	20	20	19	19	19	20	20
Rmat4-3	79	84	83	89	90	80	83	84	89	96	82	80	84	90	94
Real-1	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248
Real-2	116	119	119	119	127	116	116	119	123	129	119	117	117	126	129
Real-3	8	8	8	8	8	9	9	9	9	8	9	9	9	9	9
Real-4	6849	6849	6849	6849	6849	130	132	142	148	153	6848	6849	6855	6851	6865
Real-5	11	10	10	10	11	11	10	10	10	10	11	10	10	11	10
Real-6	6	6	6	6	6	2	2	2	2	2	6	6	6	6	6
Real-7	10	10	10	10	10	31	31	32	31	34	10	10	10	10	10
Real-8	20	24	28	30	53	20	22	28	33	47	22	23	28	37	49
Real-9	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Real-10	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Real-11	39	39	39	39	39	2	2	2	2	2	40	45	46	47	48
Real-12	31	31	31	31	31	31	31	32	32	33	31	31	32	31	34
Real-13	135	136	135	136	137	141	140	141	146	153	138	139	139	141	153
Real-14	1507	1507	1507	1507	1507	1512	1510	1524	1510	1520	1512	1510	1524	1510	1520

3.7: *Number of Colors: SS, IS, IIS*

Graph	HTM-RCU					FineGrain					EE				
Rmat1-1	14	14	14	15	14	14	14	14	14	14	14	14	14	14	14
Rmat1-2	30	30	31	31	33	31	30	31	32	32	29	29	29	29	30
Rmat1-3	128	125	128	130	133	127	126	126	129	132	126	125	129	130	122
Rmat2-1	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
Rmat2-2	47	45	47	48	50	47	46	48	49	49	45	44	45	47	46
Rmat2-3	208	206	208	211	216	208	203	202	206	209	205	202	204	204	194
Rmat3-1	32	32	32	32	32	32	32	32	32	32	31	32	32	32	32
Rmat3-2	76	73	76	78	80	75	74	75	78	79	71	71	70	73	73
Rmat3-3	340	334	341	345	345	338	331	330	332	339	330	325	328	326	313
Rmat4-1	10	10	10	10	10	10	10	10	10	11	10	10	10	10	10
Rmat4-2	20	20	20	21	21	20	20	20	21	22	20	19	19	20	20
Rmat4-3	79	78	81	82	84	80	78	80	81	81	78	78	82	82	79
Real-1	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248
Real-2	121	121	125	125	135	120	120	124	128	130	125	129	121	128	125
Real-3	9	10	10	9	9	9	10	9	9	9	10	9	10	9	9
Real-4	6849	6849	6849	6848	6849	6849	6849	6849	6848	6848	6848	6849	6848	6848	6848
Real-5	11	11	11	10	11	11	11	11	11	10	11	10	12	11	11
Real-6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Real-7	8	9	7	5	12	7	8	8	6	9	8	9	10	9	12
Real-8	20	26	29	27	32	19	21	21	21	21	23	23	26	32	27
Real-9	6	3	6	5	6	7	3	6	7	5	7	3	5	7	7
Real-10	4	3	6	6	6	5	3	6	7	6	6	3	6	7	6
Real-11	45	45	48	48	48	45	45	48	48	48	45	45	45	45	48
Real-12	30	31	33	34	34	30	32	33	36	35	34	34	32	32	34
Real-13	136	133	128	131	134	135	135	128	127	129	129	125	125	126	124
Real-14	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507

3.8: *Number of Colors: HTM-RCU, FineGrain, EE*

Graph	VE-1					VE-1.5					VE-1.8				
Rmat1-1	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Rmat1-2	29	29	29	29	29	29	29	29	30	30	29	29	29	30	30
Rmat1-3	125	121	122	122	120	125	120	129	126	121	124	124	126	128	122
Rmat2-1	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
Rmat2-2	44	45	45	44	45	45	45	44	46	46	45	45	44	46	45
Rmat2-3	202	199	197	192	193	200	198	205	204	192	203	201	206	202	193
Rmat3-1	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
Rmat3-2	70	70	71	73	72	72	71	71	74	72	71	71	70	74	73
Rmat3-3	329	328	317	318	309	330	327	327	319	310	331	328	333	321	311
Rmat4-1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Rmat4-2	19	19	20	20	19	19	20	20	20	21	19	20	19	21	20
Rmat4-3	78	77	76	77	76	78	75	80	81	78	79	79	82	81	79
Real-1	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248
Real-2	123	129	129	127	133	121	129	121	127	126	123	129	122	128	124
Real-3	9	9	9	9	10	10	9	10	9	9	10	9	10	9	9
Real-4	6849	6849	6849	6848	6849	0	0	6848	6848	6848	6848	6849	6848	6848	6848
Real-5	10	11	12	11	11	10	10	11	11	11	11	11	11	11	11
Real-6	6	6	7	6	6	6	6	6	6	6	6	6	6	6	6
Real-7	9	7	7	7	10	10	11	10	10	12	7	8	10	8	11
Real-8	19	22	29	31	27	23	23	25	31	30	22	23	26	34	28
Real-9	4	3	5	6	7	6	3	5	7	7	7	3	6	6	7
Real-10	6	3	5	7	6	7	3	7	7	7	6	3	7	7	7
Real-11	45	45	45	48	48	45	45	45	45	48	45	45	45	45	48
Real-12	30	32	32	34	36	34	0	34	34	33	34	34	34	33	34
Real-13	130	126	127	127	124	127	126	124	128	122	129	127	128	127	123
Real-14	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507

3.9: *Number of Colors: VE-1, VE-1.5, VE-1.8*

Graph	VE-2					VE-3					VE-4				
Rmat1-1	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Rmat1-2	29	29	29	30	31	29	29	28	29	30	29	29	28	29	30
Rmat1-3	126	124	128	128	122	127	125	130	131	123	127	125	130	131	123
Rmat2-1	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
Rmat2-2	45	44	44	46	46	45	45	44	47	46	45	45	44	47	46
Rmat2-3	204	202	205	202	193	201	202	205	205	193	201	202	205	205	193
Rmat3-1	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
Rmat3-2	71	72	70	74	72	72	71	70	73	73	72	71	70	73	73
Rmat3-3	330	330	333	322	308	331	329	329	326	314	331	329	329	326	314
Rmat4-1	10	10	10	10	10	10	10	10	10	11	10	10	10	10	11
Rmat4-2	19	20	19	20	20	19	20	20	20	20	19	20	20	20	20
Rmat4-3	78	78	81	80	78	79	78	81	81	80	79	78	81	81	80
Real-1	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248	3248
Real-2	123	129	121	128	125	125	130	121	129	125	125	130	121	129	125
Real-3	10	9	10	9	9	10	9	10	9	9	10	9	10	9	9
Real-4	6848	6849	6848	6848	6848	6848	6849	6848	6848	6848	6848	6849	6848	6848	6848
Real-5	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
Real-6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Real-7	7	8	10	14	14	9	11	9	9	12	9	11	9	9	12
Real-8	21	23	25	34	28	22	24	31	30	26	22	24	31	30	26
Real-9	6	3	5	6	7	7	3	5	7	7	7	3	5	7	7
Real-10	6	3	7	7	7	6	3	7	8	7	6	3	7	8	7
Real-11	45	45	45	45	48	45	45	45	45	48	45	45	45	45	48
Real-12	36	35	34	33	33	34	35	33	33	36	34	35	33	33	36
Real-13	128	127	125	125	124	130	128	123	127	124	130	128	123	127	124
Real-14	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507	1507

3.10: *Number of Colors: VE-2, VE-3, VE-4*

Graph	SS	IS	IIS	HTM	FineGrain	EE	VE-1	VE-1.5	VE-1.8	VE-2	VE-3	VE-4
Rmat1-1	**0,00**	**0,00**	**0,00**	0,41	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Rmat1-2	0,55	11,28	0,52	1,17	0,89	0,52	**0,41**	0,55	0,55	0,82	0,75	0,75
Rmat1-3	8,89	55,96	10,98	3,54	3,06	3,19	**1,72**	3,35	2,17	2,56	3,45	3,45
Rmat2-1	**0,00**	2,86	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Rmat2-2	**0,52**	17,72	0,89	1,90	1,63	1,03	0,52	0,75	0,63	0,89	1,03	1,03
Rmat2-3	18,17	92,09	17,52	5,01	3,31	**4,03**	4,13	4,69	4,34	4,26	4,40	4,40
Rmat3-1	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	0,41	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Rmat3-2	0,82	**0,75**	0,75	2,99	2,59	1,21	1,21	1,10	1,47	1,33	1,17	1,17
Rmat3-3	25,26	29,30	26,40	4,96	**3,74**	7,13	9,35	8,64	8,85	9,85	6,97	6,97
Rmat4-1	**0,00**	**0,00**	**0,00**	**0,00**	0,41	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	0,41	0,41
Rmat4-2	**0,41**	0,52	0,55	0,52	0,84	0,52	0,55	0,63	0,75	0,52	0,41	0,41
Rmat4-3	5,20	6,79	6,38	2,64	1,64	2,16	**0,75**	2,14	1,76	1,51	1,63	1,63
Real-1	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Real-2	**4,03**	5,27	5,47	6,40	5,33	4,82	5,95	4,84	4,68	4,80	5,20	5,20
Real-3	**0,00**	0,52	0,41	0,75	0,63	0,75	0,63	0,75	0,75	0,75	0,75	0,75
Real-4	**0,00**	11,54	6,46	0,41	0,52	0,52	0,41	0,55	0,52	0,52	0,52	0,52
Real-5	0,55	0,52	0,55	0,41	0,41	0,63	0,63	0,52	**0,00**	**0,00**	**0,00**	**0,00**
Real-6	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	0,41	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Real-7	**0,00**	1,21	**0,00**	2,43	1,41	1,37	1,51	0,84	1,55	2,95	1,26	1,26
Real-8	12,52	10,65	11,38	5,09	9,36	**4,43**	5,21	4,58	5,28	5,40	4,63	4,63
Real-9	**0,00**	**0,00**	**0,00**	1,75	2,10	2,23	1,87	2,10	2,14	1,94	2,23	2,23
Real-10	**0,00**	**0,00**	**0,00**	1,76	1,94	2,00	1,94	2,35	2,25	2,25	2,43	2,43
Real-11	**0,00**	**0,00**	3,76	3,51	3,51	2,95	3,29	2,95	2,95	2,95	2,95	2,95
Real-12	**0,00**	0,82	1,21	1,72	2,32	1,33	2,17	1,21	1,21	1,75	1,75	1,75
Real-13	**0,75**	5,91	6,10	3,10	4,08	4,51	4,23	4,83	4,27	4,42	4,69	4,69
Real-14	**0,00**	6,65	6,65	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**

3.11: *Standard Deviation of Number of Colors*

Graph	**4**	**7**	**14**	**28**	**56**
SS	20	20	18	18	18
IS	17	15	12	14	12
IIS	15	14	12	13	12
HTM-RCU	14	15	13	10	11
FineGrain	14	14	13	13	11
EE	16	17	15	14	15
VE-1	19	19	17	18	19
VE-1.5	17	18	16	14	15
VE-1.8	16	17	16	13	16
VE-2	16	17	16	14	15
VE-3	17	16	16	13	14
VE-4	16	15	15	13	13

3.12: *Per Thread*

Κεφάλαιο 4

Αλγόριθμοι Ισορροπημένου Χρωματισμού Γράφων

Όπως έχουμε ήδη αναφέρει, το πρόβλημα του χρωματισμού χρησιμοποιείται από πολλές εφαρμογές για τον παραλληλισμό των δεδομένων τους. Οι αλγόριθμοι που είδαμε στο κεφαλαίου 3, προσπαθούν να παράξουν έναν χρωματισμό με όσο το δυνατόν λιγότερα χρώματα χρησιμοποιώντας την ευριστική First Fit. Αυτό σημαίνει πως ένας κόμβος παίρνει το πρώτο διαθέσιμο χρώμα. Έτσι, οδηγούμαστε σε καταστάσεις όπου τα πρώτα χρώματα έχουν τους περισσότερους κόμβους. Αυτό μπορεί να οδηγήσει σε υποχρησιμοποίηση των πόρων, καθώς ορισμένα νήματα μπορεί να έχουν μεγαλύτερο φόρτο εργασίας από άλλα. Για την επίλυση αυτού του προβλήματος στο [14] έχουν προταθεί 4 αλγόριθμοι που παράγουν ισορροπημένο χρωματισμό, δηλαδή προσπαθούν να χρωματίσουν τον γράφο έτσι ώστε κανένας γείτονας να μην ανοίκει στο ίδιο χρώμα προσπαθώντας ταυτόχρονα να βάλουν σε κάθε χρώμα τον ίδιο αριθμό κόμβων. Φυσικά, επειδή το πρώτο μας κριτήριο είναι η εξάρτηση των κόμβων, υπάρχει μία χαλάρωση αναφορικά με τον αριθμό των κόμβων σε κάθε χρώμα. Έτσι, πέρα από το κριτήριο του χρόνου εκτέλεσης, οι αλγόριθμοι του κεφαλαίου αυτού αναπτύχθηκαν με γνώμονα του πόσο καλή ισορροπία μπορούν να δημιουργήσουν.

Οι αλγόριθμοι που θα εξετάσουμε είναι οι CLU, VFF, Scheduled και Recoloring οι οποίοι αποτελούνται από 2 κύριες φάσεις. Η πρώτη φάση είναι ένας αρχικός χρωματισμός που αγνοεί την ισορροπία, κατά την οποία παράγεται ο αριθμός των χρωμάτων. Η δεύτερη φάση είναι η εξισορρόπηση, κατά την οποία με βάση τον αρχικό χρωματισμό προσπαθεί να εξισορροπήσει τον αριθμό των κορυφών μεταξύ των χρωμάτων.

