

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μελέτη και ανάπτυξη
αλγορίθμων χρωματισμού σε
κατανεμημένα δίκτυα
αισθητήρων

Συγγραφέας:
Αθανάσιος ΚΑΤΑΡΑΣ

Επιβλέπων:
Καθηγητής Πάυλος
ΣΠΥΡΑΚΗΣ

Τριμελής Επιτροπή:
κ.ΣΠΥΡΑΚΗΣ Πάυλος
κ.ΝΙΚΟΛΕΤΣΕΑΣ Σωτήρης
κ.ΚΑΚΛΑΜΑΝΗΣ Χρήστος

26 Απριλίου 2010

Περιεχόμενα

1	Εισαγωγή	7
2	Το πρόβλημα του χρωματισμού γράφων	9
2.1	Εισαγωγή	9
2.2	Χρωματισμός γράφων	10
2.2.1	Περιγραφή των προβλημάτων χρωματισμού	10
2.2.2	Πρακτική εφαρμογή του χρωματισμού γράφων	12
2.3	Θεωρία πολυπλοκότητας και κλάσεις NP,P	13
2.3.1	Πολυπλοκότητα	13
2.3.2	Κλάσεις Προβλημάτων P και NP	14
2.4	Προσεγγιστικοί αλγόριθμοι και Θεωρία παιγνίων	16
3	Κατανεμημένα Συστήματα-Δίκτυα αισθητήρων	21
3.1	Εισαγωγή	21
3.2	Κατανεμημένα Συστήματα	22
3.2.1	Ιστορία των κατανεμημένων	22
3.2.2	Πρακτικές Εφαρμογές	22
3.2.3	Coloring σε κατανεμημένα δίκτυα	23
3.3	Ασύρματα Δίκτυα Αισθητήρων	25
3.3.1	Πρακτικές εφαρμογές	26
3.3.2	Προκλήσεις στον τομέα των Wireless Sensor Networks (WSN)	28
3.3.3	Διαφορές με άλλα ad-hoc δίκτυα	31
3.3.4	Μηχανισμοί και απαιτήσεις στους αλγορίθμους που αναπτύσσουμε	32
4	Βασικά εργαλεία υλοποίησης	35
4.1	Εισαγωγή	35
4.2	Περιβάλλοντα ανάπτυξης	35
4.2.1	Ο Εξομοιωτής shawn	35
4.2.2	Η βιβλιοθήκη wiselib	41
4.3	Χαρακτηριστικά υλοποίησης	45
4.3.1	Tora Routing	45
4.3.2	C/C++,Templates,delegates	46
4.3.3	Δομές δεδομένων	47

5	Αλγόριθμοι και Εξομοίωση	49
5.1	Εισαγωγή	49
5.2	Προτεινόμενοι αλγόριθμοι	50
5.2.1	Υπάρχοντες αλγόριθμοι	50
5.2.2	Ο Αλγόριθμος με δικαστή	50
5.2.3	Σειριακός αλγόριθμος πολλαπλών δικαστών	52
5.2.4	Παράλληλος αλγόριθμος πολλαπλών δικαστών	54
5.2.5	Αλγόριθμοι περιορισμένης γνώσης πλήρως κατανεμημένοι	56
5.3	Εξομοίωση	60
5.3.1	Δημιουργία των γράφων	60
5.3.2	Ομαδοποίηση κόμβων και ανάθεση σε δικαστή	66
6	Αξιολόγηση και Συμπεράσματα	71
6.1	Εισαγωγή	71
6.2	Μετρικές	72
6.2.1	Χρωματικός Αριθμός	72
6.2.2	Αριθμός Μηνυμάτων	72
6.2.3	Χρόνος εκτέλεσης	73
6.3	Συγκρίσεις αλγορίθμων	74
6.3.1	Αλγόριθμος με έναν δικαστή	74
6.3.2	Σειριακός αλγόριθμος με πολλούς δικαστές	79
6.3.3	Παράλληλος αλγόριθμος πολλαπλών δικαστών	83
6.3.4	Αλγόριθμος δύο βημάτων	87
6.4	Συμπεράσματα	92
6.5	Μελλοντική Εργασία και Στόχοι	92

Ευχαριστίες

Για την υλοποίηση και την συγγραφή της παρούσης διπλωματικής, θα ήθελα στο σημείο αυτό να ευχαριστήσω τους ανθρώπους που βοήθησαν για την ολοκλήρωσή της, η οποία θα ήταν αδύνατη χωρίς την συμβολή τους:

Τον κ. Σπυράκη Παύλο, επιβλέποντα της εργασίας για την εύρεση και την ανάθεση του θέματος, καθώς και για όλη την βοήθεια που παρείχε και την εμπιστοσύνη που μου έδειξε καθ'όλη την διάρκεια της εργασίας.

Τα δυο μέλη της επιτροπής κ.κ. καθηγητή Κακλαμάνη Χρήστο και επίκουρο καθηγητή Νικολετσέα Σωτήρη, για την παρουσία τους και τα σχόλια τους κατά την παρουσίαση της εργασίας.

Τον Δρ. Χατζηγιαννάκη Ιωάννη ΠΔ407 του τμήματος Μηχανικών Η/Υ και πληροφορικής και όλη την ομάδα του στην ερευνητική μονάδα 1 του ΕΑΙΤΥ για την βοήθειά του σε όλα τα τεχνικά ζητήματα που προέκυψαν κατά την διάρκεια της εργασίας.

Την Δρ. Παναγοπούλου Παναγιώτα για την βοήθεια της στον σχεδιασμό των αλγορίθμων και τις επεξηγήσεις στην θεωρία που βασίστηκε η εργασία.

Τέλος θα ήθελα να ευχαριστήσω όλους τους φίλους και την οικογένεια μου για την συμπαράστασή τους όλο αυτό τον καιρό.

Κεφάλαιο 1

Εισαγωγή

Το θέμα της εργασίας: Στην παρούσα διπλωματική θα μελετήσουμε το θέμα του χρωματισμού γράφων με τα λιγότερα δυνατά χρώματα έτσι ώστε δύο οποιεσδήποτε κορυφές που συνδέονται μεταξύ τους να μην έχουν το ίδιο χρώμα. Το πρόβλημα αυτό αναλύεται στα πλαίσια ενός είδους κατανεμημένων δικτύων, γνωστά ως ασύρματα δίκτυα αισθητήρων.

Στόχοι: Στόχος είναι η ανάπτυξη και η αιολόγηση αλγορίθμων που βασίζονται σε υπάρχουσες προτάσεις (π.χ. A Game Theoretic Approach for Efficient Graph Coloring από Παύλο Σπυράκη και Παναγιώτα Παναγοπούλου), με την διαφορά ότι θα μελετήσουμε παραλλαγές τους ώστε να λειτουργούν ικανοποιητικά σε κατανεμημένα περιβάλλοντα. Η επιλογή αυτή έγινε αφενός λόγω της έλλειψης ερευνητικής εργασίας στο συγκεκριμένο αντικείμενο και αφετέρου γιατί έχουμε στο μέλλον την δυνατότητα να μελετήσουμε την απόδοση και την πρακτικότητα του αλγορίθμου σε πραγματικό περιβάλλον, που αποτελείται από ένα σύνολο αισθητήρων εγκατεστημένων στο Ερευνητικό Ακαδημαϊκό Ινστιτούτο Τεχνολογίας Υπολογιστών στην Πάτρα.

Προσφορά: Το πρόβλημα του χρωματισμού είναι ένα πρόβλημα μεγάλης σημασίας τόσο στον τομέα των υπολογιστών (προβλήματα χρονισμού , πρόβλημα κατανομής καταχωρητών (registers), προβλήματα κρυπτογραφίας κλπ), όσο και σε άλλους τομείς (απόδοση συχνοτήτων σε κεραίες στις τηλεπικοινωνίες).

Ευελπιστούμε με αυτήν την εργασία να δώσουμε στην επιστημονική κοινότητα για πρώτη φορά την δυνατότητα να δει αποτελέσματα σε πρακτικό επίπεδο πάνω στον χρωματισμό δικτύων αισθητήρων πράγμα που από όσο ξέρουμε δεν έχει συμβεί μέχρι τώρα. Εκτός όμως από αυτό η υλοποίηση αλγορίθμων σε περιβάλλον που μπορεί να μεταφερθεί εύκολα σε πραγματικές συσκευές θα προσφέρει για πρώτη φορά κώδικα που οι μελλοντικοί ερευνητές στα θέματα ασύρματων δικτύων αισθητήρων να μπορούν αν χρησιμοποιήσουν όταν αυτό χρειαστεί.

Δομή: Αρχικά θα δούμε αναλυτικά την θεωρία που απαιτείται για την κατανόηση του προβλήματος του χρωματισμού και ειδικότερα τις κλάσεις πολυπλοκότητας που ανήκει, τα προβλήματα που δημιουργούν και πως επιλύονται [κεφάλαιο 2]. Αμέσως μετά περιγράφονται τα κατανεμημένα δίκτυα και τα ασύρματα δίκτυα αισθητήρων (χρησιμότητα/λειτουργίες) [κεφάλαιο 3].

Μετά το πέρας των βασικών εννοιών θα δοθούν οι αλγόριθμοι [Κεφάλαιο 5] και οι ρυθμίσεις των δοκιμών [Κεφάλαιο 6] που πραγματοποιήθηκαν σε ένα περιβάλλον εξομοίωσης (simulation) που ονομάζεται shawn , έχει αναπτυχθεί σε C++ και είναι ιδανικό για εξομοιώσεις ασύρματων δικτύων αισθητήρων. Τα δίκτυα αυτά αποτελούν ένα καταναμεμημένο δίκτυο και θα μπορούσαν να ωφεληθούν ιδιαίτερα από την ύπαρξη ενός αλγορίθμου χρωματισμού. Μόλις ολοκληρωθούν και δοκιμαστούν οι αλγόριθμοι στόχος είναι να μεταφερθούν στο πραγματικό περιβάλλον για περαιτέρω μελέτη. Τόσο το shawn όσο και η βιβλιοθήκη wiselib που χρησιμοποιήθηκε για την υλοποίηση μας ούτως ώστε να είναι εύκολη η μεταφορά των αλγορίθμων σε άλλα περιβάλλοντα αναλύονται εκτενώς [κεφάλαιο 4].