Για τους αλγορίθμους αυτούς, θα χρησιμοποιήσουμε τους ακόλουθους όρους, C είναι ο αριθμός των χρωμάτων που παράγονται από τον αρχικό χρωματισμό, **γ** είναι το άνω όριο των κορυφών μέσα σε ένα χρώμα που δίνεται από τη σχέση $\gamma = \frac{V}{C}$ όπου V είναι ο αριθμός των κορυφών, Least Used είναι η ευριστική που χρησιμοποιεί το λιγότερο χρησιμοποιούμενο επιτρεπτό χρώμα ενώ First Fit χρησιμοποιεί το μικρότερο επιτρεπτό χρώμα. Για τους δείκτες χρώματος θα χρησιμοποιούνται εναλλάξ οι όροι χρώμα και κλάση χρώματος. Επιπροσθέτως τα χρώματα χωρίζονται σε under-full και over-full υποδηλώνοντας χρώματα με κορυφές λιγότερες και περισσότερες από το γ αντίστοιχα.

Οι CLU, VFF και Scheduled, ακολουθούν την ίδια ιδέα. Μετά τον αρχικό χρωματισμό κατηγοριοποιούν τις χρωματικές κλάσεις σε under full / over full και προσπαθούν να τις φέρουν σε ισορροπία μετακινώντας κορυφές από το ένα σύνολο στο άλλο. Πέρα από αυτό,

υπάρχουν κάποιες διαφορές στον τρόπο με τον οποίο επεξεργάζονται τις κορυφές καθώς και στην ευρηστική για την επιλογή του επιτρεπόμενου χρώματος. Πιο συγκεκριμένα, ο CLU είναι χρωματοκεντρικός, δηλαδή επεξεργάζεται τις κορυφές ανά κλάση χρώματος ενώ η ευρηστική είναι η Least Used. Ο VFF είναι κορυφοκεντρικός αλγόριθμος, δηλαδή επεξεργάζεται τις κορυφές μία προς μία και η ευρηστική είναι η First Fit. Όσον αφορά τον αλγόριθμο Scheduled, χρησιμοποιεί την ευρηστική First Fit ενώ επεξεργάζεται τις κορυφές με προκαθορισμένο τρόπο βασιζόμενος στο χρώμα κάθε κορυφής.Ο τελευταίος αλγόριθμος, ο Recoloring, μετά τον αρχικό χρωματισμό χρωματίζει ξανά και ξανά τον γράφο μέχρι να υπάρξει ισορροπία.

4.1 CLU

Αυτός ο αλγόριθμος μετά τον αρχικό χρωματισμό(γραμμή 2), δημιουργεί το σύνολο Q που περιέχει τα under full χρώματα (γραμμή 3). Για κάθε χρώμα j στο Q (γραμμή 4) και για κάθε κορυφή u στο j (γραμμή 6) αναζητά ένα under full χρώμα k (γραμμή 7) στο οποίο θα μετακινηθεί. Εάν το k είναι έγκυρο χρώμα (γραμμή 8), η μετακίνηση πραγματοποιείται (γραμμή 9) και η κλάση του k ενημερώνεται (γραμμή 10). Το βασικό σημείο αυτού του αλγορίθμου είναι ότι επεξεργάζεται τις κορυφές με βάση την κλάση χρώματος όπου ανοίκουν.

μ 4.1: *CLU*

```
1:  Input : G(V, E)
2:  Phase 1: initial Coloring
3:  Q = set of over-full colors
4:  for j ∈ Q do                                   ▷ beginning of balancing phase
5:      V(j) = set of vertices with color j
6:      for u ∈ V(j) in parallel do
7:          k = index of least used under-full color that is permissible to u   ▷ LU
8:          if k exists then
9:              color[u] ← k
10:             update size of color k                          ▷ Synchronized step
11:         end if
12:     end for
13: end for
```

4.2 VFF

Αυτός ο αλγόριθμος μετά τον αρχικό χρωματισμό (γραμμή 2), δημιουργεί το σύνολο U που περιέχει τις κορυφές των under full χρωμάτων και για κάθε κορυφή u του U (γραμμή 5) βρίσκει το πρώτounder full χρώμα k που είναι επιτρεπτό για το u (γραμμή 6) και αν είναι έγκυρο (γραμμή 7) ενημερώνει το χρώμα του u (γραμμή 8) και το χρώμα του k (γραμμή 9). Επειδή ο αλγόριθμος αυτός επεξεργάζεται παράλληλα τις κορυφές με βάση τον δείκτη τους, κατά τη μεταφορά από μια χρωματική κλάση σε μια άλλη μπορεί να προκύψουν ασυνέπειες,

συνεπώς ακολουθεί η φάση ανίχνευσης συγκρούσεων (γραμμές 10-14) όπου αν υπάρχουν συγκρούσεις (γραμμή 14) η όλη διαδικασία πραγματοποιείται ξανά.

μ 4.2: *VFF*

```
1: Input : G(V, E)
2: Phase 1: initial Coloring
3: U = set of vertices from over-full colors
4: while U ≠ 0 do                                   ▷ beginning of balancing phase
5:     for u ∈ U in parallel do
6:         k = smallest index of under-full color that is permissible to u   ▷ FF
7:         if k exists then
8:             color[u] ← k
9:             update size of color k                ▷ Synchronized step
10:        end if
11:    end for
12:    R ← ∅
13:    for u ∈ U in parallel do
14:        for w ∈ adj(u) do
15:            if color[w] = color[u] and u > w then
16:                R ← R ⋃ u
17:            end if
18:        end for
19:    end for
20: end while
21: U ← R
```

4.3 Scheduled

Ο αρχικός χρωματισμός με τη χρήση του FF παράγει έναν χρωματισμό έτσι ώστε τα μικρότερα χρώματα να έχουν την πλειοψηφία των κορυφών ενώ τα μεγαλύτερα χρώματα να έχουν λίγα. Αυτή η παρατήρηση οδήγησε στο συμπέρασμα ότι μπορούμε να προκαθορίσουμε τον τρόπο με τον οποίο θα κάνουμε μετακινήσεις από over full σε under full. Συγκεκριμένα, προσπαθούμε να μετακινήσουμε κορυφές από τα πρώτα over full χρώματα στα τελευταία under full χρώματα. Για το λόγο αυτό υπάρχουν δύο σύνολα, Q_o και Q_u που αντιπροσωπεύουν τα over full και under full χρώματα αντίστοιχα. Το Q_o είναι σε αύξουσα σειρά και το Q_u σε φθίνουσα σειρά. Για κάθε over full χρώμα (γραμμή 6) εξάγουμε ένα υποσύνολο κορυφών V'(j) (γραμμή 8) έτσι ώστε το j χρώμα να είναι σε ισορροπία. Η επιλογή των κορυφών εδώ είναι αυθαίρετη χωρίς καμία προϋπόθεση. Στη συνέχεια, για κάθε under full χρώμα (γραμμή 9), εξάγουμε από το V'(j) (γραμμή 10) ένα υποσύνολο κορυφών V_k τέτοιο ώστε το χρώμα k να γίνει ισορροπημένο και αποθηκεύουμε την πιθανή κίνηση στη λίστα L(γραμμή 11). Επίσης, εδώ η επιλογή του V_k είναι αυθαίρετη. Είναι σημαντικό να επισημάνουμε εδώ, ότι δημιουργούμε μόνο τις πιθανές κινήσεις και δεν υπάρχει καθόλου μετακίνηση. Μετά από αυτή τη φάση, επεξεργαζόμαστε τις κορυφές (γραμμή 14) από κάθε V_k (γραμμή 13) ελέγχοντας αν μπορούν να μετακινηθούν στο χρώμα k (γραμμή 15) και αν ναι η κορυφή παίρνει το χρώμα k (γραμμή 16). Ένα άλλο βασικό σημείο είναι ότι μετά

την εξαγωγή του υποσυνόλου V_k οι κορυφές που το αποτελούν, είτε θα μετακινηθούν στο χρώμα k είτε θα παραμείνουν στο αρχικό τους χρώμα οδηγώντας σε μια ποιότητα ισορροπίας λιγότερο καλή σε σχέση με τους προηγούμενους αλγορίθμους όπως θα δούμε στο επόμενο κεφάλαιο.

μ 4.3: *Scheduled*

```
1: Input : G(V, E)
2: Phase 1: initial Coloring
3: Q_o = set of over-full colors in increasing order
4: Q_u = set of under-full colors in decreasing order
5: L = ∅ list of moves from over-full to under-full
6: for j ∈ Q_o do                                   ▷ for every over-full we extract
7:     V(j) = set of vertices with color j          ▷ a subset of vertices to be moved
8:     V'(j) = vertices from V(j) such that |V'(j)| = |V(j)| − γ   ▷ in an under-full
9:     for k ∈ Q_u do
10:        V_k = a subset of V'(j) that can be moved from j to k
11:        L ← L ⋃ V_k
12:        V'(j) ← V'(j) ⧵ V_k
13:    end for
14: end for
15: for V_k ∈ L do
16:    for u ∈ V_k in parallel do
17:        if k is permissible to u then
18:            color[u] ← k
19:        end if
20:    end for
21: end for
```

4.4 Recoloring

Σε αυτόν τον αλγόριθμο δημιουργούμε ένα σύνολο U (γραμμή 4) που περιέχει τις κορυφές σε φθίνουσα σειρά με βάση την κατηγορία χρώματος που προέκυψε από τον αρχικό χρωματισμό. Μετά από αυτό, επεξεργαζόμαστε τις κορυφές στο U χρωματίζοντας εκ νέου κάθε κορυφή. Το βασικό σημείο εδώ είναι ότι ο επαναχρωματισμός γίνεται μόνο μία φορά στην πρώτη επανάληψη της while (γραμμή 6) και μετά προσπαθούμε να επιλύσουμε τυχόν συγκρούσεις που προέκυψαν σε αυτόν τον επαναχρωματισμό. Ο λόγος για την επεξεργασία των κορυφών με αυτόν τον τρόπο είναι ότι τα τελευταία χρώματα περιέχουν είτε τις τελευταίες κορυφές του γράφου είτε τις κορυφές με πολλούς γείτονες, οπότε η ταχύτερη ταξινόμηση αυτών των κορυφών μπορεί να οδηγήσει σε λιγότερες συγκρούσεις, άρα σε λιγότερες επαναλήψεις.

μ 4.4: *Recoloring*

1: *Input* : $G(V, E)$
2: *Phase 1: initial Coloring*
3: $V(j)$ = *set of vertices with color* j
4: U = *ordered set of* $V(j)$ *in decreasing order of* j: $[V(C), V(C-1)...V(1)]$
5: $color[i] = 0$ *for* $i = 1, ...C$
6: **while** $U \neq \emptyset$ **do** ▷ fresh coloring
7: **for** $u \in U$ **in parallel do**
8: $color[u] \leftarrow$ *smallest permissible color k such that* $color[k] < \gamma$ ▷ FF
9: $color[k] \leftarrow color[k] + 1$
10: **end for**
11: $R \leftarrow \emptyset$
12: **for** $u \in U$ **in parallel do** ▷ Detect conflicts
13: **for** $w \in adj(u)$ **do**
14: **if** $color[w] = color[u]$ and $u > w$ **then**
15: $R \leftarrow R \bigcup u$
16: **end if**
17: **end for**
18: **end for**
19: $U \leftarrow R$
20: **end while**

Κεφάλαιο 5

Αλγόριθμοι Ισοροπημένου Χρωματισμού Γράφων: Ανάλυση και Αξιολόγηση

Όπως και στο κεφάλαιο 3 θα ακολουθήσουμε και εδώ την ίδια δομή. Αρχικά, θα δούμε εάν υπάρχει όφελος από τη χρήση πολλαπλών νημάτων. Φυσικά εδώ δεν υπάρχει ισοδύναμος αλγόριθμος του άπληστου, οπότε η επιτάχυνση θα είναι σε σχέση με τη σειριακή έκδοση του εκάστοτε αλγορίθμου. Στη συνέχεια, θα συγκρίνουμε τους αλγορίθμους χρησιμοποιώντας την επιτάχυνση και τέλος θα αξιολογήσουμε την ικανότητά τους όσον αφορά την ποιότητα της ισορροπίας που επιτυγχάνουν.

5.1 Μεθοδολογία

Για τα πειράματά μας χρησιμοποιήσαμε ένα σύνολο 26 γράφων 5.1. 12 παραχθέντες (Rmat−××), που στο εξής θα αναφέρονται ως συνθετικοί, και 14 πραγματικούς (Real−××).Για τους συνθετικούς γράφους έχουμε την εξής παραμετροποίηση:

- Rmat(1|2|3|4)-1: A = 0.25 B = 0.25 C = 0.25 D = 0.25
- Rmat(1|2|3|4)-2: A = 0.45 B = 0.15 C = 0.15 D = 0.25
- Rmat(1|2|3|4)-3: A = 0.55 B = 0.15 C = 0.15 D = 0.15

Συνθετικοί γράφοι με ίσα A,B,C και D είναι πιο κανονικοποιημένοι και κατ' επέκταση πιο εύκολοι στους υπολογισμούς ενώ όσο διαφοροποιούμε τις παραμέτρους αυτές γίνονται όλο και πιο ακανόνιστοι δηλαδή δύσκολοι. Όσον αφορά τους πραγματικούς γράφους, είναι όλοι ακανόνιστοι.