Τέλος στο κομμάτι της μελέτης των αλγορίθμων [κεφάλαιο 6], αρχικά θα κάνουμε πειραματικές μετρήσεις για να δούμε την απόδοσή τους σε σχέση με τους μη καταναμεμημένους προσεγγιστικούς αλγορίθμους, ως προς τον χρωματικό αριθμό αλλά και ως προς την ταχύτητα με την οποία τελειώνει ο αλγόριθμος, αριθμό μηνυμάτων που ανταλλάχτηκαν κλπ.

Κεφάλαιο 2

Το πρόβλημα του χρωματισμού γράφων

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα μελετήσουμε το πρόβλημα του χρωματισμού γράφων. Αρχικά θα μιλήσουμε για την ιστορία του, πως ξεκίνησε και που βρισκόμαστε τώρα, θα συνεχίσουμε με τους ορισμούς του γενικού προβλήματος και θα δούμε επιγραμματικά κάποια άλλα, πολύ σημαντικά προβλήματα που συσχετίζονται με το πρόβλημα χρωματισμού.

Επιπλέον παρουσιάζουμε την υπολογιστική πολυπλοκότητα των προβλημάτων αυτών, διάφορες προσεγγιστικές λύσεις για το κυριότερο από τα προβλήματα αυτά για την εργασία μας (το Vertex Coloring), και ασχολούμαστε ιδιαίτερα με τον προσεγγιστικό αλγόριθμο που βασίζεται στην θεωρία παιγνίων (στον οποίο βασίζουμε τις παραλλαγές του αλγορίθμου για τα δίκτυα που μας απασχολούν).

Ένα κομμάτι επίσης του κεφαλαίου θα ασχοληθεί με την θεωρία που απαιτείται για να κατανοήσει ο αναγνώστης τους διάφορους ορισμούς που θα χρησιμοποιηθούν σε όλη την εργασία. Τέλος θα δούμε διάφορες πρακτικές ανάγκες που μας οδήγησαν στο να ασχοληθούμε με το συγκεκριμένο πρόβλημα, και πως η επίλυση του προβλήματος έστω και προσεγγιστικά μας βοηθάει με αυτές.

2.2 Χρωματισμός γραφών

Ιστορική αναδρομή Το 1852 ο Francis Guthrie ήταν ο πρώτος που παρατήρησε το εξής παράξενο φαινόμενο. Είδε στην προσπάθειά του να ζωγραφίσει έναν πολιτικό χάρτη της Αγγλίας ότι μάλλον δεν θα χρειαζόταν να χρησιμοποιήσει παραπάνω από 4 χρώματα για να καταφέρει να μην έχει καμία περιοχή ίδιο χρώμα με την διπλανή της. Ο ίδιος άνθρωπος προχώρησε ακόμα πιο μακριά με το να προτείνει ότι δεν χρειάζεται παραπάνω από τέσσερα χρώματα για να χρωματίσει έναν οποιονδήποτε χάρτη.

Η ιδέα αυτή, που φαίνεται εύκολο να αποδειχτεί ότι είτε ισχύει είτε όχι, προχώρησε από τον Francis Guthrie στον αδερφό του και κατ' επέκταση στον καθηγητή του Augustus De Morgan στον οποίο μάλιστα αποδίδεται και το θεώρημα στην πρώτη του γραπτή μορφή. Αυτό που ούτε ο Francis ούτε ο Augustus θα μπορούσαν όμως να φανταστούν, είναι ότι άνοιξαν τον δρόμο για την έρευνα ενός από τα πιο ενδιαφέροντα σύνολα προβλημάτων μεγάλης υπολογιστικής πολυπλοκότητας στην ιστορία των υπολογιστών.

Το θεώρημα που δημιουργήθηκε τότε, είναι ένα από τα πολύ λίγα που προέκυψαν στο μέλλον σε σχέση με τον χρωματισμό γραφών και αποδείχτηκαν. Χρειάστηκε να περάσουν 153 χρόνια από τότε ,ραγδαία ανάπτυξη της τεχνολογίας , ένας μεγάλος αριθμός από επιστήμονες κατά την διάρκεια όλων αυτών των χρόνων, για να φτάσουμε το 2005 από τους Benjamin Werner και Georges Gonthier[1] να έχουμε την αποδοχή της απόδειξης (και κατ' επέκταση και του θεωρήματος) με την χρήση ηλεκτρονικών υπολογιστών που έγινε από τους Appel, Kenneth και Haken, Wolfgang το 1989[2].

Ιδιαίτερο ενδιαφέρον για εμάς έχει η εποχή από το 1970 και μετά. Αυτή ήταν η εποχή που το γενικευμένο πρόβλημα του χρωματισμού γραφών άρχισε να μελετάται ως αριθμητικό πρόβλημα. Το πρόβλημα του χρωματισμού αριθμού είναι από τα πιο γνωστά NP-complete προβλήματα και έχουν χρησιμοποιηθεί πολλές φορές στον παρελθόν για την απόδειξη άλλων τέτοιων προβλημάτων (τα NP-complete εξηγούνται παρακάτω). Ο Karp ήταν αυτός που απέδειξε ότι πρώτος ότι το πρόβλημα του vertex coloring - μία λίγο διαφορετική εκδοχή του coloring και που είναι και το κύριο πρόβλημα της παρούσας εργασίας- είναι NP-hard (Ο Karp έγραφε για αυτό και άλλα 21 προβλήματα το 1972). Εδώ αξίζει να αναφέρουμε ότι μόλις έναν χρόνο νωρίτερα το 1971 ο Stephen Cook είχε μόλις δώσει στο βιβλίο του[4] μία απόδειξη για το πρώτο NP-complete πρόβλημα που αποδείχτηκε η τάξη πολυπλοκότητας του, και είναι το Satisfiability Problem .

2.2.1 Περιγραφή των προβλημάτων χρωματισμού.

Χρωματισμός Κορυφών (vertex coloring) Στην γενικευμένη του μορφή το πρόβλημα χρωματισμού ονομάζεται πρόβλημα χρωματισμού κορυφών.

Ορισμός: Δεδομένου ενός μη κατευθυνόμενου γράφου $G=(V,E)$ όπου V είναι το σύνολο των κορυφών και E το σύνολο των ακμών που ενώνουν τις κορυφές, και δεδομένου ενός συνόλου χρωμάτων $x = \{x_1, \dots, x_n\}$, να βρεθεί χρωματισμός κορυφών τέτοιος ώστε δύο οποιεσδήποτε γειτονικές κορυφές να μην έχουν το ίδιο χρώμα.

Το παραπάνω πρόβλημα είναι πολύ γενικό και φαίνεται εύκολο στην επίλυση του. Μια γενική λύση είναι για παράδειγμα να χρωματίσουμε όλες τις κορυφές με διαφορετικό χρώμα και συνεπώς να έχουμε μέσα σε πολύ μικρό χρόνο μία σωστή επίλυση.

Το Vertex Coloring δεν είναι όμως τόσο απλό. Στην πραγματικότητα στον ορισμό του προστίθεται και ένας ακόμη παράγοντας. Θέλουμε να βρούμε έναν χρωματισμό ο οποίος να χρησιμοποιεί τον μικρότερο δυνατό αριθμό από χρώματα. Ο αριθμός αυτός ονομάζεται χρωματικός και η εύρεση ενός τέτοιου χρωματισμού είναι NP-hard πρόβλημα [3].

Το πρόβλημα αυτό έχει πολλές εφαρμογές για τις οποίες θα μιλήσουμε παρακάτω και είναι αυτό που θα μας απασχολήσει περισσότερο.

Χρωματισμός Κορυφών με βάρη (weighted vertex coloring) Μία επέκταση του Vertex Coloring εμπεριέχει βάρη στις διάφορες κορυφές. Ο στόχος μας εδώ είναι να ελαχιστοποιήσουμε τα βάρη του κάθε χρώματος. Το τελικό βάρος του κάθε χρώματος είναι το μεγαλύτερο από τα βάρη των κορυφών οι οποίες έχουν το χρώμα αυτό. Και εδώ θα πρέπει δύο οποιεσδήποτε γειτονικές κορυφές να έχουν το διαφορετικό χρώμα. Το πρόβλημα αυτό είναι όπως και το Vertex Coloring NP-hard.

Εύρεση χρωματικού αριθμού Όπως αναφέραμε νωρίτερα ορίζουμε ως χρωματικό αριθμό $\chi(G)$ ενός γράφου G , τον ελάχιστο αριθμό χρωμάτων που απαιτείται για τον χρωματισμό του ώστε δύο οποιεσδήποτε γειτονικές κορυφές να έχουν διαφορετικό χρώμα.

Η εύρεση του χρωματικού αριθμού (χωρίς να είναι απαραίτητο να βρούμε και κάποια λύση που να τον εφαρμόζει) έχει αποδειχτεί ότι είναι NP-Complete πρόβλημα[5].

Chromatic Polynomial Μία ακόμα επέκταση του προβλήματος χρωματισμού είναι και η εύρεση του αριθμού των δυνατών χρωματισμών ενός γράφου με το πολύ k χρώματα. Είναι εμφανές ότι το συγκεκριμένο πρόβλημα είναι ακόμα πιο δύσκολο από τα προηγούμενα και αυτό φαίνεται και από το γεγονός ότι δεν έχουμε ακόμα βρεί κάποιον καλό προσεγγιστικό αλγόριθμο για να το επιλύσουμε.

Το πρόβλημα ανήκει στα #P προβλήματα που είναι μία κλάση προβλημάτων ακόμα πιο δύσκολη από εκείνη των NP.

Χρωματισμός Ακμών (Edge Coloring) Όπως ακριβώς και με το Vertex Coloring, έτσι κι εδώ έχουμε ένα πρόβλημα με παρόμοιο ορισμό.

Ορισμός: Δεδομένου ενός μη κατευθυνόμενου γράφου $G=(V,E)$ όπου V είναι το σύνολο των κορυφών και E το σύνολο των ακμών που ενώνουν τις κορυφές, και δεδομένου ενός συνόλου χρωμάτων $x = \{x_1, \dots, x_n\}$, να βρεθεί χρωματισμός ακμών τέτοιος ώστε καμία κορυφή να μην έχει ακμές με το ίδιο χρώμα, αλλά και να ελαχιστοποιείται ο αριθμός των χρωμάτων που χρησιμοποιήσαμε.

Στην περίπτωση του Vertex Coloring ένας χρωματισμός με k χρώματα ονομάζεται K -coloring (k -χρωματισμός), ενώ στο Edge Coloring ονομάζεται k -edge-coloring.

Ο ελάχιστος αριθμός από χρώματα που απαιτείται ονομάζεται χρωματικός δείκτης (chromatic index). Το edge-coloring είναι NP-complete πρόβλημα [6].

K-coloring Πολύ συχνά στην θεωρία πολυπλοκότητας αλγορίθμων συναντάμε προβλήματα όπως: Είναι ο δεδομένος γράφος να χρωματιστεί με 4 χρώματα. Τέτοια ερωτήματα περιγράφουν ουσιαστικά το συγκεκριμένο πρόβλημα που στην γενική του περίπτωση ο αριθμός των χρωμάτων είναι k .

Πέραν των περιπτώσεων για $k = 1$ και $k = 2$, έχει αποδειχτεί ότι το πρόβλημα είναι NP-complete.

Άλλα προβλήματα Εκτός από τα παραπάνω προβλήματα, υπάρχει ένα μεγάλο σύνολο από εκδοχές τους, και εδώ θα αναφέρουμε μερικά επιγραμματικά:

- Χρωματισμός λίστας - List coloring
- Χρωματισμός-ακμών λίστας - List edge-coloring
- Πλήρης χρωματισμός - Complete coloring
- Δυναμικός χρωματισμός - Strong coloring
- Χρωματισμός-T -T-coloring
- Χρωματισμός Μονοπατιού - Path coloring
- Αθροιστικός χρωματισμός - Sum-coloring

Και πολλά άλλα.

2.2.2 Πρακτική εφαρμογή του χρωματισμού γράφων

Ο χρωματισμός γράφων έχει πολλές πρακτικές εφαρμογές σε διάφορους τομείς. Από πολύ απλές εφαρμογές της καθημερινότητάς μας όπως είναι το γνωστό σε όλους μας παιχνίδι Sudoku, μέχρι πολύ σημαντικές υπολογιστικές εφαρμογές όπως η απόδοση δεδομένων σε καταχωρητές. Ακολουθούν οι περιγραφές μερικών από αυτές. Πολλά από τα παραδείγματα περιγράφονται και από τον Marx (2004) [7].

Χρονικός προγραμματισμός (scheduling) Το βασικό παράδειγμα είναι το εξής:

Ας θεωρήσουμε ότι θέλουμε να προγραμματίσουμε την εκτέλεση ενός συνόλου από εργασίες. Κάποιες από αυτές τις εργασίες δεν είναι δυνατόν να πραγματοποιηθούν ταυτόχρονα (π.χ. τις αναλαμβάνει ο ίδιος εργάτης).

Έστω ο γράφος G τον οποίο θα ονομάσουμε γράφο συγχρούσεων. Οι κορυφές παριστάνουν τις εργασίες και οι ακμές υποδηλώνουν ότι οι δύο κορυφές που ενώνονται είναι εργασίες που δεν μπορούν να γίνουν ταυτόχρονα. Τα χρώματα είναι διαστήματα χρόνου ίσης αξίας (στην χειρότερη περίπτωση η μεγαλύτερη από τις αξίες αν δεν απαιτούν ακριβώς τον ίδιο χρόνο οι εργασίες).

Είναι εμφανές ότι οποιαδήποτε λύση μας δίνει μία διάρκεια π.χ. k διαστήματα χρόνου που θα χρειαστούν για να ολοκληρωθούν οι παραπάνω εργασίες. Προφανώς ψάχνουμε έναν τρόπο να ελαχιστοποιήσουμε τον χρόνο αυτό, πράγμα που είναι ισοδύναμο με το να βρούμε την χρωματικό αριθμό του γράφου. Πρακτικά παραδείγματα που απαιτούν τέτοιο χρονοπρογραμματισμό είναι ο προγραμματισμός πτήσεων (έχουμε k αεροπλάνα και n προγραμματισμένες πτήσεις και θέλουμε τον μικρότερο αριθμό αεροπλάνων) ο οποίος λύνεται βέλτιστα σε πολυωνυμικό χρόνο, το πρόβλημα των Bi-processor tasks (όπου κάθε εργασία απαιτεί δύο επεξεργαστές ταυτόχρονα) το οποίο είναι edge-coloring και είναι NP-hard και άλλα πολλά.

Κατανομή Καταχωρητών (Register Allocation) Από το πιο σημαντικά προβλήματα σε επίπεδο υλικού στον τομέα των ηλεκτρονικών υπολογιστών, είναι η κατανομή των καταχωρητών. Είναι γνωστό πως κάθε επεξεργαστής έχει έναν περιορισμένο αριθμό από καταχωρητές στους οποίους μπορεί να φορτώσει της μεταβλητές των προγραμμάτων. Ο στόχος είναι να μπορούμε να έχουμε όσο το δυνατόν περισσότερες μεταβλητές στους καταχωρητές για να ελαχιστοποιήσουμε τον χρόνο που χρειάζεται για να εκτελεστούν τα προγράμματα.

Όταν υπάρχουν παραπάνω «ζωντανές» μεταβλητές απ' ότι καταχωρητές έχουμε υπερχειλίσει καταχωρητών. Ο επεξεργαστής είναι υποχρεωμένος να μεταφέρει μεταβλητές από τους καταχωρητές στην κύρια μνήμη, πράγμα που μεταφράζεται σε χρονικό κόστος.

Για την ελαχιστοποίηση των μεταφορών στην κύρια μνήμη πρέπει να αφαιρούμε από τους καταχωρητές τις μεταβλητές εκείνες που θα χρησιμοποιηθούν λιγότερο στο μέλλον πράγμα δύσκολο να το γνωρίζουμε.

Ο αλγόριθμος του Chaitin [8] είναι ο πιο διαδεδομένος για την κατανομή καταχωρητών και χωρίζεται σε δύο κομμάτια.

- Στην πρώτη φάση δημιουργούνται εντολές με την θεώρηση ότι έχουμε άπειρο αριθμό από καταχωρητές. Ουσιαστικά εδώ αναγνωρίζουμε ποιες είναι οι μεταβλητές που απαιτούν καταχωρητές. Στο δεύτερο βήμα οι θεωρητικοί καταχωρητές αντικαθίστανται από πραγματικούς.
- Στο δεύτερο κομμάτι δημιουργείται ένας γράφος διαφωνιών. Οι θεωρητικοί καταχωρητές είναι οι κορυφές, και δύο κορυφές συνδέονται αν αυτές οι δύο μεταβλητές είναι «ζωντανές» σε ίδια χρονική περίοδο.

Από τα παραπάνω καταλαβαίνουμε ότι αν ο γράφος χρωματίζεται με k χρώματα τότε οι μεταβλητές μπορούν να αποθηκευτούν σε R καταχωρητές. Εδώ αξίζει να σημειώσουμε ότι μιας και το coloring είναι NP-hard πρόβλημα, ο Chaitin χρησιμοποιεί προσεγγιστικούς για να το λύσει.

Άλλες πρακτικές εφαρμογές Ο Hideo Ogawa [9] μας δίνει ένα παράδειγμα όπου το VCP (Vertex Coloring Problem) έχει εφαρμογή, ενώ άλλες εφαρμογές βρίσκουμε σε ζητήματα απόδοσης συχνοτήτων (όπου δύο κοντινοί σταθμοί απαγορεύεται να έχουν την ίδια συχνότητα και το εύρος που μας διατίθεται είναι περιορισμένο και στοιχίζει ανάλογα με το πόσο χρησιμοποιούμε)

2.3 Θεωρία πολυπλοκότητας και κλάσεις NP,P

2.3.1 Πολυπλοκότητα

Η θεωρία πολυπλοκότητας είναι ένα πολύ μεγάλο εύρος κομμάτι της θεωρίας των αλγορίθμων των ηλεκτρονικών υπολογιστών. Χωρίζεται σε δύο τομείς, ο ένας εκ των οποίων ασχολείται κυρίως με το κατά πόσο είναι δυνατόν να υπολογιστεί ένα πρόβλημα με την χρήση ηλεκτρονικού υπολογιστή, και ο δεύτερος με το κατά πόσο είναι εύκολο να υπολογιστεί η απάντηση όταν αυτό είναι εφικτό.

Το δεύτερο κομμάτι είναι αυτό που θα μας απασχολήσει στην παρούσα εργασία, μιας και θεωρούμε δεδομένο ότι το πρόβλημα Vertex Coloring μπορεί να επιλυθεί από ηλεκτρονικό υπολογιστή, ενώ αυτό που μας απασχολεί είναι ο χρόνος που απαιτείται.

Για να αναλύσουμε τον χρόνο ο οποίος χρειάζεται για να ολοκληρωθεί ένας αλγόριθμος, εκφράζουμε τον χρόνο ή τον χώρο ως συνάρτηση της εισόδου του προβλήματος. Αν για παράδειγμα θέλουμε να ταξινομήσουμε μία λίστα ακεραίων τότε για δύο αριθμούς αυτό είναι εύκολο. Το ερώτημα είναι για k αριθμούς πόσο πιο δύσκολο είναι όσο το k μεγαλώνει. Αν χρησιμοποιήσουμε τον αλγόριθμο ταξινόμησης merge-sort τότε η πολυπλοκότητα είναι $O(n \log n)$.

Ο συμβολισμός $O()$, απαντάται πολύ συχνά στην θεωρία πολυπλοκότητας και είναι ένας γενικός τύπος που δίνει την ασυμπτωτική συμπεριφορά του αλγορίθμου όσο αυξάνεται η είσοδος του ανεξάρτητα από το είδος της μηχανής που τον επιλύει και άλλους εξωτερικούς παράγοντες. Εδώ είναι σημαντικό να δώσουμε και κάποιους ορισμούς που θα μας φανούν χρήσιμοι παρακάτω.