Για την αξιολόγηση των αλγορίθμων, καθώς και για τη σύγκρισή τους, χρησιμοποιούμε τη μετρική επιτάχυνσης (speedup) η οποία δίνεται από την ακόλουθη έκφραση

$$Speedup = \frac{T_s}{T_p}$$

όπου T_s είναι ο χρόνος εκτέλεσης της σειριακής έκδοσης και T_p της παράλληλης. Η επιτάχυνση δείχνει πόσες φορές η παράλληλη έκδοση ενός αλγορίθμου είναι ταχύτερη από τη σειριακή. Μια άλλη μετρική για σύγκριση είναι ο αριθμός των χρωμάτων που παράγει κάθε σχήμα. Πέρα από τις μετρικές που προσανατολίζονται στο χρόνο για να συγκρίνουμε τα γραφήματα με βάση τα χαρακτηριστικά τους, χρησιμοποιούμε τη μετρική τυπική απόκλιση

Graph	Source	GraphID	V	E	MaxDeg	AvgDeg	STD
rmat-10Mx100M	Generated	Rmat1-1	10M	100M	103	19,9	10,3
rmat-10Mx100M	Generated	Rmat1-2	10M	100M	2193	19,9	27,8
rmat-10Mx100M	Generated	Rmat1-3	10M	100M	46303	19,9	107,7
rmat-10Mx200M	Generated	Rmat2-1	10M	200M	183	39,9	19,6
rmat-10Mx200M	Generated	Rmat2-2	10M	200M	4381	39,9	55,2
rmat-10Mx200M	Generated	Rmat2-3	10M	200M	83878	39,7	211,1
rmat-10Mx400M	Generated	Rmat3-1	10M	400M	339	79,9	38,2
rmat-10Mx400M	Generated	Rmat3-2	10M	400M	8819	79,9	110
rmat-10Mx400M	Generated	Rmat3-3	10M	400M	147359	79,1	410
rmat-10Mx50M	Generated	Rmat4-1	10M	50M	54	9,9	5,6
rmat-10Mx50M	Generated	Rmat4-2	10M	50M	1127	9,9	14
rmat-10Mx50M	Generated	Rmat4-3	10M	50M	24861	9,9	54,7
arabic-2005	[18]	Real-1	22M	639M	9905	28,1	78,8
com-Orkut	[18]	Real-2	3M	117M	33313	76,2	154,8
FullChip	[18]	Real-3	2.9M	26M	2312481	8,9	1806,8
indochina-2004	[18]	Real-4	7M	194M	6985	26,1	215,8
kmer_A2a	[18]	Real-5	170M	180M	40	2,1	0,6
kmer_V1r	[18]	Real-6	214M	232M	8	2,1	0,6
mawi_2015	[18]	Real-7	226M	240M	2107954	2,1	14016
mycielskian19	[18]	Real-8	0.3M	451M	196607	2296,9	4530
nlpkkt200	[18]	Real-9	27M	401M	28	27,6	2,4
nlpkkt240	[18]	Real-10	16M	232M	28	27,6	2,2
Queen_4147	[18]	Real-11	4M	166M	81	79,4	6,3
stokes	[18]	Real-12	11M	349M	720	30,5	41,4
sx-stackoverflow	[18]	Real-13	2M	36M	38148	13,9	137,8
webbase-2001	[18]	Real-14	118M	1019M	3841	8,6	26

5.1: *Graph Suite*

(std) η οποία μας δείνει πόσο διαφέρει ο βαθμός των κορυφών από το μέσο βαθμό του γράφου.

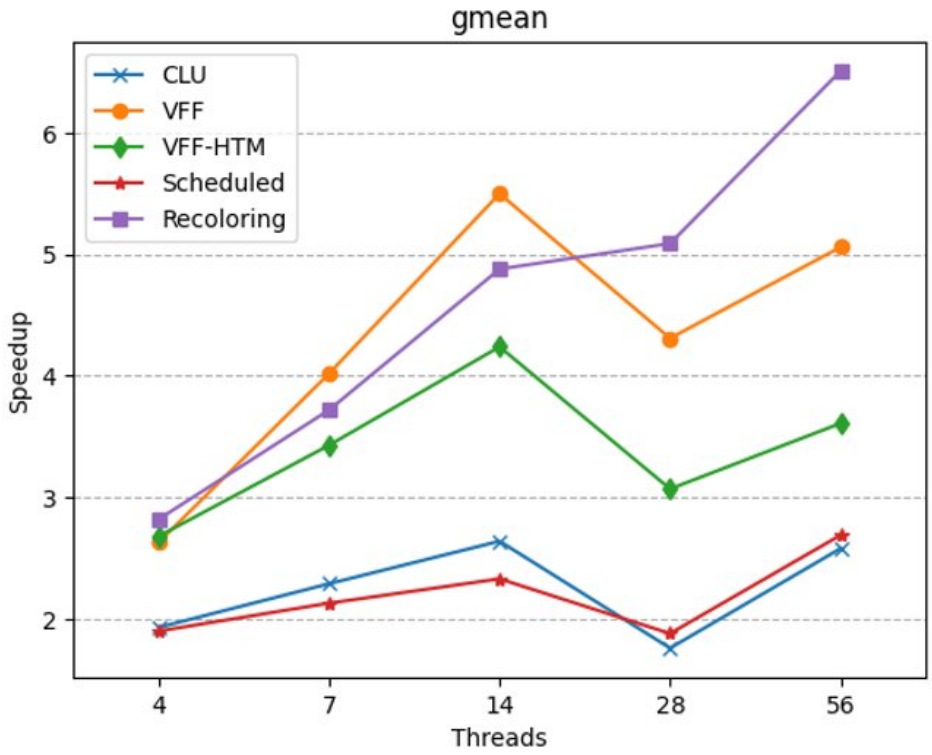
Τα πειράματα πραγματοποιήθηκαν στη μηχανή Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz η οποία έχει 2 υποδοχές με 1 cpu η κάθε μία και σε κάθε cpu υπάρχουν 28 νήματα, οπότε έχουμε στη διάθεσή μας 56 νήματα υλικού. Τέλος, πήραμε αποτελέσματα για 1, 4, 7, 14, 28 και 56 νήματα.

5.2 Κλιμακωσιμότητα

Στην εικόνα 5.1 βλέπουμε την επιτάχυνση που πετυχαίνει κάθε αλγόριθμος έναντι του εκάστοτε σειριακού. Αρχικά βλέπουμε πώς όλοι οι αλγόριθμοι ξεκινώντας από τα 4 νήματα και συνεχίζοντας με περισσότερα πετυχαίνουν επιταχύνσεις έναντι των σειριακών εκδόσεων τους και δεν υπάρχει ούτε μία περίπτωση στην οποία καποιος παράλληλος αλγόριθμος τρέχει πιο αργά απο τον σειρακό του. Συνεπώς η παραλληλοποίηση πετυχαίνει το στοίχημα της μείωσης του χρόνου εκτέλεσης. Στη συνέχεια παρατηρούμε πως από τα 4 έως τα 14 νήματα, έχουμε είτε γραμμική (VFF-HTM, VFF, Recoloring) είτε σχεδόν γραμμική (CLU, Scheduled) κλίση, αυτό έχει να κάνει με το γεγονός πως σε αυτές τις εκδοχές, τα νήματα ανήκουν στην ίδια cpu, συνεπώς η μεταφορά των δεδομένων και ο συγχρονισμός μεταξύ των νημάτων δεν είναι τόσο χρονοβόρος, από την άλλη βλέπουμε πώς μόλις μεταφερόμαστε στην έκδοση των 28 νημάτων, ενω εξακολουθούμε να έχουμε επιτάχυνση, αυτή μειώνεται αισθητά. Αυτό έχει να κάνει με το γεγονός πως πλέον δεν ανοίκουν όλα τα νήματα στην ίδια cpu και συνεπώς πέρα από τον συγχρονισμό των νημάτων χρειαζόμαστε και έναν επιπλέον χρόνο για τυχούσες μεταφορές δεδομένων μεταξύ νημάτων που επεξεργάζονται ίδιες κορυφές αλλά ανοίκουν σε διαφορετική cpu. Τέλος, όταν φτάνουμε στην έκδοση των 56 νημάτων, βλέπουμε πως η επιτάχυνση είναι καλύτερη από τα 28, γεγονός που δικαιολογείται απλώς από τη χρήση περισσότερων νημάτων.

Συμπερασματικά, η παραλληλοποίηση των αλγορίθμων βοηθάει σημαντικά στον χρόνο εκτέλεσης, μάλιστα, όσο αυξάνονται τα νήματα τόσο αυξάνεται και η επιτάχυνση και σημαντικό ρόλο παίζει το γεγονός εαν τα νήματα ανήκουν στην ίδια cpu ή όχι.

Μία διαφορά με την αντίστοιχη γραφική παράσταση που δείξαμε στο κεφάλαιο 3, είναι πως οι 5 γραφικές αυτές παραστάσεις δεν μπορούν να συγκριθούν καθώς δεν έχουν την ίδια βάση. Κάθε αλγόριθμος συγκρίνεται με τον αντίστοιχο σειριακό του, στο επόμενο κεφάλαιο όμως επιλέγουμε έναν αλγόριθμο σαν βάση ώστε να μπορέσουμε να τους συγκρίνουμε.



Εικόνα 5.1: *Speedup*

5.3 Σύγκριση Αλγορίθμων

Τώρα, θα συγκρίνουμε τα CLU, VFF, VFF_HTM, Scheduled και Recoloring ως προς την επιτάχυνσή τους. Για τη σύγκριση θα χρησιμοποιήσουμε τον CLU ως βάση, πράγμα που σημαίνει ότι θα συγκρίνουμε σε κάθε λειτουργία νήματος αν ένας αλγόριθμος είναι πιο αργός ή πιο γρήγορος από τον CLU και όχι από τη σειριακή έκδοση.

Για να δούμε τι μπορούμε να περιμένουμε από τα αποτελέσματα, θα ασχοληθούμε με τους αλγορίθμους και τις ιδέες πίσω από αυτούς:

- Το σχήμα CLU είναι μια επαναληπτική διαδικασία που επεξεργάζεται τις κορυφές ανά χρώμα. Επαναλαμβάνει μέχρι να μην υπάρχουν κινήσεις που πρέπει να γίνουν.

- Το σχήμα VFF/VFF_HTM είναι μια επαναληπτική διαδικασία που επεξεργάζεται τις κορυφές ανά δείκτη. Επαναλαμβάνει μέχρι να μην υπάρχουν κινήσεις που πρέπει να γίνουν.

- Το σχήμα Scheduled αρχικά βρίσκει τις πιθανές κινήσεις και στη συνέχεια είτε τις πραγματοποιεί είτε όχι.

- Το σχήμα επαναχρωματισμού είναι μια επαναληπτική διαδικασία που αρχικά επαναχρωματίζει το γράφο με βάση τις κατηγορίες χρωμάτων που προκύπτουν από τν αρχικό χρωματισμό και μετά επαναλαμβάνει μέχρι να υπάρξει ισορροπία.

Από τις παραπάνω παρατηρήσεις μπορούμε να πούμε ότι από τη μία έχουμε τα CLU, VFF, VFF_HTM και Recoloring που όλα επαναλαμβάνονται μέχρι να υπάρξει ισορροπία και από την άλλη έχουμε το σχήμα Scheduled το οποίο μπορεί να τελειώσει χωρίς να έχουν γίνει όλες οι πιθανές κινήσεις. Αυτό οδηγεί στο συμπέρασμα ότι το Scheduled μπορεί να τρέχει γρηγορότερα από τα άλλα σχήματα λόγω της χαλάρωσής του στην εξισορρόπηση και είναι ακριβώς αυτό που είδαμε στα πειράματα μας.

Στους πίνακες 5.2, 5.3, 5.4, 5.5, 5.6 και 5.7 βλέπουμε την επιτάχυνση σε σχέση με το CLU στις εκδόσεις σειριακό, 4, 7, 15, 28, 56 νημάτων αντίστοιχα. Βλέπουμε ότι σε όλους τους τρόπους, εκτός από τα 14 νήματα, στις περισσότερες περιπτώσεις το Scheduled ξεπερνάει όλα τα άλλα σχήματα. Συγκεκριμένα παρατηρούμε (5.8) ότι το Scheduled τρέχει ταχύτερα για πάνω από 20 εισόδους από τις συνολικές 26. Ένα ενδιαφέρον σημείο είναι ότι στη λειτουργία 14 νημάτων, το σχήμα Recoloring υπερέχει σε 18 εισόδους και με μεγάλη διαφορά από τα άλλα σχήματα. Αυτό οφείλεται στο συνδυασμό του προγραμματισμένου τρόπου επεξεργασίας των κορυφών και της επαναληπτικής λογικής. Ο επαναχρωματισμός των κορυφών με βάση την κατηγορία χρώματος τους σε φθίνουσα σειρά μπορεί να οδηγήσει στο να υπάρχουν λιγότερες συγκρούσεις και επομένως λιγότερες επαναλήψεις που πρέπει να γίνουν. Τα πειράματά μας έδειξαν ότι τα 14 νήματα είναι το καλύτερο σενάριο για το σχήμα Recoloring όπου ο συγχρονισμός με τη βοήθεια της προγραμματισμένης επεξεργασίας των κορυφών προκαλεί λίγες συγκρούσεις προκειμένου να επιτευχθούν καλύτερες επιταχύνσεις. Όσον αφορά τη σύγκριση CLU και VFF, η CLU υπερτερεί της VFF σε λειτουργίες με χαμηλό αριθμό νημάτων και καθώς αυξάνονται τα νήματα η VFF αρχίζει να υπερτερεί της CLU. Αυτή η συμπεριφορά δείχνει ότι οταν έχουμε πολλά νήματα είναι καλύτερο να επεξεργαζόμαστε τις κορυφές με βάση το δείκτη παρά με βάση την κατηγορία χρώματος. Αναφορικά με τα CLU, VFF και VFF_HTM βλέπουμε ότι η παραλλαγή HTM ξεπερνάει την CLU καθώς και την VFF σε όλα τα πειράμτα ακόμα και στα λίγα νήματα και οφείλεται στο συγχρονισμό που παρέχει το HTM.

Συμπερασματικά, ο αλγόριθμος Scheduled είναι ο καλύτερος αλγόριθμος για 1,4,7,28 και 56 νήματα, το Recoloring είναι ο καλύτερος αλγόριθμος για 14 νήματα και το VFF_HTM είναι καλύτερο από το CLU και το VFF σε όλες τις λειτουργίες.