- O : Συνήθως αναφέρεται στο άνω όριο από βήματα που απαιτούνται για την ολοκλήρωση του αλγορίθμου.
- Ω : Αναφέρεται στο κάτω όριο από βήματα που απαιτούνται για την ολοκλήρωση του αλγορίθμου.
- Θ : Αναφέρεται σε κοντινά άνω και κάτω όρια.

Συνηθίζεται όταν ασχολούμαστε με προβλήματα υπολογιστικής πολυπλοκότητας, να κατατάσσουμε τα διάφορα προβλήματα και αλγορίθμους σε κλάσεις πολυπλοκότητας. Οι συμμετέχοντες στην ίδια κλάση έχουν παρόμοιες απαιτήσεις σε πόρους, έχουν δηλαδή παρόμοιες πολυπλοκότητες. Στο σημείο αυτό θα δούμε και θα εξηγήσουμε τις πιο βασικές, και όσες αναφέρθηκαν νωρίτερα σε σχέση με το πρόβλημα χρωματισμού γραφών.

2.3.2 Κλάσεις Προβλημάτων P και NP

Συνηθίζεται όταν ασχολούμαστε με προβλήματα υπολογιστικής πολυπλοκότητας, να κατατάσσουμε τα διάφορα προβλήματα και αλγορίθμους σε κλάσεις πολυπλοκότητας. Οι συμμετέχοντες στην ίδια κλάση έχουν παρόμοιες απαιτήσεις σε πόρους, έχουν δηλαδή παρόμοιες πολυπλοκότητες. Παρακάτω θα περιγράψουμε τις αναγκαίες για την κατανόηση τις εργασίας μας.

Η κλάση προβλημάτων P Είναι η πιο βασική κλάση προβλημάτων στην θεωρία πολυπλοκότητας. Στην κλάση αυτή ανήκουν όλα εκείνα τα προβλήματα απόφασης που μπορούν να λυθούν από μία ντετερμινιστική μηχανή Turing (η μηχανή Turing είναι μια φανταστική μηχανή που μπορεί να επιλύσει όλα τα προβλήματα που επιλύει ένας ηλεκτρονικός υπολογιστής και χρησιμοποιείται για να αναλύουμε ανεξάρτητα από ιδιομορφίες μεμονωμένων μηχανών αλλά και ανεξάρτητα από τεχνολογική ανάπτυξη), σε πολυωνυμικό χρόνο.

Ο Alan Cobham (1965) υποστηρίζει ότι στην P ανήκουν εκείνα τα προβλήματα που επιλύονται ικανοποιητικά. Αν και αυτό δεν ισχύει πάντα, στις περισσότερες περιπτώσεις είναι σωστή υπόθεση.

Παραδείγματα της κλάσης αυτής είναι η εύρεση του μέγιστου κοινού διαιρέτη και ο γραμμικός προγραμματισμός (που χρησιμοποιείται για πολλές προσεγγίστες λύσεις πιο δύσκολων προβλημάτων).

Η κλάση προβλημάτων NP Η κλάση των NP προβλημάτων είναι η δεύτερη πιο γνωστή και περιλαμβάνει προβλήματα πιο περίπλοκα από αυτά της P. Ο ορισμός της είναι ο εξής: Στην κλάση NP ανήκουν όλα τα προβλήματα που μπορούν να λυθούν από μία μη ντετερμινιστική μηχανή Turing σε πολυωνυμικό χρόνο.

Επιπλέον ορισμοί/χαρακτηριστικά είναι:

- Η κλάση των προβλημάτων απόφασης για τα οποία υπάρχει μη ντετερμινιστικός αλγόριθμος που τα επιλύει
- Η κλάση προβλημάτων που είναι λίγο πιο «δύσκολα» από τα P
- Σε γενικές γραμμές ένας εκθετικός αριθμός από υπο-προβλήματα που επιλύονται σε πολυωνυμικό χρόνο
- Για ένα αλγόριθμο που ανήκει στην NP μπορούμε να πούμε ότι υπάρχει γνωστό άνω φράγμα στην υπολογιστική του πολυπλοκότητα

Ας δούμε όμως πρώτα τι είναι μια μη ντετερμινιστική μηχανή Turing. Είναι μια φανταστική μηχανή που μπορούμε να την φανταστούμε ως εξής. Έχουμε για παράδειγμα το απλό πρόβλημα της εύρεσης του μέγιστου κοινού διαιρέτη. Έστω ένα πρόβλημα που μας λέει να βρούμε τον μέγιστο κοινό διαιρέτη εκθετικά σε αριθμό φορές. Π.χ. Έστω n αριθμοί να βρούμε όλους τους μέγιστους κοινούς διαιρέτες όλων των συνδυασμών των n αριθμών. Χοντρικά μία μη ντετερμινιστική μηχανή Turing μπορεί ταυτόχρονα να υπολογίσει τα αποτελέσματα όλων αυτών των συνδυασμών.

Η κλάση P εμπεριέχεται προφανώς στην NP. Η NP όμως περιέχει και άλλα προβλήματα, τα δυσκολότερα των οποίων είναι τα NP-complete για τα οποία δεν υπάρχει ακόμα ντετερμινιστικός αλγόριθμος που να τα επιλύει σε πολυωνυμικό χρόνο. Σημαντικό εδώ είναι ότι για να ανήκει ένα πρόβλημα στην κλάση αυτή θα πρέπει να υπάρχει αλγόριθμος που να μπορεί να επιλύσει οποιοδήποτε στιγμιότυπο σε πολυωνυμικό χρόνο.

Καλύτερα όμως να πάρουμε για παράδειγμα το 4-coloring πρόβλημα. Ένας αλγόριθμος που μπορεί να το επιλύσει είναι ο εξής. Πάρε όλους τους δυνατούς χρωματισμούς και δες αν υπάρχουν δυο κορυφές γειτονικές με το ίδιο χρώμα. Καταλαβαίνουμε εδώ ότι ο αριθμός διαφορετικών χρωματισμών είναι εκθετικά μεγάλος. Κάθε δεδομένος χρωματισμός γίνεται όμως να επιβεβαιωθεί σε πολυωνυμικό χρόνο.

NP-Hardness Ένα πρόβλημα (ας το ονομάσουμε Π), είναι NP-hard αν η ύπαρξη ενός αλγορίθμου πολυωνυμικού χρόνου υπονοεί την ύπαρξη αλγορίθμου πολυωνυμικού και πάλι χρόνου για οποιοδήποτε πρόβλημα ανήκει στην κλάση των NP.

Άρα συμπεραίνουμε:

Π είναι NP-hard $\iff$ Αν το Π επιλύεται σε πολυωνυμικό χρόνο τότε $P=NP$

Το πιο σημαντικό συμπέρασμα που μπορούμε να βγάλουμε από τον παραπάνω ορισμό, είναι ότι στην κλάση αυτή των προβλημάτων υπάρχουν όλα τα προβλήματα τα οποία είναι τουλάχιστον τόσο δύσκολα (σε σχέση με την πολυπλοκότητά τους φυσικά) όσο τα δυσκολότερα προβλήματα της κλάσης των NP προβλημάτων (που όπως φαίνεται από τα παραπάνω είναι τα NP-Complete). Σημαντικό επίσης είναι

να μην θεωρήσουμε ότι όλα τα NP-hard προβλήματα είναι και NP-complete . Το αντίθετο συμβαίνει πάντα.

Ενδιαφέρον ακόμα προκαλεί και ο τρόπος με τον οποίο αποδεικνύουμε ότι ένα πρόβλημα είναι πράγματι NP-hard. Ο Stephen Cook το 1971 [11] απέδειξε ότι το πρόβλημα SAT (satisfiability) είναι NP-hard. Βάση της απόδειξής τους ξεκίνησαν όλες οι αποδείξεις για τέτοια προβλήματα, με την μέθοδο της αναγωγής. Ουσιαστικά αυτό που κάνουμε όταν αποδεικνύουμε κάτι τέτοιο, είναι να δείχνουμε ότι αν επιλυθεί το πρόβλημά μας σε πολυωνυμικό χρόνο τότε επιλύεται και το NP-hard πρόβλημα στο οποίο κάνουμε την αναγωγή σε πολυωνυμικό χρόνο, αλλά και το αντίστροφο.

Γνωρίζουμε πολλά προβλήματα που είναι NP-hard και εδώ θα αναφέρουμε επιγραμματικά το Sat , 3-sat, Hamiltonian cycle, Vertex Cover, Clique και πολλά άλλα.

2.4 Προσεγγιστικοί αλγόριθμοι και Θεωρία παιγνίων

Μέχρι τώρα μιλήσαμε και αναλύσαμε εκτενώς το θέμα του vertex coloring και της πολυπλοκότητας που έχει για να επιλυθεί. Είδαμε ακόμα και τις πολλές χρησιμότητες που έχει σε διάφορους τομείς στην επιστήμη των υπολογιστών αλλά και έξω από αυτήν. Βλέπουμε λοιπόν ότι οδηγούμαστε σε ένα αδιέξοδο ως προς το πώς θα μπορέσουμε να δώσουμε μία εύκολη λύση για το πρόβλημα.

Την απάντηση στο ερώτημα αυτό μας δίνουν οι προσεγγιστικοί αλγόριθμοι, και θα δούμε έναν από αυτούς που χρησιμοποιεί την θεωρία παιγνίων για να δώσει λύσεις στο συγκεκριμένο πρόβλημα. Την λύση αυτή θα την χρησιμοποιήσουμε αργότερα για να δούμε την συμπεριφορά της σε περιβάλλοντα που μας ενδιαφέρουν, καθώς και θα προτείνουμε διάφορες παραλλαγές μικρότερες σε απόδοση αλλά και σε ενέργεια, που σε κατανεμημένα δίκτυα αισθητήρων είναι εξίσου σημαντική.