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	0,46	1,39	**2,76**	0,33	Real-2	0,77	1,07	**2,47**	0,66
Rmat1-2	0,51	0,89	**1,67**	0,63	Real-3	0,46	0,64	**1,50**	0,86
Rmat1-3	0,69	1,01	**2,96**	0,88	Real-4	1,35	1,40	1,67	**3,43**
Rmat2-1	0,37	1,48	**3,11**	0,36	Real-5	0,82	0,70	**2,75**	0,88
Rmat2-2	0,44	0,87	**0,96**	0,48	Real-6	0,66	0,58	**0,81**	0,68
Rmat2-3	0,78	1,14	**2,54**	0,75	Real-7	0,77	**1,18**	0,63	0,95
Rmat3-1	0,33	1,87	**4,75**	0,25	Real-8	0,37	0,38	**2,31**	0,27
Rmat3-2	0,45	0,93	**2,61**	0,41	Real-9	0,29	0,93	**219,49**	0,10
Rmat3-3	0,87	1,25	**3,22**	0,98	Real-10	0,31	0,90	**213,20**	0,06
Rmat4-1	0,48	1,06	**1,29**	0,52	Real-11	0,76	1,08	**3,19**	0,41
Rmat4-2	0,54	0,81	**1,78**	0,82	Real-12	0,51	**0,93**	0,66	0,45
Rmat4-3	0,71	0,97	**3,56**	1,02	Real-13	0,75	0,98	**3,50**	1,08
Real-1	1,35	1,44	**15,76**	2,26	Real-14	1,22	1,25	**10,32**	2,52

5.2: *Speedup over CLU:Serial*

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	0,57	1,56	**2,43**	0,30	Real-2	0,73	1,14	**2,59**	0,55
Rmat1-2	0,73	1,10	**1,65**	0,45	Real-3	1,29	1,56	**2,98**	0,73
Rmat1-3	1,01	1,37	**1,77**	0,85	Real-4	1,73	**1,88**	1,34	1,72
Rmat2-1	0,48	1,77	**3,01**	0,25	Real-5	1,43	**1,51**	1,50	0,90
Rmat2-2	0,54	1,02	**1,32**	0,30	Real-6	1,31	1,27	**1,46**	0,64
Rmat2-3	0,89	1,41	**2,49**	0,66	Real-7	1,49	**2,07**	1,59	1,11
Rmat3-1	0,34	1,88	**3,67**	0,19	Real-8	0,38	0,38	**1,39**	0,36
Rmat3-2	0,53	1,13	**2,03**	0,36	Real-9	1,03	3,13	**281,29**	0,09
Rmat3-3	0,93	1,38	**3,72**	0,76	Real-10	1,34	4,22	**459,40**	0,12
Rmat4-1	0,21	1,36	**1,76**	0,34	Real-11	0,69	0,93	**4,01**	0,17
Rmat4-2	0,83	1,03	**1,41**	0,49	Real-12	0,60	0,81	**1,73**	0,37
Rmat4-3	0,99	1,22	**1,33**	0,66	Real-13	0,98	1,29	**2,07**	0,57
Real-1	1,47	1,61	**16,58**	1,52	Real-14	1,37	1,46	**6,45**	1,64

5.3: *Speedup over CLU:4 Threads*

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	0,76	1,80	**2,49**	0,33	Real-2	0,65	**1,18**	1,15	0,40
Rmat1-2	0,78	1,01	**1,15**	0,40	Real-3	1,64	1,53	**2,74**	0,86
Rmat1-3	1,07	1,23	**1,43**	0,65	Real-4	2,66	**2,83**	2,13	2,52
Rmat2-1	0,53	1,72	**2,54**	0,32	Real-5	**1,65**	1,32	1,35	0,87
Rmat2-2	0,61	1,02	**1,42**	0,37	Real-6	**1,54**	1,29	1,40	0,67
Rmat2-3	1,10	1,41	**2,10**	0,89	Real-7	2,09	**2,46**	1,82	1,26
Rmat3-1	0,38	1,98	**3,06**	0,25	Real-8	0,49	0,43	**3,62**	0,20
Rmat3-2	0,62	1,22	**1,99**	0,44	Real-9	2,68	8,22	**482,99**	0,13
Rmat3-3	1,09	1,37	**3,43**	0,87	Real-10	2,60	8,45	**608,24**	0,13
Rmat4-1	0,63	**1,38**	1,36	0,37	Real-11	0,84	0,64	**3,75**	0,22
Rmat4-2	1,09	1,10	**1,23**	0,54	Real-12	0,84	0,80	**1,53**	0,35
Rmat4-3	1,19	1,27	**1,31**	0,83	Real-13	1,06	1,25	**1,75**	0,96
Real-1	1,49	1,56	**11,42**	1,69	Real-14	1,37	1,41	**5,93**	2,13

5.4: *Speedup over CLU:7 Threads*

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	0,89	1,51	1,72	**2,19**	Real-2	0,83	1,04	**2,03**	0,79
Rmat1-2	1,08	1,11	1,45	**1,81**	Real-3	1,82	1,58	**2,78**	0,41
Rmat1-3	1,09	1,36	1,31	**1,99**	Real-4	3,72	3,78	3,45	**8,07**
Rmat2-1	0,79	1,82	**2,53**	2,51	Real-5	1,70	1,24	1,20	**24,08**
Rmat2-2	0,90	1,17	1,43	**2,58**	Real-6	1,37	1,32	1,21	**29,05**
Rmat2-3	1,15	1,40	1,36	**2,55**	Real-7	1,78	1,70	1,30	**31,41**
Rmat3-1	0,49	1,92	2,66	**3,02**	Real-8	0,51	0,35	**4,36**	0,48
Rmat3-2	0,77	1,25	1,82	**3,41**	Real-9	4,49	16,12	**522,72**	9,43
Rmat3-3	1,47	1,88	2,81	**3,37**	Real-10	2,50	9,27	**370,37**	16,22
Rmat4-1	0,96	1,60	**2,03**	1,56	Real-11	1,30	0,72	**5,77**	0,98
Rmat4-2	1,17	1,28	1,43	**1,45**	Real-12	0,80	0,83	0,92	**2,53**
Rmat4-3	1,33	1,48	1,38	**1,65**	Real-13	1,20	1,52	1,61	0,42
Real-1	1,69	1,65	8,67	**11,60**	Real-14	1,49	1,48	4,98	**30,89**

5.5: *Speedup over CLU:14 Threads*

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	1,30	1,82	**2,78**	0,62	Real-2	1,05	0,95	**1,86**	1,08
Rmat1-2	1,16	1,21	**1,64**	0,92	Real-3	1,11	1,05	**2,16**	0,71
Rmat1-3	1,20	1,19	**1,38**	1,04	Real-4	4,97	4,37	3,62	**5,27**
Rmat2-1	0,99	1,72	**2,83**	0,49	Real-5	**1,48**	1,31	1,41	0,94
Rmat2-2	0,93	1,07	**1,62**	0,72	Real-6	1,30	1,47	**1,58**	0,84
Rmat2-3	1,32	1,59	**1,81**	1,26	Real-7	1,37	**2,19**	1,53	1,23
Rmat3-1	0,84	2,31	**3,41**	0,51	Real-8	0,40	0,34	**4,65**	0,24
Rmat3-2	1,21	1,53	**2,18**	0,92	Real-9	11,24	55,50	**1886,65**	0,74
Rmat3-3	1,01	1,20	**2,00**	1,10	Real-10	7,89	45,45	**1214,67**	0,69
Rmat4-1	2,14	1,52	**2,19**	0,70	Real-11	1,17	0,43	**5,38**	0,54
Rmat4-2	1,38	1,27	**1,70**	1,04	Real-12	1,03	0,95	**1,93**	0,73
Rmat4-3	**1,39**	1,33	1,34	1,24	Real-13	1,34	1,28	**1,75**	1,42
Real-1	1,82	1,75	**8,27**	3,41	Real-14	1,37	1,29	**4,01**	2,13

5.6: *Speedup over CLU:28 Threads*

Graph	VFF	HTM	Sched	Recol	Graph	VFF	HTM	Sched	Recol
Rmat1-1	1,17	1,62	**3,59**	0,64	Real-2	0,82	0,82	**1,86**	0,86
Rmat1-2	1,09	1,19	**2,28**	0,94	Real-3	1,29	1,29	**5,25**	0,95
Rmat1-3	1,02	1,09	**1,86**	0,91	Real-4	2,35	2,28	0,00	**1,78**
Rmat2-1	1,11	2,27	**4,30**	0,75	Real-5	1,37	1,35	**1,97**	1,24
Rmat2-2	0,57	0,85	**1,49**	0,51	Real-6	1,38	1,34	**1,92**	0,94
Rmat2-3	0,99	1,08	**1,60**	0,88	Real-7	1,73	**2,14**	1,86	1,31
Rmat3-1	0,63	2,06	**3,66**	0,49	Real-8	0,30	0,20	**2,23**	0,18
Rmat3-2	0,65	0,96	**1,55**	0,57	Real-9	4,88	22,98	**701,64**	0,41
Rmat3-3	1,13	1,38	**2,33**	1,11	Real-10	3,90	10,72	**128,07**	0,35
Rmat4-1	0,93	1,39	**2,94**	0,68	Real-11	0,90	0,24	**3,69**	0,47
Rmat4-2	1,56	1,28	**2,57**	1,11	Real-12	0,57	0,78	**2,62**	0,78
Rmat4-3	1,27	1,05	**1,71**	1,01	Real-13	1,07	0,92	**1,84**	0,99
Real-1	1,87	1,71	**10,82**	2,99	Real-14	1,89	1,83	**7,16**	2,89

5.7: *Speedup over CLU:56*

Mode	CLU	Scheduled	VFF_HTM	Recoloring	VFF
Serial	3	21	1	1	0
4	0	23	3	0	0
7	0	20	4	0	2
14	0	8	0	18	0
28	0	22	1	1	2
56	0	24	1	1	0

5.8: *Number of inputs that a scheme outperforms*

5.4 Ποιότητα ισορροπίας

Όπως ήδη είπαμε για την ποιότητα της ισορροπίας θα χρησιμοποιήσουμε τη μετρική RSTD. Η ισορροπία είναι καλύτερη όσο πιο κοντά είναι αυτή η μετρική στο 0.00%. Από τα πειράματά μας διαπιστώσαμε ότι τα σχήματα VFF και CLU αποδίδουν καλύτερα από όλα τα άλλα σχήματα. Αυτό οφείλεται στις στρατηγικές τους- να κάνουν κινήσεις μέχρι να υπάρξει ισορροπία. Από την άλλη πλευρά, το επίπεδο ισορροπίας που επιτεύχθηκε ήταν χαμηλό με το σχήμα Scheduled, όπως είναι αναμενόμενο δεδομένης της στρατηγικής του. Η επανάληψη της διαδικασίας σταθερού αριθμού φορών είναι μια τεχνική για την ενίσχυση της απόδοσης της προγραμματισμένης στρατηγικής, αλλά το αντιστάθμισμα είναι ότι θα αυξήσει τον χρόνο εκτέλεσης. Αυτό το αντιστάθμισμα είναι επίσης παρόν στα σχήματα CLU/VFF, καθώς το Scheduled ξεπερνά όλους τους άλλους αλγορίθμους όσον αφορά τον χρόνο εκτέλεσης. Συμπερασματικά, το σχήμα που θα επιλέξουμε για μια πραγματική εφαρμογή εξαρτάται από το αν είναι πιο σημαντικός ο χρόνος εκτέλεσης ή η ποιότητα της ισορροπίας.

Γραφη	**ΎΛΥ**	**ΎΦΦ**	**ΎΦΦ_ΗΤΜ**	**Σχεδυλεδ**	**Ρεσολορινγ**	**Βεστ**
Ρματ1-1	0,00%	0,00%	3,92%	24,03%	13,36%	ΎΦΦ
Ρματ1-2	0,01%	0,02%	5,00%	27,93%	16,63%	ΎΛΥ
Ρματ1-3	0,22%	0,08%	4,31%	21,94%	26,59%	ΎΦΦ
Ρματ2-1	0,00%	0,00%	3,94%	23,34%	14,55%	ΎΦΦ
Ρματ2-2	0,01%	0,03%	5,78%	30,98%	14,34%	ΎΛΥ
Ρματ2-3	0,35%	0,15%	5,09%	25,15%	28,03%	ΎΦΦ
Ρματ3-1	0,60%	0,00%	3,72%	18,58%	2,95%	ΎΦΦ
Ρματ3-2	0,39%	1,97%	6,53%	34,36%	11,98%	ΎΛΥ
Ρματ3-3	0,50%	0,25%	5,97%	28,63%	26,13%	ΎΦΦ
Ρματ4-1	0,00%	0,00%	3,99%	19,94%	15,81%	ΎΦΦ
Ρματ4-2	0,00%	0,01%	4,22%	21,21%	16,59%	ΎΛΥ
Ρματ4-3	0,05%	0,04%	3,60%	19,28%	25,52%	ΎΦΦ
Ρεαλ-1	0,48%	0,72%	2,36%	6,05%	42,52%	ΎΛΥ
Ρεαλ-2	0,11%	0,70%	3,99%	20,69%	22,61%	ΎΛΥ
Ρεαλ-3	0,00%	0,02%	2,09%	53,82%	32,91%	ΎΛΥ
Ρεαλ-4	3,61%	2,92%	2,85%	0,01%	39,78%	Σχηεδυλεδ
Ρεαλ-5	0,00%	0,00%	0,94%	9,91%	15,32%	ΎΦΦ
Ρεαλ-6	0,00%	0,00%	2,02%	16,89%	6,80%	ΎΦΦ
Ρεαλ-7	43,85%	43,85%	44,11%	41,63%	23,70%	Ρεσολορινγ
Ρεαλ-8	19,41%	26,64%	41,66%	93,27%	22,68%	ΎΛΥ
Ρεαλ-9	1,48%	1,48%	1,48%	12,76%	57,62%	ΎΛΥ
Ρεαλ-10	1,23%	1,23%	1,23%	12,60%	57,84%	ΎΛΥ
Ρεαλ-11	0,05%	8,22%	5,47%	31,84%	12,49%	ΎΛΥ
Ρεαλ-12	0,00%	3,12%	3,86%	22,00%	16,00%	ΎΛΥ
Ρεαλ-13	0,43%	0,35%	4,98%	23,91%	36,03%	ΎΦΦ
Ρεαλ-14	0,18%	0,68%	0,95%	4,50%	45,89%	ΎΛΥ

Πίνακας 5.9: *Ρελατιε Σταυδαρδ Δειατιον*

Κεφάλαιο 6

Πραγματικές Εφαρμογές

Όπως είπαμε στην εισαγωγή, το πρόβλημα του χρωματισμού χρησιμοποιείται από διάφορες πραγματικές εφαρμογές για τον παραλληλισμό των δεδομένων τους. Σε αυτή την ενότητα διερευνούμε δύο εφαρμογές που χρησιμοποιούν χρωματισμό/εξισορρόπηση, τις αναλύουμε από την άποψη του χρόνου εκτέλεσης προσπαθώντας να συμπεράνουμε αν η ευρετική του χρωματισμού είναι χρήσιμη ή όχι. Αυτές οι εφαρμογές είναι η ΠαγεΡανκ και η ανίχνευση κοινοτήτων.

6.1 PageRank

Το PageRank που παρουσιάστηκε για πρώτη φορά το [7], είναι ένας αλγόριθμος που χρησιμοποιείται από τη μηχανή αναζήτησης της Google για την κατάταξη ιστοσελίδων. Τις κατατάσσει χρησιμοποιώντας την παρακάτω εξίσωση.

$$r(u) = \frac{1-f}{|V|} + \sum_{w \in adj(u)} (f \frac{r(w)}{d(w)})$$

Ο αλγόριθμος είναι επαναληπτικός, επαναλαμβάνοντας για έναν δεδομένο αριθμό L(γραμμή 3). Για κάθε επανάληψη διατρέχει το γράφο παράλληλα (γραμμή 5). Για κάθε κορυφή u υπολογίζει την τιμή $\frac{pr(u)}{d(u)}$ και την ωθεί σε κάθε κορυφή $w \in adj(u)$ (γραμμές 7). Τέλος, ενημερώνει την κατάταξη σελίδας του u (γραμμές 8-9).

Αλγοριθμος 6.1: *PageRank*

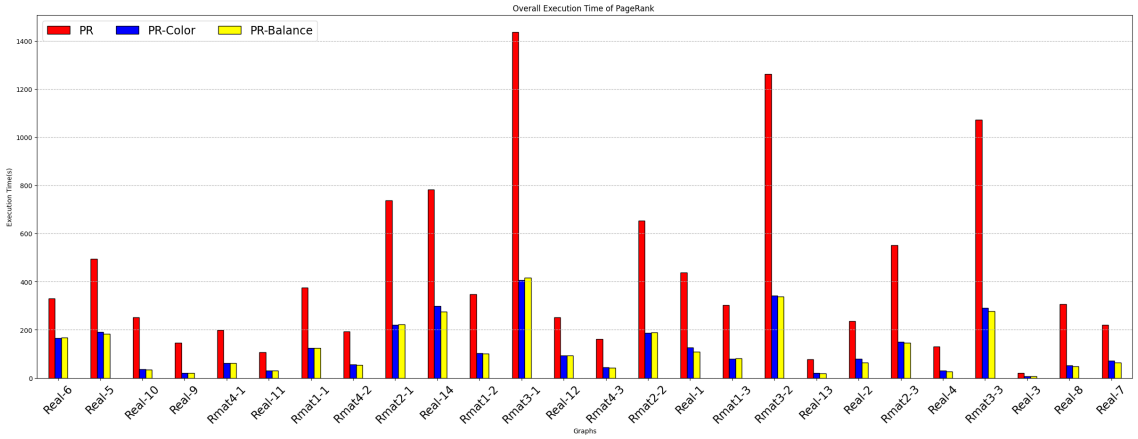
```
1:  Input : G(V, E), L, f
2:  pr[1...u] = [f...f]                          ▷ Init PR values
3:  for i ∈ range(L) do
4:      new_pr[1...n] = [0...0]
5:      for u ∈ V in parallel do
6:          for w ∈ adj(w) do
7:              new_pr(w) += f pr(v)/d(v)
8:          end for
9:          new_pr(u) += (1−f)/|V|
10:         pr(u) = new_pr(u)
11:     end for
12: end for
```

Στη γραμμή 7 βλέπουμε ότι υπάρχει εξάρτηση εγγραφής επειδή ενημερώνει κορυφές που μπορεί να επεξεργαστούν και από άλλα νήματα, επομένως πρέπει να αποφύγουμε τέτοιες περιπτώσεις. Η χρήση ατομικών πράξεων θα μπορούσε να είναι μια λύση αλλά οι τιμές είναι δεκαδικές και ατομικές πράξεις υπάρχουν μόνο για ακέραιους, οπότε η μόνη λύση είναι η χρήση κλειδωμάτων. Με το κλείδωμα του κρίσιμου τμήματος της γραμμής 7, προσθέτουμε μια σημαντική επιβάρυνση συγχρονισμού στο χρόνο εκτέλεσης, επομένως χρησιμοποιούμε ευρετικές μεθόδους χρωματισμού/εξισορρόπησης για τον παραλληλισμό των δεδομένων και την αποφυγή οποιασδήποτε χρήσης κλειδωμάτων.

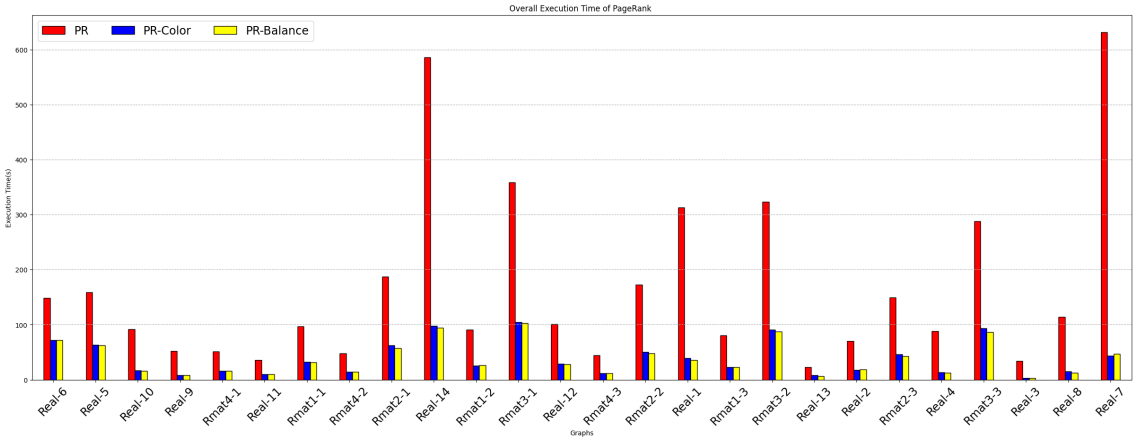
6.1.1 Αποτελέσματα

Για τα πειράματά μας, χρησιμοποιήσαμε τρεις εκδόσεις των αλγορίθμων. Την έκδοση PR, η οποία είναι ο PR με τη χρήση κλειδωμάτων, την έκδοση PR-Color, η οποία είναι ο PR με σχήμα χρωματισμού και την PR-Balance, η οποία είναι ο PR με σχήματα χρωματισμού και εξισορρόπησης.

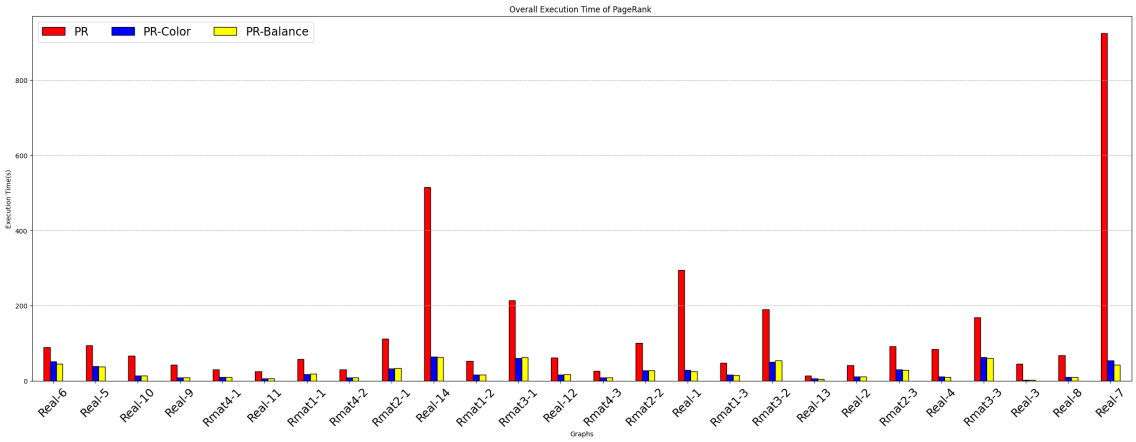
Στα σχήματα 6.1-6.6 βλέπουμε ότι ο παραλληλισμός των κορυφών με βάση το ευρετικό σύστημα χρωματισμού/εξισορρόπησης, βελτιώνει σημαντικά το συνολικό χρόνο εκτέλεσης της εφαρμογής. Όπως είναι αναμενόμενο, ο χρόνος εκτέλεσης μειώνεται καθώς αυξάνουμε τον αριθμό των νημάτων, αλλά η αναλογία μεταξύ PR και PR-Color/PR-Balance είναι πάντα η ίδια. Ένα ενδιαφέρον σημείο είναι ότι οι εκδόσεις PR-Color και PR-Balance έχουν σχεδόν τους ίδιους χρόνους εκτέλεσης και σε ορισμένες περιπτώσεις, η PR-Balance έχει μεγαλύτερους. Έτσι, το επόμενο βήμα είναι να συγκρίνουμε τον συνολικό χρόνο εκτέλεσης της PR με Coloring και με Coloring+Balancing. Στα σχήματα 6.7-6.12 μπορούμε να δούμε ότι η έκδοση με το βήμα εξισορρόπησης για όλες σχεδόν τις περιπτώσεις προσθέτει απλώς επιπλέον χρόνο εκτέλεσης. Αυτή η συμπεριφορά δεν αλλάζει καθώς αυξάνεται ο αριθμός των νημάτων που σημαίνει ότι για το ΠαγεΡανκ ένας χρωματισμός γράφου είναι αρκετός για να επιταχύνει την απόδοσή του.



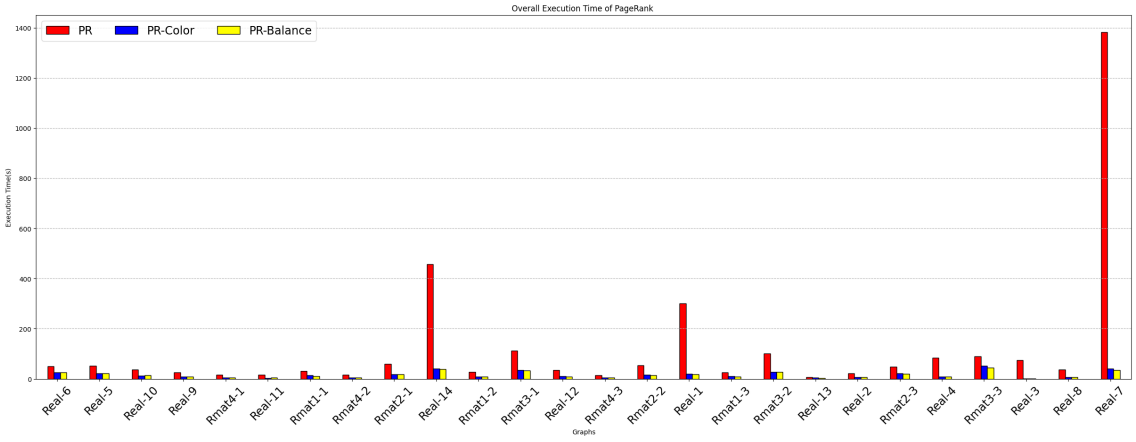
Εικόνα 6.1: *Overall Execution: Serial*



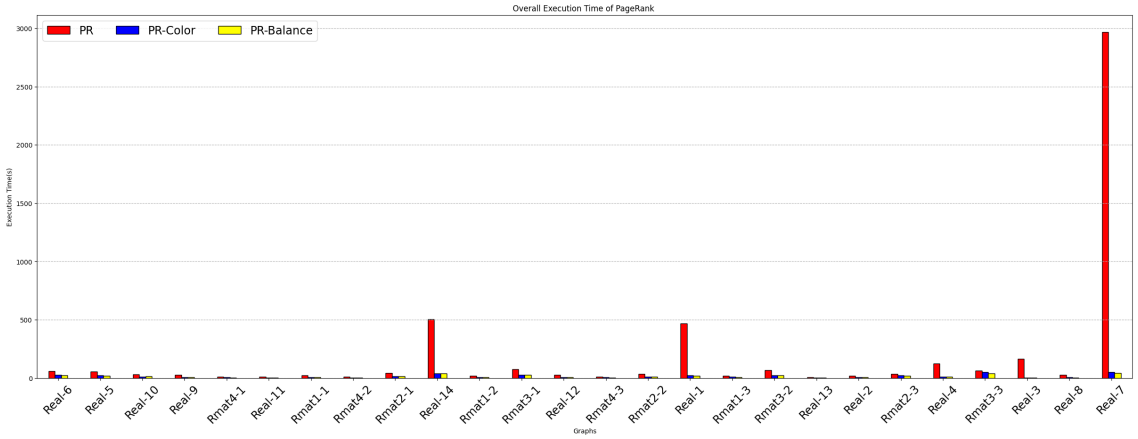
Εικόνα 6.2: *Overall Execution: 4 Threads*