Προσεγγιστικοί αλγόριθμοι για την επίλυση NP-hard προβλημάτων Είναι εύκολο στο σημείο αυτό για τον αναγνώστη να κατανοήσει ότι δεν μπορούμε να λύσουμε τα NP-hard προβλήματα. Για να το κάνουμε ακόμα πιο κατανοητό ,ας φανταστούμε ένα γράφο που θέλουμε να χρωματίσουμε. Αν "τρέξουμε" έναν κανονικό-συμβατικό αλγόριθμο ο οποίος έχει πολυπλοκότητα $O(n^n)$, τότε για ένα όχι πολύ μεγάλο γράφο των 1000 κορυφών, ο αριθμός σε πράξεις δεν χωράει σε μία μόνο γραμμή. Ακόμα και σύγχρονοι υπέρ-υπολογιστές θα δυσκολεύονταν πολύ να το επιλύσουν.

Υπάρχουν όμως πολλές εφαρμογές για τον αλγόριθμο χρωματισμού, αλλά και για πολλά άλλα προβλήματα που ανήκουν στην κατηγορία. Υπάρχουν ακόμα και προβλήματα που επιλύονται σε πολυωνυμικό χρόνο έχουν όμως πολυπλοκότητα της τάξης του $O(n^{1000})$, οπότε καταλαβαίνουμε ότι για μεγάλα n δεν είναι πρακτικές οι λύσεις τους.

Όλα τα παραπάνω έχουν οδηγήσει σε δύο νέες κατηγορίες αλγορίθμων που χρησιμοποιούνται και ερευνώνται κατά κόρον από τους επιστήμονες της επιστήμης των υπολογιστών. Είναι οι ευριστικοί αλγόριθμοι heuristics και οι προσεγγιστικοί.

Ευριστικοί Αλγόριθμοι: Μπορούν να δημιουργήσουν μία "αποδεκτή" λύση σε αποδεκτό χρόνο αλλά δεν υπάρχει απόδειξη ότι είναι η σωστή. Γενικά προσπαθούμε να δημιουργούμε αλγορίθμους με αποδεδειγμένα "καλά" αποτε-

λέσματα για όλες τις περιπτώσεις και που να τρέχουν σε αποδεδειγμένα χαμηλούς χρόνους. Οι ευριστικοί μπορεί να αγνοούν τον ένα ή και τους δύο αυτούς όρους. Χρησιμοποιούνται κυρίως όταν δεν υπάρχει τρόπος να βρούμε βέλτιστη λύση.

Προσεγγιστικοί Αλγόριθμοι: Ουσιαστικά είναι οι αλγόριθμοι που χρησιμοποιούμε όταν ψάχνουμε να βρούμε τη βέλτιστη λύση από το σύνολο των αποδεκτών λύσεων. Τα προβλήματα αυτά τα ονομάζουμε προβλήματα βελτιστοποίησης. Στα NP-hard προβλήματα δεν μπορούμε να βρούμε ακριβώς την βέλτιστη λύση. Όπως όμως διαβάζουμε στο βιβλίο του Vijay Vajirani - Approximation Algorithms μας δίνουν πατήματα για να βρούμε μία λύση πολύ κοντά στην βέλτιστη.

Επιπλέον μας λέει και το εξής πολύ σημαντικό. Ο σχεδιασμός τέτοιων αλγορίθμων δεν διαφέρει πολύ από τον σχεδιασμό ενός οποιουδήποτε άλλου αλγορίθμου, στον οποίο αναλύουμε την δομή του προβλήματος και ψάχνουμε τον τρόπο λειτουργίας ώστε να βρούμε την βέλτιστη λύση (αν το πρόβλημα ανήκει στην P).

Θα ονομάσουμε NP-optimisation προβλήματα, αυτά για τα οποία μπορούμε να βρούμε προσεγγιστική λύση και έχουν τα παρακάτω χαρακτηριστικά.

- Είναι προβλήματα μεγιστοποίησης ή ελαχιστοποίησης
- Υπάρχει πολυωνυμικός τρόπος ώστε να γνωρίζουμε την ορθότητα μίας λύσης, και την τιμή που πρέπει να πάρουμε από την λύση αυτή.
- Ορίζουμε επίσης ως $OPT_P(I)$ το αποτέλεσμα που δίνει η βέλτιστη (optimal) λύση. Εδώ το P είναι το πρόβλημα και I είναι ένα αποδεκτό στιγμιότυπο του.
- Ένας προσεγγιστικός αλγόριθμος A, για ένα πρόβλημα P, παράγει σε πολυωνυμικό χρόνο μία αποδεκτή λύση κοντά στο $OPT_P(I)$. Όταν αναφερόμαστε στο κοντά εννοούμε μία τιμή μεταξύ καθορισμένων και βέβαιων ορίων.
- Για να βρούμε την αποδοτικότητα του αλγορίθμου μας, συγκρίνουμε την λύση μας με την βέλτιστη για την χειρότερη περίπτωση.

Για να γίνει η σύγκριση, πρέπει φυσικά να έχουμε τρόπο να βρούμε την βέλτιστη. Αυτό στα προβλήματα που μελετάμε είναι φυσικά αδύνατο. Για αυτό τον λόγο βρίσκουμε ένα κάτω όριο για την βέλτιστη τιμή (που είναι σίγουρα μεγαλύτερο ή ίσο με την τιμή που ψάχνουμε) και το συγκρίνουμε με αυτό. Αν θεωρήσουμε $f(x)$ το κόστος του προσεγγιστικού και OPT το κόστος του βέλτιστου, τότε κάθε επιτυχής ανάλυση καταλήγει ως εξής:

$$m = \max \left(\frac{OPT}{f(y)}, \frac{f(y)}{OPT} \right)$$

Ουσιαστικά το m μας δίνει τον συντελεστή απόδοσης. Αν π.χ. είναι 2 και το πρόβλημα είναι ο χρωματισμός, αυτό σημαίνει ότι στην χειρότερη περίπτωση ο αλγόριθμός θα μας δώσει λύση με τα διπλάσια χρώματα (αν ήταν 2 χρώματα θα μας δώσει λύση με 4), το οποίο δεν είναι κακό αποτέλεσμα, αν σκεφτεί κανείς ότι δεν μπορούμε να βρούμε καλύτερο (στην πραγματικότητα δεν έχουμε τόσο καλό προσεγγιστικό για το coloring ακόμα) και ότι αν μελετάμε μεγάλους γράφους δεν θα είναι ιδιαίτερα αισθητή η διαφορά.

Οι μεθοδολογίες και παραδείγματα (πέραν της λύσης με θεωρία παιγνίων που ακολουθεί), δεν κρίνεται σκόπιμο να παρουσιαστούν εδώ. Ο αναγνώστης μπορεί να ανατρέξει στο [12] για περισσότερες πληροφορίες.

Η λύση με χρήση θεωρίας παιγνίων Πριν περιγράψουμε τον ακριβή αλγόριθμο, πρέπει να αναφερθούμε επιγραμματικά στην θεωρία παιγνίων και την χρησιμότητά της στην εύρεση αλγορίθμων.

Θεωρία Παιγνίων Η θεωρία παιγνίων είναι ένας ευρύς κλάδος των εφαρμοσμένων μαθηματικών, με πολλές εφαρμογές στα μαθηματικά, υπολογιστές, βιολογία, πολιτικές επιστήμες κ.α. Είναι μια προσπάθεια να "μαθηματικοποιήσουμε" τις συμπεριφορές σε στρατηγικής σημασίας καταστάσεις, στις οποίες η επιτυχία ενός, εξαρτάται άμεσα από το τι επιλογές θα κάνει κάποιος άλλος. Ένα παράδειγμα από την καθημερινότητά μας είναι οι γνωστές σε όλους μας δημοπρασίες. Σε μία δημοπρασία, η τελική τιμή που θα πληρώσουμε για να αγοράσουμε ένα προϊόν εξαρτάται άμεσα από τα ποσά που διατίθενται να δώσουν οι "συμπαίχτες" μας. Με την θεωρία παιγνίων και δεδομένων χρημάτων που μπορούμε-θέλουμε να δώσουμε, υπάρχουν κινήσεις (τόσο για εμάς όσο και για τους υπόλοιπους) που είτε θα συμφέρουν εμάς περισσότερο και λιγότερο τους αντιπάλους είτε το αντίστροφο, ή στην δίκαιη περίπτωση θα συμφέρουν εξίσου και τους δύο.

Αυτή η τελευταία περίπτωση είναι αυτό που αναφέρεται στην θεωρία παιγνίων ως ισορροπία Nash . Αξίζει εδώ να αναφέρουμε τον John Nash από τον οποίο πήρε το όνομά της η ισορροπία αυτή για τα ανταγωνιστικά παιχνίδια. Σε γενικές γραμμές σε μία ισορροπία Nash με δύο παίκτες, θεωρούμε ότι ο καθένας παίρνει τις καλύτερες δυνατές αποφάσεις γνωρίζοντας τι απόφαση θα πάρει ο αντίπαλός του. Ουσιαστικά δεν χρειάζεται κανένας να αλλάξει στρατηγική κατά την διάρκεια του παιχνιδιού αφού του αντιπάλου του θα μείνει ίδια και συνεπώς παίρνει ήδη τις καλύτερες δυνατές αποφάσεις. Εξ'άλλου στο παραπάνω πρόβλημα ο παίχτης που δίνει περισσότερα είναι αυτός που θα κερδίσει, το θέμα είναι να μην δώσει παραπάνω από όσα θα μπορούσε.

Βασίζόμενοι στην θεωρία αυτή, οι Panagiotis N. Panagopoulou, Paul G. Spirakis [13] πρότειναν τον αλγόριθμο που ακολουθεί, απέδειξαν ότι οδηγεί το παιχνίδι σε Nash Equilibrium και έδωσαν πολύ καλά άνω και κάτω όρια ως προς την αποτελεσματικότητα του αλγορίθμου σε σχέση με τον βέλτιστο.