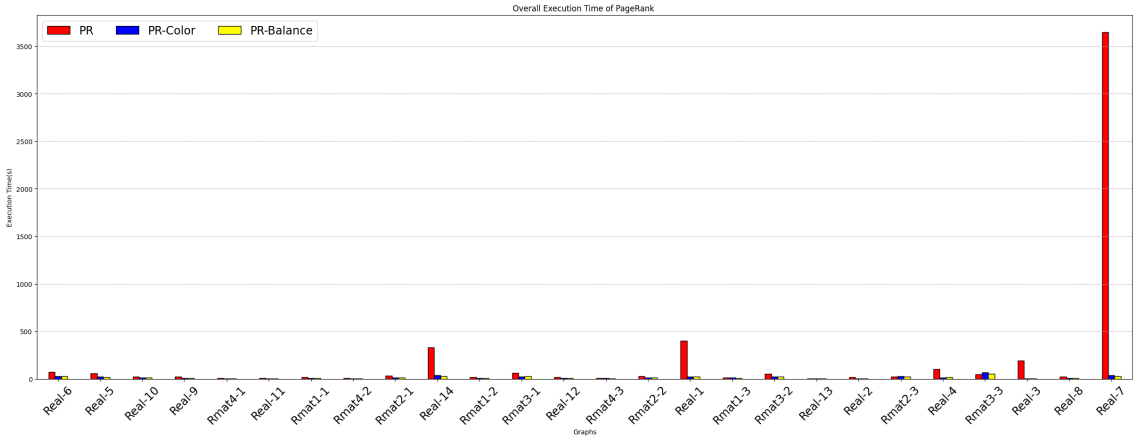
Εικόνα 6.3: *Overall Execution: 7 Threads*



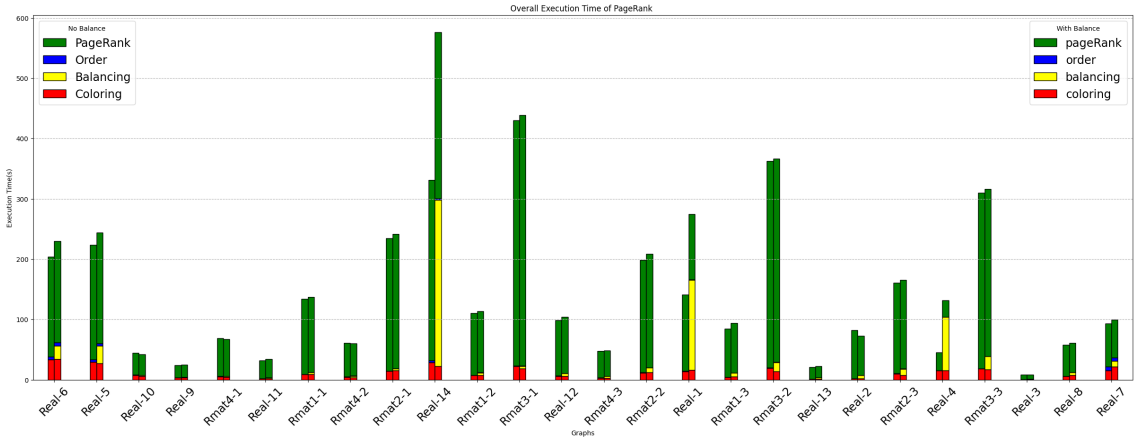
Εικόνα 6.4: *Overall Execution: 14 Threads*



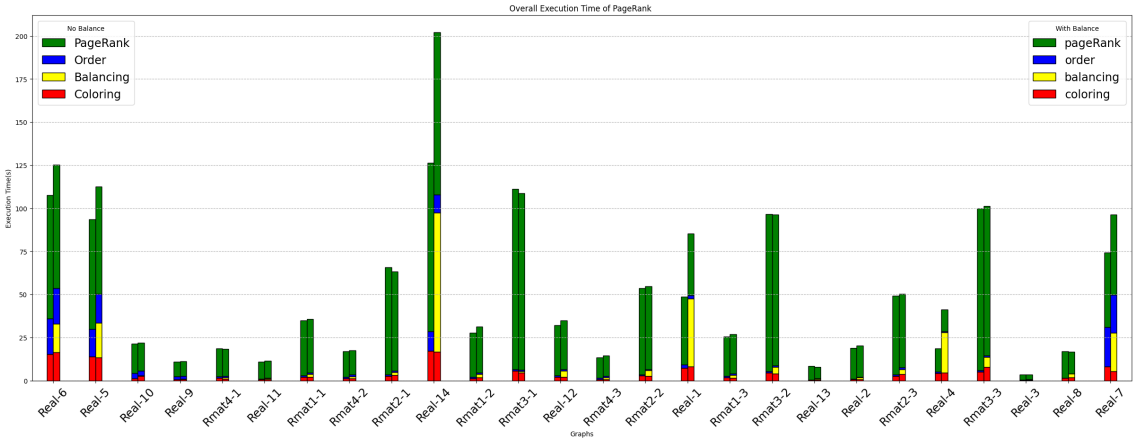
Εικόνα 6.5: *Overall Execution: 28 Threads*



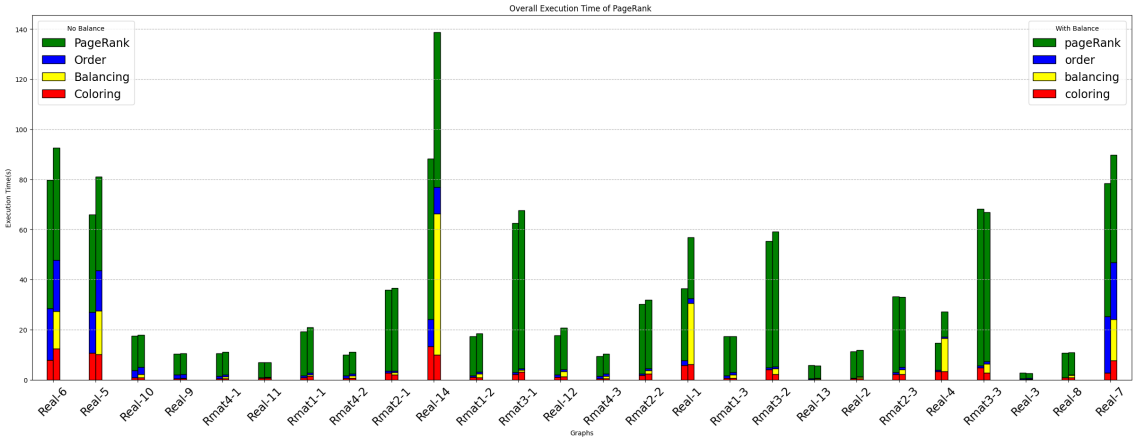
Εικόνα 6.6: *Overall Execution: 56 Threads*



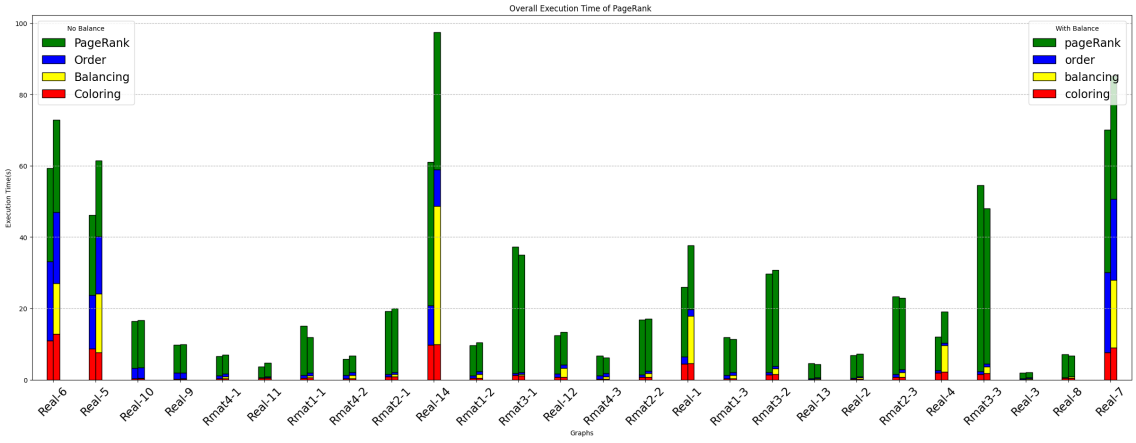
Εικόνα 6.7: *PR-Color vs PR-Balance: Serial*



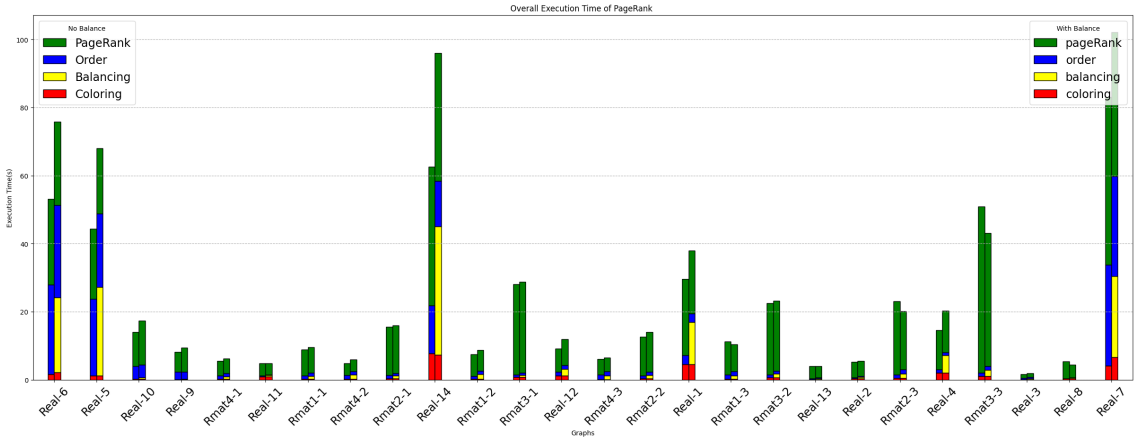
Εικόνα 6.8: *PR-Color vs PR-Balance: 4 Threads*



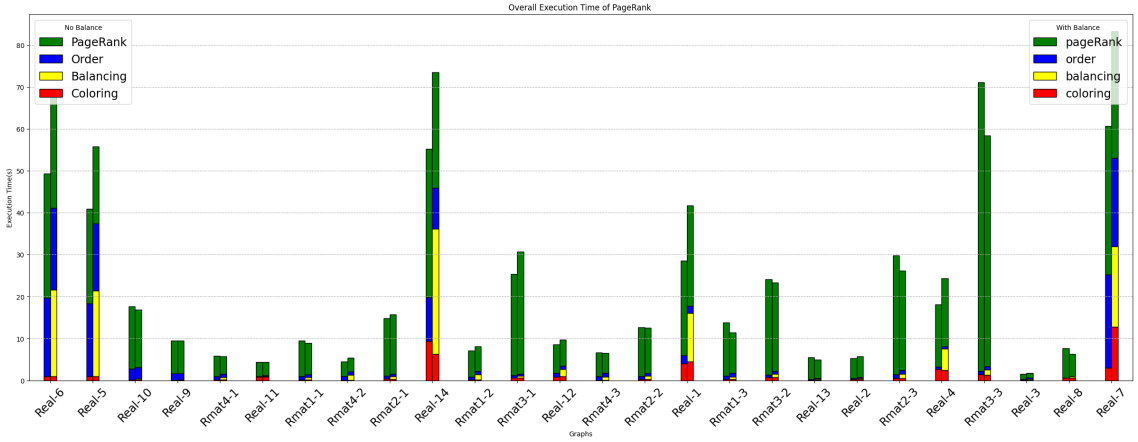
Εικόνα 6.9: *PR-Color vs PR-Balance: 7 Threads*



Εικόνα 6.10: *PR-Color vs PR-Balance: 14 Threads*



Εικόνα 6.11: *PR-Color vs PR-Balance: 28 Threads*



Εικόνα 6.12: *PR-Color vs PR-Balance: 56 Threads*

6.2 Εύρεση Κοινοτήτων

Η δεύτερη πραγματική εφαρμογή που θα χρησιμοποιήσουμε για να αξιολογήσουμε τα σχήματα χρωματισμού/εξισορρόπησης γράφων είναι η ανίχνευση κοινοτήτων. Η ανίχνευση κοινοτήτων είναι ένας αλγόριθμος που ταξινομεί τις κορυφές ενός γράφου σε κοινότητες, όπου μια κοινότητα αποτελείται από κορυφές που συνδέονται περισσότερο εσωτερικά παρά εξωτερικά. Ο ίδιος ο πυρήνας ανίχνευσης κοινοτήτων αποτελεί σημαντικό πυρήνα πολλών άλλων εφαρμογών στον τομέα της ανάλυσης δικτύων.

Υπάρχουν πολλοί αλγόριθμοι για την επίλυση του προβλήματος της ανίχνευσης κοινοτήτων, αλλά εδώ θα εξερευνήσουμε τη μέθοδο Modularity. Η Modularity είναι μια μετρική που ποσοτικοποιεί την ποιότητα μιας κοινότητας στο εσωτερικό ενός γράφου και η περίπτωση εδώ είναι η μεγιστοποίηση της τιμής αυτής. Η ίδια η μεγιστοποίηση της, είναι ένα NP-πλήρες πρόβλημα επομένως η χρήση μιας ευρετικής μεθόδου είναι μονόδρομος. Μια από αυτές τις ευρετικές μεθόδους είναι η μέθοδος Louvain, η οποία είναι μια επαναληπτική, άπληστη μέθοδος που δημιουργεί μια ιεραρχία κοινοτήτων και περιγράφεται παρακάτω

Μέθοδος Louvain: Αρχικά ξεκινάμε με κάθε κορυφή να αποτελεί και μία κοινότητα (γραμμή 1: C_{init}) και για κάθε κορυφή (γραμμή 9) εξερευνούμε μόνο τη γειτονιά της. Για κάθε γείτονα υπολογίζουμε αν υπάρχει κέρδος στην μετρική του modularity εαν και εφόσον η κορυφή αυτή μετακινηθεί στην κοινότητα της αρχικής κορυφής (γραμμές 10-15). Εάν υπάρχει μετακίνηση που μεγιστοποιεί το modularity για μια συγκεκριμένη κορυφή, η κορυφή θα μετακινηθεί(γραμμές 14,15), διαφορετικά θα παραμείνει στην αρχική της κοινότητα. Στο τέλος μιας επανάληψης, όλες οι κορυφές έχουν επεξεργαστεί και οι νέες κοινότητες βρίσκονται στο σύνολο C_curr. Συνεχίζοντας, εάν φτάσουμε σε ένα σημείο κατά το οποίο τυχόν μετακινήσεις δεν επιφέρουν κάποιο κέρδος, πρακτικά να είναι μεγαλύτερο από ένα κατώφλι που καθορίζεται από τον χρήστη, ο αλγόριθμος τερματίζεται(γραμμή 19), διαφορετικά συνεχίζομαι με μια ακόμη επανάληψη όπου η κοινότητα εισόδου για τη νέα επανάληψη είναι το σύνολο κοινοτήτων της τρέχουσας επανάληψης(γραμμή 21).