Το παιχνίδι χρωματισμού Ορισμός: Ξεκινώντας από έναν χρωματισμό που ικανοποιεί την συνθήκη να μην υπάρχουν γειτονικές κορυφές με το ίδιο χρώμα (για παράδειγμα κάθε κορυφή παίρνει διαφορετικό χρώμα), και αρχίζοντας το παιχνίδι από τυχαία κορυφή, κάθε κόμβος (ένας την κάθε φορά με την σειρά) επιλέγει ένα χρώμα ώστε να παραμένει αποδεκτά χρωματισμένος ο γράφος αλλά και να έχει μεγαλύτερη χρωματική πληθικότητα από αυτήν που είχε. Ο αλγόριθμος τελειώνει όταν δεν υπάρχει κορυφή που να μπορεί να κάνει μετάβαση και να αυξηθεί η χρωματική της πληθικότητα.

Οι Panagopoulou, Spirakis επέλεξαν την ανάλυση με θεωρία παιγνίων εξαιτίας της φυσικής αντιστοιχίας που έχει το τοπικό βέλτιστο της μεθόδου με την ισορροπία Nash ενός καλά ορισμένου στρατηγικού παιγνίου. Ποίο συγκεκριμένα, στον αλγόριθμο αυτό οι κορυφές του γράφου $G(V,E)$ είναι οι παίκτες. Ο κάθε παίχτης με την σειρά του έχει να αποφασίσει από ένα σύνολο ενεργειών, που κάθε μια από αυτές είναι η μετάβαση σε ένα από τα χρώματα του γράφου και είναι δηλαδή το πολύ $|N|$ σε αριθμό. Σε ένα στιγμιότυπο το κέρδος που έχει μία κορυφή u , είναι μηδενικό αν το χρώμα το έχει κάποιος από τους γείτονές της. Διαφορετικά το κέρδος της είναι ίσο με τον αριθμό των κορυφών που έχουν ήδη επιλέξει το χρώμα αυτό. Σε μία "καθαρή" Nash ισορροπία κανένας από τους παίκτες-κορυφές

δεν μπορεί να βελτιώσει παραπάνω το κέρδος της με προσωπική μόνο αλλαγή. Αυτό φυσικά σημαίνει ότι αν υπάρχει η ισορροπία αυτή ο αλγόριθμος τερματίζει και μάλιστα σίγουρα με αποδεκτό χρωματισμό. Αυτό γιατί δεν μπορεί κανένας γράφος να πάρει χρώμα γείτονά του από την στιγμή που με το δικό του αρχικό χρώμα έχει κέρδος ένα και με χρώμα γείτονά του θα αποκτήσει κέρδος μηδέν.

Συμπεραίνουμε από τα παραπάνω ότι ο προτεινόμενος αλγόριθμος δίνει σίγουρα χρωματισμό και άρα έχουμε πάντα αποτέλεσμα. Το ερώτημα φυσικά παραμένει ως προς το πόσο 'καλός' είναι σε σχέση με τον χρωματικό αριθμό που επιτυγχάνει. Με την χρήση των εργαλείων που δίνει η θεωρία παιγνίων οι συγγραφείς απέδειξαν ότι είναι ικανοποιητικά καλός, καλύτερος μάλιστα σε σχέση με οτιδήποτε είχε προταθεί μέχρι τότε.

Αποτελέσματα του αλγορίθμου Η ανάλυση του μοντέλου οδήγησε στα παρακάτω συμπεράσματα:

- Πρώτο και κυριότερο, σε οποιαδήποτε ακολουθεία από 'εγωιστικές' κινήσεις, αν ξεκινήσουμε από κάποιο αποδεκτό χρωματισμό, θα οδηγήσει σε $O(n \cdot \alpha(G))$ κινήσεις σε ισορροπία. Το $\alpha(G)$ είναι ο βαθμός ανεξαρτησίας του γράφου.
- Οποιαδήποτε Nash ισορροπία του παιχνιδιού, είναι κανονικός χρωματισμός που χρησιμοποιεί το πολύ k χρώματα, φραγμένο από πάνω από όλα τα γενικώς γνωστά άνω φράγματα για τον χρωματικό αριθμό. Πιο συγκεκριμένα έστω $n, m, \chi(G)$ και $\omega(G)$ είναι ο αριθμός από κορυφές, ακμές, χρωματικός, κλίκας και βαθμού αντίστοιχα. Αν $\Delta_2(G)$ είναι ο μεγαλύτερος βαθμός που μία κορυφή v μπορεί να έχει υπό την προϋπόθεση ότι η κορυφή έχει τουλάχιστον έναν γείτονα με βαθμό όχι μικρότερο από αυτόν της ίδιας, φαίνεται ότι το k ικανοποιεί την σχέση:

$$k \leq \{\Delta_2(G) + 1, \frac{n + \omega(G)}{2}, n - \alpha(G), \frac{1 + \sqrt{1 + 8m}}{2}\}.$$

- Υπάρχουν περιπτώσεις (μιας και ο χρωματικός αριθμός δεν υπολογίζεται σε πολυωνυμικό χρόνο), όπου ο αλγόριθμος παράγει χρώματα πολύ περισσότερα του χρωματικού. Τέτοια περίπτωση είναι οι bitparite γράφοι. Παρόλα αυτά ο αλγόριθμος δεν είναι απαραίτητο να έχει τόσο άσχημα αποτελέσματα ($\frac{n+1}{2}$) σε σύγκριση με τον χρωματικό που είναι 2. Στην πραγματικότητα με μεγάλη πιθανότητα ο αλγόριθμος θα δώσει $\chi(G)=2$. Επιπλέον οι συγγραφείς προτείνουν την χρήση του σε έτοιμους χρωματισμούς από άλλους προσεγγιστικούς για την βελτίωση των αποτελεσμάτων τους.

Διαφορετικές προσεγγίσεις Παλαιότερα, αρκετοί προσεγγιστικοί αλγόριθμοι έχουν προταθεί για το πρόβλημα του χρωματισμού κορυφών. Ο καλύτερος από αυτούς που γνωρίζουμε έχει λόγο απόδοσης $O(n(\log \log n)^2 / (\log n)^3)$. Είναι γνωστό ότι ο χρωματικός αριθμός δεν γίνεται να προσεγγιστεί σε βαθμό μεγαλύτερο από $\Omega(n^{1-\epsilon})$, εκτός και αν η κλάση NP είναι υποσύνολο της κλάσης co-RP. Υπάρχουν πολλοί ευριστικοί αλγόριθμοι. Κανένας όμως δεν πιστοποιεί αποτελέσματα τόσο καλά όσο του παραπάνω αλγορίθμου. Επίσης έχουν μελετηθεί και παιχνίδια, αλλά είναι με μόνο δύο παίκτες που αντιμετωπίζουν ο ένας τον άλλο, και είναι σε τελείως διαφορετική φιλοσοφία.

Κεφάλαιο 3

Κατανεμημένα Συστήματα-Δίκτυα αισθητήρων

3.1 Εισαγωγή

Στο πρώτο κεφάλαιο μιλήσαμε για το πρόβλημα χρωματισμού και τις κλάσεις πολυπλοκότητας. Στο σημείο αυτό έχουμε ολοκληρώσει την θεωρία που χρειαζόμαστε σχετικά με τους αλγόριθμους και θα μιλήσουμε για δύο άλλα πολύ σημαντικά θέματα που μας απασχολούν στην εργασία αυτή.

Το πρώτο είναι γενικά τα κατανεμημένα συστήματα, τα οποία αναφέρονται ουσιαστικά σε μεμονωμένα συστήματα υπολογιστών, τα οποία συνδέονται μεταξύ τους για να πραγματοποιήσουν έναν κοινό σκοπό είτε γιατί ο σκοπός αυτός απαιτεί από όλους ξεχωριστά κάποια πληροφορία (για παράδειγμα σε εμάς απαιτείται το χρώμα) ή γιατί το καθένα ξεχωριστά δεν έχει την υπολογιστική δύναμη να το εκπληρώσει σε ικανοποιητικό χρόνο.

Επειτα θα δούμε ένα τέτοιο σύνολο συστημάτων, που είναι τα δίκτυα αισθητήρων. Τα συγκεκριμένα δίκτυα είναι μικροί σε μέγεθος υπολογιστές, που ο καθένας ξεχωριστά έχει μικρή υπολογιστική δύναμη και μνήμη, αλλά όλοι μαζί μπορούν να συνδυαστούν για πολύ σημαντικές εφαρμογές. Θα δούμε ότι χρησιμοποιούνται ήδη σε πολλές εφαρμογές και ελπίζουμε στο μέλλον να επεκταθούν σε ακόμη περισσότερες. Τα δίκτυα αυτά μας αφορούν ιδιαίτερα στην έρευνά μας, μιας και θα είναι το περιβάλλον στο οποίο θα δοκιμάσουμε την αποδοτικότητα των αλγόριθμων μας.

3.2 Κατανεμημένα Συστήματα

Η επιστήμη των κατανεμημένων συστημάτων, ασχολείται με τον υπολογισμό απαντήσεων σε μεγάλα προβλήματα, χωρίζοντας τα σε μικρότερα και τελικά, συγκεντρώνοντας τα αποτελέσματα από αυτά, τα συνδυάζει και δίνει απαντήσεις. Αρχικά αναφερόταν σε ηλεκτρονικούς υπολογιστές που ήταν απομακρυσμένοι γεωγραφικά ο ένας από τον άλλο. Σήμερα όμως αναφερόμαστε σε κατανεμημένα συστήματα ακόμα και με την έννοια των διαφορετικών διεργασιών μέσα στο ίδιο σύστημα. Επιπλέον κατανεμημένα συστήματα μπορούν να αποτελούνται από υπολογιστές με άσχετους μεταξύ τους σκοπούς, με στόχο να διαχειρίζονται πόρους που τους μοιράζονται.

Γενικότερα όμως τα συστήματα αυτά συνήθως δεχόμαστε ότι έχουν κάποια συγκεκριμένα χαρακτηριστικά.

- Αποτελούνται από αρκετές αυτόνομες οντότητες, κάθε μία από της οποίες έχει δική της μνήμη.
- Οι οντότητες αυτές επικοινωνούν μεταξύ τους χρησιμοποιώντας μηνύματα.

Πέρα από τα χαρακτηριστικά αυτά, έχουν συνήθως και κάποιες ή όλες από τις παρακάτω ιδιότητες:

- Το συνολικό σύστημα πρέπει να μπορεί να ανέχεται και να ανακάμπτει από σφάλματα ενός ή περισσότερων συμμετεχόντων.
- Η δομή του συστήματος, με την έννοια του hardware , γεωγραφικής τοποθεσίας, τοπολογίας, αριθμός συμμετεχόντων κ.λ.π. δεν είναι απαραίτητο να είναι γνωστά από την αρχή, αλλά μπορούν να αλλάξουν και κατά την διάρκεια της εκτέλεσης των αλγορίθμων.
- Κάθε υποσύστημα έχει περιορισμένη γνώση του συνολικού συστήματος. Μπορεί να γνωρίζει μόνο ένα κομμάτι της εισόδου.

3.2.1 Ιστορία των κατανεμημένων

Η ιδέα των διαδικασιών (processess) οι οποίες θα επικοινωνούσαν με την αποστολή μηνυμάτων έχει ξεκινήσει από το 1960. Από το 1970 και μετά άρχισαν τα πρώτα μεγάλα τέτοιου είδους συστήματα, όπως το ethernet κ.λ.π. Το γνωστό σε όλους ARPANET που είναι ο προκάτοχος του σημερινού internet είναι όπως και ο απόγονός του κατανεμημένο σύστημα.

3.2.2 Πρακτικές Εφαρμογές

Το καλύτερο παράδειγμα πρακτικής εφαρμογής των παραπάνω συστημάτων είναι πιθανότητα ο παγκόσμιος ιστός. Είναι ένα σύστημα που αποτελείται από πολλές ετερογενείς και διαφορετικού σκοπού ηλεκτρονικούς υπολογιστές, και παρόλα αυτά απαρτίζει το μεγαλύτερο σύστημα διακίνησης πληροφοριών σήμερα.

Πέραν από το Internet όμως υπάρχουν πάρα πολλοί τομείς που χρησιμοποιούν κατανεμημένα συστήματα. Η τηλεφωνία είναι ένας από αυτούς, τόσο η κινητή όσο και η σταθερή, στην οποία καταλαβαίνουμε ότι η εφαρμογή τέτοιων συστημάτων ήταν επιβεβλημένη από νωρίς εξαιτίας της γεωγραφικής έκτασης που πρέπει να καλύπτουν οι φορείς παροχής τέτοιων υπηρεσιών.

Κατανεμημένες βάσεις δεδομένων χρησιμοποιούνται από βιβλιοθήκες παγκοσμίως. Κατανεμημένα συστήματα από την καθημερινότητά μας είναι και τα δίκτυα διαμοιρασμού αρχείων peer-to-peer όπου ο κάθε υπολογιστής που συμμετέχει αποτελεί κομμάτι της κατανεμημένης εργασίας που επιτελείται. Στα peer-to-peer συστήματα η κατανομή εργασιών είναι πολλές φορές αυτή που προστατεύει τα προσωπικά δεδομένα του κάθε χρήστη (ο κάθε κόμβος μπορεί να μεταφορτώνει αρχεία για να τα μεταβιβάσει και όχι για να τα κρατήσει). Επιπλέον τα παιχνίδια στο διαδίκτυο είναι κατανεμημένα μόνο που σε αντίθεση με τα peer-to-peer έχουν ιδιαίτερες απαιτήσεις σε εξυπηρετητές για να λειτουργήσουν και να συντονιστούν. Επιπλέον υπάρχει 'σχεδόν' single point of failure στην περίπτωση που χαθεί ο server .

Πέραν της γενικής χρησιμότητας, όχι μόνο στις παραπάνω περιπτώσεις αλλά και σε τράπεζες (το εννιαίο τραπεζικό σύστημα συναλλαγών), στις αεροπορικές εταιρίες και άλλα πολλά, υπάρχουν πολλά ανοιχτά έργα που πραγματοποιούνται καθημερινά σε ολόκληρο τον κόσμο με την βοήθεια χρηστών που θέλουν να συμμετέχουν σε αυτά χρησιμοποιώντας απλώς την ισχύ των επεξεργαστών τους όταν αυτοί δεν κάνουν τίποτα. Παρακάτω αναφέρουμε τα πιο γνωστά:

- **Folding@home.** Σημαίνει αναδίπλωση στο σπίτι, και αναφέρεται στην αναδίπλωση των πρωτεϊνών, ένα πολύ σημαντικό κομμάτι της βιολογίας που παραμένει μυστήριο μέχρι σήμερα. Με τους συμμετέχοντες στο πρόγραμμα αυτό όμως σήμερα έχουμε 5 petaFlops ισχύ για να το μελετήσουμε. Είναι το μεγαλύτερο από τέτοιου είδους εγχειρήματα.
- **BOINC .** Είναι μία ανοιχτή δομή κατασκευασμένη από το πανεπιστήμιο Berkeley , και χρησιμοποιείται για επιστημονικούς υπολογισμούς. Είναι μπορεί να πει κανείς ένας υπερ-υπολογιστής (με 4 petaFlops ισχύ) που μπορεί να τον χρησιμοποιήσει όποιος τον χρειάζεται. Ξεκίνησε σαν πλατφόρμα για την διαχείριση του SETI@home .
- **SETI@home.** Ψάξε για εξωγήινους από το σπίτι σημαίνει πρακτικά το όνομα αυτό και είναι το πιο γνωστό από τα κατανεμημένα συστήματα.
- **Einstein@home MilkyWay@home** και πολλά άλλα ακόμα, εκμεταλλεύονται τις δυνατότητες που προσφέρουν τα κατανεμημένα συστήματα και ιδιαίτερα την μεγάλη εύρους ανεκμετάλλευτη ισχύ που υπάρχει από τα εκατομμύρια των προσωπικών υπολογιστών.

Από την γενικότερη ιδέα των δικτύων που κάνει ολοκληρωμένη εργασία κατανέμοντας την προέκυψαν και τα δίκτυα αισθητήρων που από μόνοι τους αποτελούν ένα ολόκληρο κομμάτι στην σύγχρονη επιστήμη των υπολογιστών με εφαρμογές σε πολλούς τομείς.

3.2.3 Coloring σε κατανεμημένα δίκτυα

Παρά το ότι τα τελευταία χρόνια άρχισε να υπάρχει ιδιαίτερο ενδιαφέρον στα κατανεμημένα συστήματα, ήταν σίγουρο από την αρχή ότι η θεωρία γράφων θα έπαιζε σπουδαίο ρόλο στα δίκτυα αυτά. Επιπλέον τα δίκτυα αισθητήρων που απαιτούν πολύ καλό χρόνο-προγραμματισμό, εξαιτίας της περιορισμένης μπαταρίας που έχουν, επωφελούνται παραπάνω από τους αλγορίθμους χρωματισμού. Για αυτούς τους λόγους έχει υπάρξει σε έναν βαθμό έρευνα πάνω στον τομέα.

Αλγόριθμοι ενός γύρου και ενός γύρου με επαναλήψεις Οι Fabian Kuhn και Roger Wattenhofer[14], προτείνουν δύο αλγόριθμους για να λύσουν το πρόβλημα του χρωματισμού. Στον πρώτο βασίζονται πιο πολύ σε μια "σχετικά" καλή (πολύ χειρότερη βέβαια από τις περισσότερες προσεγγίσεις), αλλά και πάρα πολύ γρήγορη και καθόλου ενεργοβόρα τακτική. Σχεδιάζουν έναν αλγόριθμο ο οποίος σε έναν μόνο γύρο θα ολοκληρωθεί και θα δώσει χρωματισμό. Ο αλγόριθμος ξεκινά με την προϋπόθεση ότι έχει ήδη κάποιον χρωματισμό (και η απόδοσή του εξαρτάται από τα χρώματα που τον αποτελούν) και με μόνο ένα μήνυμα από κάθε κόμβο, οι εκάστοτε κόμβοι αποφασίζουν σε τι χρώμα θα μεταβούν.

Φυσικά ο παραπάνω αλγόριθμος δεν είναι δυνατόν να προσφέρει καλά αποτελέσματα με τόσο λίγη πληροφορία, για αυτόν το λόγο, προτείνεται ένας δεύτερος αλγόριθμος που ουσιαστικά είναι ο αλγόριθμος σε έναν γύρο επαναλαμβανόμενος αρκετές φορές. Ο αλγόριθμος αυτός φαίνεται να τα καταφέρνει καλύτερα από τον προηγούμενο, και έδωσε την δυνατότητα να παίρνουν όσο το δυνατότερο καλά για τον χρήστη αποτελέσματα.

Ο αλγόριθμος του Luby Στην περίπτωση του αλγορίθμου του Luby [15] κάθε μία από τις κορυφές του γράφου ξεκινάει με ένα σύνολο από $\Delta+1$ χρώματα. Σε κάθε γύρο κάθε μη χρωματισμένος κόμβος προσπαθεί να πάρει ένα χρώμα τυχαία από αυτά τα πανομοιότυπα σύνολα. Η απόφασή του αυτή κοινοποιείται σε όλους του γείτονες. Αν κανένας από αυτούς δεν επιλέγει το ίδιο χρώμα τότε ο κόμβος ολοκληρώνει τον γύρο του και παίρνει το χρώμα για δικό του. Διαφορετικά στον επόμενο γύρο ψάχνει για ένα άλλο χρώμα.

Ο αλγόριθμος θεωρεί ότι δεν χρειάζεται να ξυπνάνε σε κάθε γύρο οι αχρωμάτιστοι κόμβοι αλλά να μόνο κατά μία πιθανότητα. Το συγκεκριμένο συμπέρασμα δεν θεωρείται πλέον σωστό και αποδείχτηκε ότι αν προτιμήσουμε πιθανότητα να ζητήσει ο κάθε αχρωμάτιστος κόμβος χρώμα είναι 1 (δηλαδή πάντα) έχουμε καλύτερη απόδοση από πλευράς της πολυπλοκότητας.