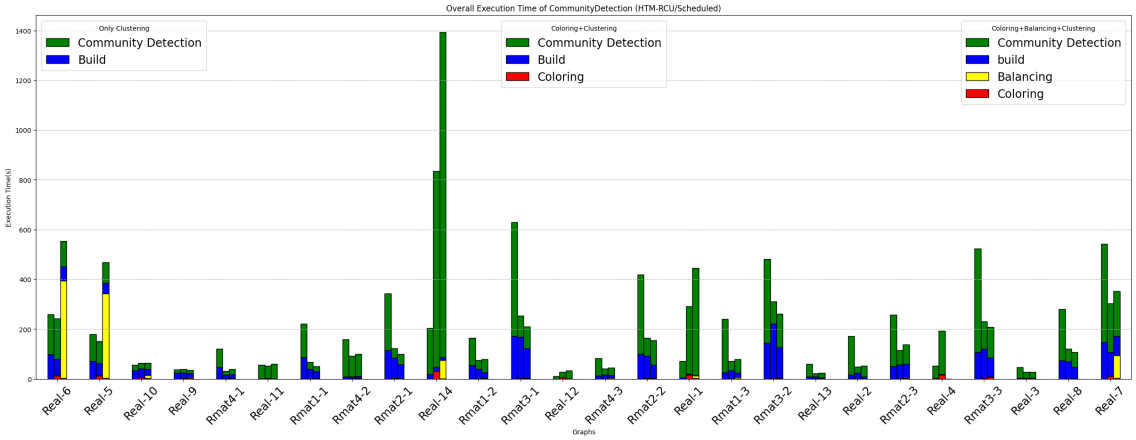
Για να διερευνήσουμε γιατί ο χρωματισμός γράφων είναι επωφελής για την ανίχνευση κοινοτήτων πρέπει πρώτα να δούμε ένα προβληματικό σενάριο που μπορεί να προκύψει όταν δεν το χρησιμοποιούμε για τον παραλληλισμό των δεδομένων. Όταν εκτελούμε τη σειριακή λειτουργία, οι κορυφές επισκέπτονται διαδοχικά, πράγμα που σημαίνει ότι ο αλγόριθμος λειτουργεί με τις τελευταίες ενημερώσεις. Ωστόσο, αυτό δεν ισχύει όταν εκτελούμε σε πολυνηματική λειτουργία, όπου δύο γειτονικές κορυφές μπορεί να υποστούν ταυτόχρονη επεξεργασία. Αν αυτές οι κορυφές αποφασίσουν να μετακινηθούν σε άλλες κοινότητες τότε έχουμε μια ανταλλαγή και αυτή η μετακίνηση δεν έχει κανένα απολύτως όφελος καθώς καθυστερεί τη σύγκλιση του modularity. Φυσικά αυτό το σενάριο μπορεί να συμβεί όχι μόνο σε ένα ζεύγος ο κορυφών αλλά και σε ένα υποσύνολο, προσθέτοντας επιβάρυνση στο χρόνο εκτέλεσης καθόλου ευκαταφρόνητη. Ωστόσο, παραλληλοποιώντας τα δεδομένα με τη βοήθεια του χρωματισμού του γράφου μπορούμε να δούμε ότι ένα τέτοιο σενάριο δεν μπορεί να συμβεί επειδή οι κορυφές με το ίδιο χρώμα επεξεργάζονται παράλληλα και αυτό ισοδυναμεί με την εγγύηση ότι δεν θα γίνει ταυτόχρονη επεξεργασία δύο γειτονικών κορυφών. Με λίγα λόγια, ο χρωματισμός γράφων σημαίνει λιγότερες επαναλήψεις για την επίτευξη σύγκλισης άρα μικρότερο χρόνο εκτέλεσης.

Αλγοριθμος 6.2: *Community Detection*

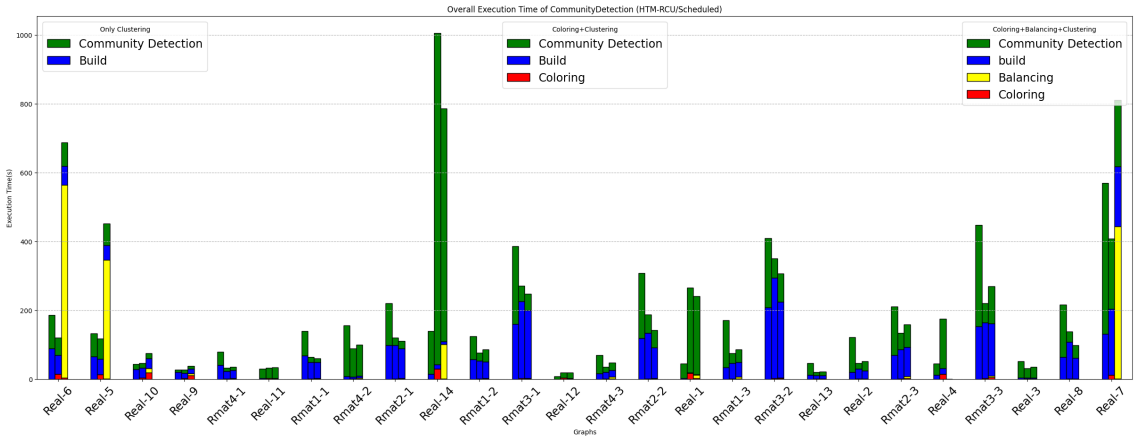
1: *Input*(V, E, ω, C_{init})
2: *color* $\leftarrow$ *Initial_Coloring*
3: $Q_{curr} \leftarrow 0$ ▷ current modularity
4: $Q_{prev} \leftarrow -\infty$ ▷ previous modularity
5: $C_{curr} \leftarrow C_{init}$ ▷ set of communities
6: **while** τρυε **do**
7: **for** $V_k \in$ *color* **do**
8: $C_{prev} \leftarrow C_{curr}$
9: **for** $i \in V_k$ ιν παραλλελ **do**
10: $N_i \leftarrow C_{prev}[i]$
11: **for** $j \in \Gamma(i)$ **do**
12: $N_i \leftarrow N_i \bigcup \{C_{prev}[j]\}$
13: **end for**
14: $target \leftarrow argmax_{t in N_i} \Delta Q_{i \to t}$
15: **if** $\Delta Q_{i \to t}$ **then**
16: $C_{curr}[i] \leftarrow target$
17: **end if**
18: **end for**
19: **end for**
20: $C_{set} \leftarrow$ *set of communities corresponding to* C_{curr}
21: $Q_{curr} \leftarrow$ *compute modularity as defined by* C_{set}
22: **if** $|\frac{Q_{curr} - Q_{prev}}{Q_{prev}}| < \partial$ **then** ▷ ∂ is user specified
23: βρεακ
24: **else**
25: $Q_{prev} \leftarrow Q_{curr}$
26: **end if**
27: **end while**

6.2.1 Αποτελέσματα

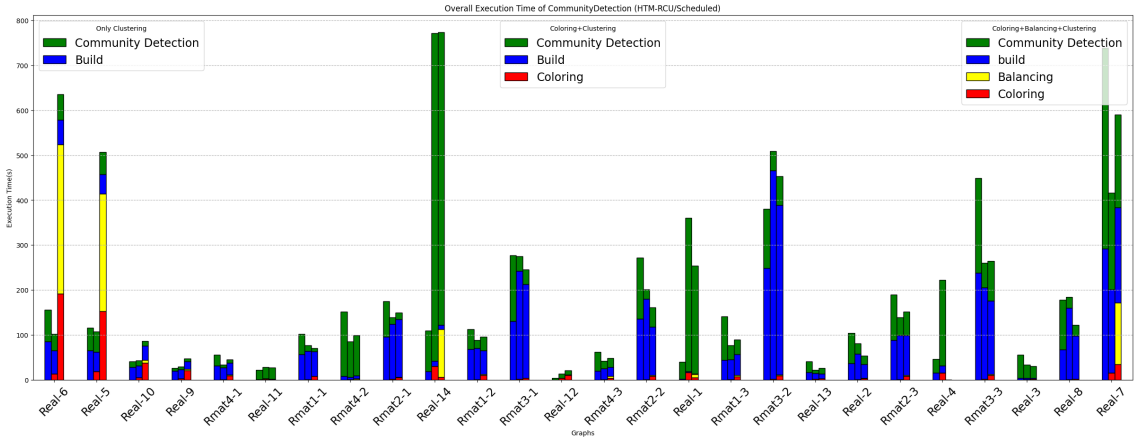
Αρχικά θα δούμε αν ο χρωματισμός και η εξισορρόπηση είναι πράγματι ωφέλιμα για την ανίχνευση της κοινότητας και αν τα αποτελέσματα ευθυγραμμίζονται με τις εικασίες μας. Στα σχήματα 6.13-6.15 συγκρίνουμε το συνολικό χρόνο εκτέλεσης της εφαρμογής σε τρεις εκδόσεις: η πρώτη έκδοση είναι η ανίχνευση κοινότητας χωρίς τη χρήση χρωματισμού ή/και εξισορρόπησης, η δεύτερη έκδοση με τη χρήση χρωματισμού και η τρίτη και τελευταία έκδοση με τη χρήση τόσο χρωματισμού όσο και εξισορρόπησης. Στο κεφάλαιο 3 διαπιστώσαμε ότι τα δύο κυρίαρχα σχήματα στη σουίτα γράφων μας είναι το SS και το HTM-RCU και στο κεφάλαιο 5 διαπιστώσαμε ότι το κυρίαρχο σχήμα όσον αφορά το χρόνο εκτέλεσης είναι το Scheduled. Επομένως, εδώ χρησιμοποιούμε για το βήμα της εξισορρόπησης το σχήμα Scheduled και για το βήμα του χρωματισμού το HTM-P"Y για να έχουμε την ίδια βάση και να μπορούμε να συγκρίνουμε τα αποτελέσματα. Η πρώτη διαπίστωση που μπορούμε να αντλήσουμε από αυτά τα διαγράμματα είναι ότι πράγματι ο χρωματισμός και η εξισορρόπηση επιταχύνουν την απόδοση σε όλους τους τρόπους στις περισσότερες περιπτώσεις. Φυσικά, καθώς αυξάνεται ο αριθμός των νημάτων, οι καταστάσεις είναι πιο περίπλοκες όσον αφορά τον συγχρονισμό, οπότε τα αποτελέσματα είναι πιο ανάμεικτα. Δεύτερον, μπορούμε να παρατηρήσουμε ότι το βήμα της ομαδοποίησης -μέθοδος Louvain- (πράσινο χρώμα) στις περισσότερες περιπτώσεις μειώνεται ακόμη και με τη χρήση της εξισορρόπησης, ένα αποτέλεσμα που δείχνει ότι η ομαδοποίηση από μόνη της βοηθιέται από την εξισορρόπηση ακόμη και αν ο συνολικός χρόνος εκτέλεσης δεν είναι τόσο βέλτιστος όσο στην έκδοση του χρωματισμού.



Εικόνα 6.13: *Overall Execution: 14 Threads*

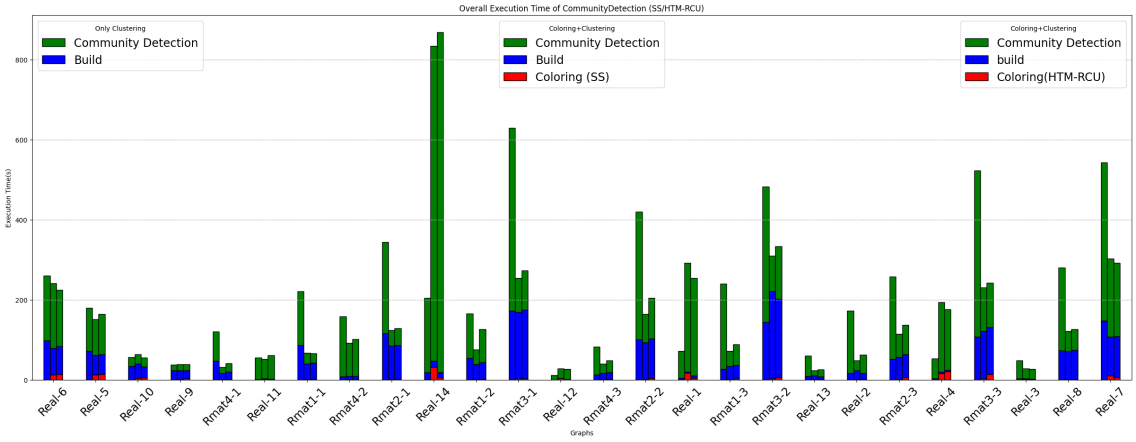


Εικόνα 6.14: *Overall Execution: 28 Threads*

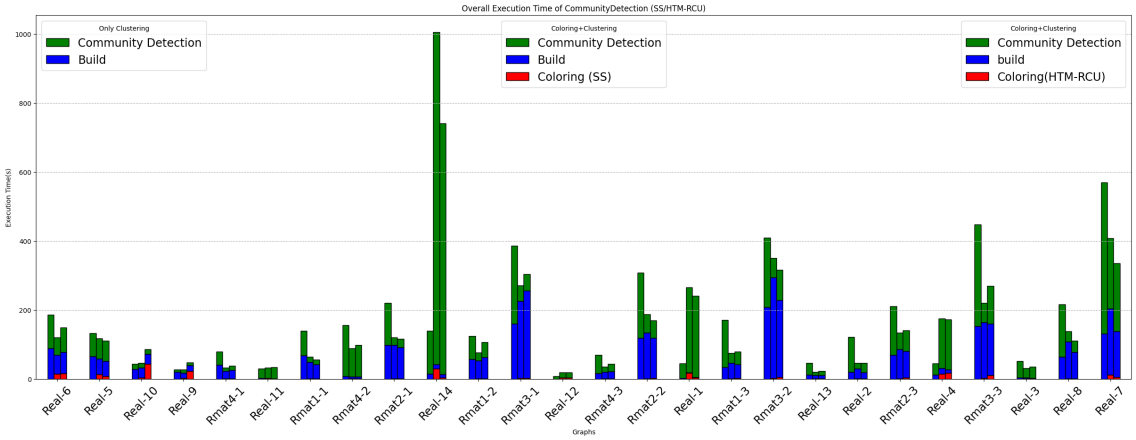


Εικόνα 6.15: *Overall Execution: 56 Threads*

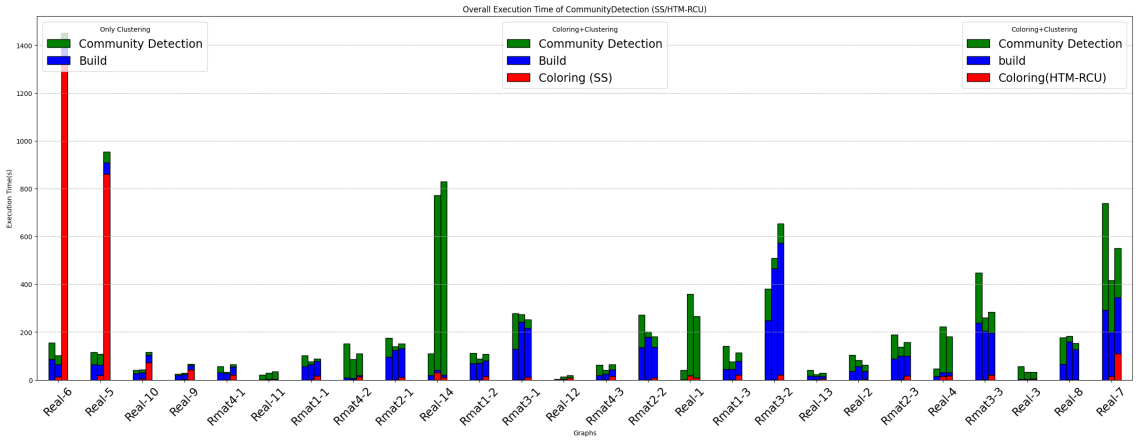
Για να συνεχίσουμε την ανάλυσή μας, στα επόμενα 3 διαγράμματα 6.16-6.18 συγκρίνουμε την ανίχνευση της κοινότητας με το βήμα του χρωματισμού χρησιμοποιώντας τα σχήματα SS και HTM-RCU. Τα αποτελέσματα εδώ διαφέρουν από τις παρατηρήσεις που αντλήσαμε στο κεφάλαιο 3, καθώς το σχήμα SS υπερτερεί του σχήματος HTM-RCU και στους 3 τρόπους. Αυτή η συμπεριφορά υποδεικνύει ότι η απόδοση όσον αφορά τον χρόνο εκτέλεσης είναι λιγότερο σημαντική από την απόδοση όσον αφορά τον αριθμό των χρωμάτων. Στο κεφάλαιο 3 διαπιστώσαμε ότι το SS υπερτερεί ως προς τον αριθμό των χρωμάτων, επομένως μπορεί να επιτρέψει περισσότερο παραλληλισμό.



Εικόνα 6.16: *Overall Execution SS vs HTM-RCU: 14 Threads*

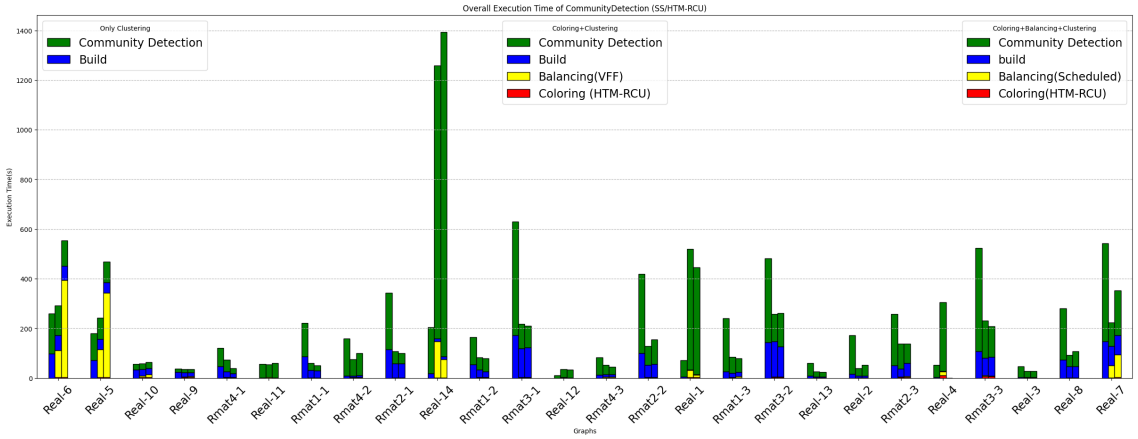


Εικόνα 6.17: *Overall Execution SS vs HTM-RCU: 28 Threads*

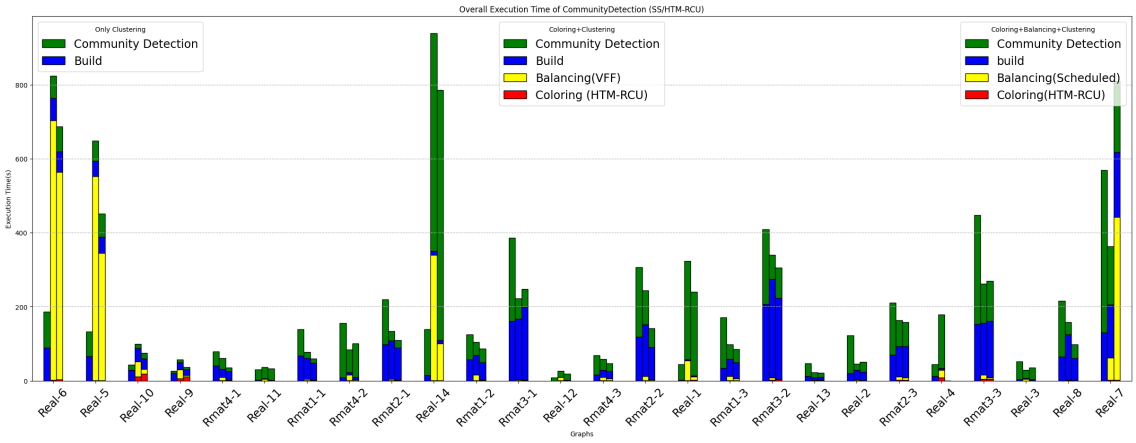


Εικόνα 6.18: *Overall Execution SS vs HTM-RCU: 56 Threads*

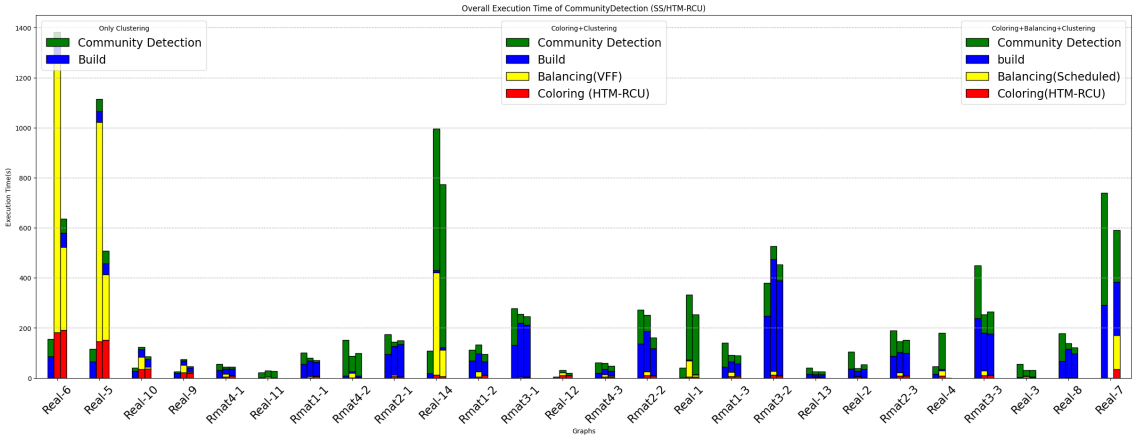
Τέλος, στα σχήματα 6.19-6.21 συγκρίνουμε για το βήμα εξισορρόπησης τα σχήματα Scheduled και VFF για να δούμε αν είναι πιο σημαντική για την ανίχνευση της κοινότητας η απόδοση ως προς τον χρόνο εκτέλεσης ή ως προς την ποιότητα της εξισορρόπησης. Σε αυτή την κατάσταση όπως βλέπουμε, σε 14 νήματα τα σχήματα μοιράζονται σχεδόν εξίσου όπου το VFF υπερτερεί σε 11 εισόδους και το Scheduled σε 9. Στα 6 υπόλοιπα η απόδοση είναι η ίδια. Ωστόσο, καθώς αυξάνουμε τον αριθμό των νημάτων αυτή η τάση δεν ισχύει πλέον καθώς το σχήμα Scheduled υπερισχύει έναντι του VFF σε 28 νήματα για 18 εισόδους και σε 56 νήματα για 15 εισόδους. Αυτή η συμπεριφορά δείχνει ότι για την ανίχνευση της κοινότητας είναι πιο σημαντικό για την εξισορρόπηση να έχει καλή απόδοση όσον αφορά τον χρόνο εκτέλεσης παρά να παράγει καλή ποιότητα ισορροπίας.



Εικόνα 6.19: *Overall Execution VFF vs Scheduled: 14 Threads*



Εικόνα 6.20: *Overall Execution VFF vs Scheduled: 28 Threads*



Εικόνα 6.21: *Overall Execution VFF vs Scheduled: 56 Threads*

Βιβλιογραφία

[1] Tim Kaler, William Hasenplaugh, Tao B. Schardl και Charles E. Leiserson. *Executing Dynamic Data-Graph Computations Deterministically Using Chromatic Scheduling*. *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, σελίδα 154–165, New York, NY, USA, 2014. Association for Computing Machinery.

[2] Gregory J. Chaitin, Marc A. Auslander, Ashok K. Chandra, John Cocke, Martin E. Hopkins και Peter W. Markstein. *Register allocation via graph coloring*. 1981.

[3] Thomas F. Coleman και Jorge J. Moré. *Estimation of sparse jacobian matrices and graph coloring problems*. *SIAM Journal on Numerical Analysis*, 20:187–209, 1983.

[4] Y. Saad. *Sparskit: a basic tool kit for sparse matrix computations*. 1994.

[5] D. J. A. Welsh και M. B. Powell. *An upper bound for the chromatic number of a graph and its application to timetabling problems*. *The Computer Journal*, 10(1):85–86, 1967.

[6] M. R. Garey, D. S. Johnson και H. C. So. *An Application of Graph Coloring to Printed Circuit Testing*. *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS '75, σελίδα 178–183, USA, 1975. IEEE Computer Society.

[7] Lawrence Page, Sergey Brin, Rajeev Motwani και Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Τεχνικαλ Ρεπορτ 1999-66, Stanford InfoLab, 1999. Πρεους νυμβερ = ΣΙΔΛ-ΩΠ-1999-0120.

[8] Hao Lu, Mahantesh Halappanavar και Ananth Kalyanaraman. *Parallel heuristics for scalable community detection*. *Parallel Computing*, 47:19–37, 2015. Γραπη αναλψσις φορ σχιεντιφις δισχοερψ.

[9] M. R. Garey, D. S. Johnson και L. Stockmeyer. *Some Simplified NP-Complete Problems*. *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, σελίδα 47–63, New York, NY, USA, 1974. Association for Computing Machinery.

[10] Assefaw Hadish Gebremedhin και Fredrik Manne. *Scalable parallel graph coloring algorithms*. *Concurrency: Practice and Experience*, 12(12):1131–1146, 2000.

[11] Erik G. Boman, Doruk Bozdağ, Umit Catalyurek, Assefaw H. Gebremedhin και Fredrik Manne. *A Scalable Parallel Graph Coloring Algorithm for Distributed Memory Computers*. *Euro-Par 2005 Parallel Processing*José C. Cunha και Pedro D. Medeiros, επιμελητές, σελίδες 241–251, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] Georgios Rokos, Gerard Gorman και Paul H J Kelly. *A Fast and Scalable Graph Coloring Algorithm for Multi-core and Many-core Architectures*, 2015.

[13] Christina Giannoula, Georgios Goumas και Nectarios Koziris. *Combining HTM with RCU to Speed Up Graph Coloring on Multicore Platforms*. *High Performance Computing*Rio Yokota, Michèle Weiland, David Keyes και Carsten Trinitis, επιμελητές, σελίδες 350–369, Cham, 2018. Springer International Publishing.

[14] Hao Lu, Mahantesh Halappanavar, Daniel Chavarría-Miranda, Assefaw Gebremedhin και Ananth Kalyanaraman. *Balanced Coloring for Parallel Computing Applications*. *2015 IEEE International Parallel and Distributed Processing Symposium*, σελίδες 7–16, 2015.

[15] Umit Catalyurek, John Feo, Assefaw Gebremedhin, Mahantesh Halappanavar και Alex Pothen. *Graph Coloring Algorithms for Muti-core and Massively Multithreaded Architectures*, 2012.

[16] Paul E. McKenney και Jack Slingwine. *READ-COPY UPDATE: USING EXECUTION HISTORY TO SOLVE CONCURRENCY PROBLEMS*. 2002.

[17] Maurice Herlihy και Nir Shavit. *The Art of Multiprocessor Programming, Revised Reprint*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1στη έκδοση, 2012.

[18] *SuiteSparse Matrix Collection*. https://sparse.tamu.edu/. Ημερομηνία πρόσβασης: 13-6-2021.