Κατανεμημένος χρωματισμός με την ελάχιστη δυνατή πληροφορία Οι Christian Scheideler, Kishore Kothapalli, Christian Schindelhauer και Melih Onus [16] έδωσαν έναν αλγόριθμο ο οποίος χρειάζεται $O(\sqrt{\log n})$ γύρους αποστέλλοντας bits με πολυπλοκότητα $\Omega(\log n)$. Ο χρωματισμός με μεγάλη πιθανότητα είναι της τάξης Δ όπου Δ είναι ο μέγιστος βαθμός του γράφου.

Στην περίπτωση αυτή η πολυπλοκότητα μετριέται με βάση τα bits δεδομένων και όχι τόσο την χρονική πολυπλοκότητα σε σχέση με την είσοδο. Επιπλέον τα καλύτερα αποτελέσματα λαμβάνονται όταν ο γράφος έχει κατευθύνσεις.

Κατανεμημένος αλγόριθμος με πρότερη γνώση μοναδικού αριθμού και λίστας χρωμάτων Ο Ojvind Johansson [17] το 2005 έδωσε και αυτός ένα καλό κατανεμημένο αλγόριθμο ο οποίος και ακολουθεί:

Όσο μία κορυφή δεν έχει χρώμα, διατηρεί μια παλέτα από χρώματα που δεν ανήκουν σε κανέναν από τους γείτονές του. Η λίστα αυτή έστω την ονομάζουμε $P(u)$ και αρχικοποιείται στα πρώτα $d(u) + 1$ χρώματα του S , όπου $d(u)$ είναι ο βαθμός του u και S μία προκαθορισμένη ακολουθία από χρώματα. Ο αλγόριθμος εκτελείται σε γύρους. Σε κάθε έναν από αυτούς, κάθε αχρωμάτιστος κόμβος, τυχαία επιλέγει ένα χρώμα από το σύνολο που έχει τοπικά. Ενημερώνει τους γείτονές του και ελέγχει αν κάποιος από αυτούς έχει επιλέξει το ίδιο χρώμα. Αν όχι το κρατάει. Αν ναι τότε δεν το κρατάει ούτε ο ίδιος ούτε και ο γείτονάς του.

Στην αρχή του επόμενου γύρου ενημερώνει ο κάθε κόμβος το σύνολο υποψηφίων χρωμάτων του ως προς τα χρώματα που πήραν οι γείτονές του στον προηγούμενο γύρο.

3.3 Ασύρματα Δίκτυα Αισθητήρων

Μέχρι σήμερα η διαχείριση της πληροφορίας γινόταν σε μεγάλα υπολογιστικά συστήματα ή σε συστήματα μεγέθους όσο περίπου ένας φορητός υπολογιστής. Το μέγεθος αυτών των συστημάτων ήταν τέτοιο ώστε αφενός να ακολουθεί την τεχνολογία-που πριν βελτιωθεί δεν ήταν εφικτές λύσεις με πολύ μικρά σε μέγεθος τρανζίστορ- αφετέρου όμως, γιατί η επεξεργασία της πληροφορίας γινόταν με βάση κάποιον χρήστη του συστήματος και στην καλύτερη περίπτωση ήταν έμμεση πληροφορία από το περιβάλλον με την έννοια της εισόδου των μετρήσεων σε αυτό από τον χρήστη.

Σήμερα όμως υπάρχει όσο ποτέ άλλοτε μία διαφορετική κατηγορία υπολογιστών που έχουν να κάνουν άμεσα με το περιβάλλον γύρω τους και τα ονομάζουμε ενσωματωμένα συστήματα. Η αλήθεια είναι ότι τέτοια συστήματα υπήρχαν από πολλά χρόνια στην παραγωγή και μάλιστα θεωρείται ότι το μεγαλύτερο ποσοστό των μηχανών που διαχειρίζονται πληροφορία είναι ενσωματωμένες και δέχονται είσοδο απευθείας από το περιβάλλον χωρίς ιδιαίτερη ανθρώπινη παρέμβαση. Όσο παράξενο και αν ακούγεται αυτό τα παραδείγματα τέτοιων συσκευών το πιστοποιούν. Πλυντήρια, φούρνοι, αυτόματα θερμομέτρα σε συστήματα κλιματισμού, όργανα μέτρησης σε παραγωγική αλυσίδα κ.α. είναι είτε καθαρά ενσωματωμένα συστήματα ή η τομή των ενσωματωμένων με τα ανθρωποκεντρικά συστήματα.

Το μέλλον προβλέπει ακόμα μεγαλύτερη εξάπλωση στην καθημερινή μας ζωή. Υπάρχει η τάση για την εφαρμογή και χρήση τέτοιων συστημάτων ακόμα και πάνω σε μικροπράγματα της καθημερινότητας μας όπως είναι τα ψώνια. Η τάση δείχνει να είναι μια μετάβαση σε μία 'Περιβαλλοντική Ευφυΐα' που θα υπάρχει τριγύρω μας. Μικρές τέτοιου είδους μηχανές θα συνεργάζονται για να ολοκληρώνουν διαδικασίες και να επικοινωνούν με τους ανθρώπους. Θα έχουμε τελικά επικοινωνία μεταξύ ανθρώπου-υπολογιστή, υπολογιστή-υπολογιστή και υπολογιστή με το περιβάλλον και συνεπώς ανθρώπου με περιβάλλον. Μια τέτοια κατάσταση μετατρέπει την επικοινωνία πιο σημαντική από την κλασική ανάλυση πληροφορίας.

Από τα παραπάνω καταλαβαίνουμε ότι για να γίνουν όλα αυτά εφικτά, ίσως το πιο σημαντικό κομμάτι που πρέπει να υλοποιηθεί είναι η επικοινωνία. Σε πολλές περιπτώσεις η συμβατική λύση με καλωδιώσεις είναι ικανή να στείλει την πληροφορία εκεί που χρειάζεται. Δεν είναι όμως λίγα τα σενάρια που τέτοιες λύσεις είναι ανέφικτες εξαιτίας από την μία πλευρά του κόστους που έχουν και από την άλλη της πρακτικότητας που δεν εξυπηρετούν (π.χ. σε ένα σπίτι δεν είναι δυνατόν να έχουνε καλώδια από όλες τις συσκευές σε όλες τις συσκευές που μπορεί να πρέπει να επικοινωνήσουν). Η λύση συνεπώς για τέτοιες περιπτώσεις είναι η ασύρματη επικοινωνία ανάμεσα στις συσκευές.

Η ανάγκη αυτή που δημιουργείται, οδήγησε σε ένα νέο είδος δικτύων τα ασύρματα δίκτυα αισθητήρων. Αποτελούνται από ένα σύνολο κόμβων οι οποίοι είναι ικανοί να επικοινωνούν με τον περιβάλλοντα χώρο τους, διαβάζοντας και ελέγχοντας της φυσικές παραμέτρους. Συνεργάζονται μεταξύ τους για την ολοκλήρωση διαδικασιών που συνήθως ένας μόνος του είναι αδύνατον να ολοκληρώσει. Φυσικά για την επικοινωνία τους χρησιμοποιούν ασύρματα μέσα. Από μόνος του ο καθένας έχει κάποια υπολογιστική δύναμη, δυνατότητα ασύρματης επικοινωνίας

και ικανότητα επικοινωνίας με το περιβάλλον. Στην πραγματικότητα υπάρχει πολύ συχνά και κάποιος συντονιστής.

Τα ασύρματα δίκτυα αισθητήρων έχουν πολλές δυνατότητες και δεν είναι σχεδιασμένα για μία συγκεκριμένη εργασία. Έχουν κινήσει το ενδιαφέρον πολλών ερευνητών που ασχολούνται με τα διαφορετικά σενάρια που εμφανίζονται. Για παράδειγμα είναι πολλές οι περιπτώσεις που δεν μπορεί να υπάρξει σύνδεση με καλώδιο και συνεπώς οι κόμβοι βασίζονται μόνο στην ενσωματωμένη μπαταρία τους. Είναι λοιπόν σημαντικό να βρεθούν λύσεις και αλγόριθμοι που καταναλώνουν την λιγότερο δυνατή ενέργεια. Ακόμα το κόστος κάθε ξεχωριστού κόμβου μπορεί να είναι απαραίτητο να ελαχιστοποιείται σε κάθε περίπτωση.

Η πραγματοποίηση συστημάτων μόνο από τέτοιους κόμβους θεωρείται απαραίτητη για την μετάβαση στην περιβαλλοντική ευφυΐα που αναφέραμε νωρίτερα. Για να συμβεί αυτό πρέπει να καθιερωθούν πρωτόκολλα λειτουργίας και να καθιερωθούν δομές κοινά αποδεκτές για τις περιπτώσεις χρήσης τους. Στην εργασία αυτή αναλύουμε την περίπτωση του χρωματισμού τους που από το κεφάλαιο 2 φαίνεται πόσο σημαντικός είναι για τις λειτουργίες τους. Μόνο να σκεφτούμε την περίπτωση του χρόνο-προγραμματισμού τους σε σχέση με την ενεργειακή κατανάλωση και τον χρόνο ζωής της μπαταρίας, θα καταλάβουμε πόσο σημαντική είναι η ύπαρξη ικανού και γρήγορου αλγορίθμου με όσο καλύτερα αποτελέσματα γίνεται. Προσθέτοντας το γεγονός ότι η αποστολή μηνύματος είναι από της πιο ενεργοβόρες διαδικασίες, πρέπει να βρούμε την χρυσή τομή ανάμεσα στην απόδοση και τον αριθμό τους. Τέλος θα δούμε και πρακτικά πόσο γρήγοροι είναι οι αλγόριθμοι αυτοί και τι ενέργεια χρειάζονται για να εκτελεστούν.

3.3.1 Πρακτικές εφαρμογές

Στο σημείο αυτό αξίζει να δούμε σενάρια χρήσης τέτοιων δικτύων για να κατανοήσουμε πλήρως ότι πράγματι οδεύουμε προς αυτήν την κατεύθυνση, καθώς και το πόσο πολύ χρήσιμα είναι για την ανάπτυξη της τεχνολογίας.

Εφαρμογές αποφυγής καταστροφών Ένα τυπικό σενάριο έχει να κάνει με την αντιμετώπιση πυρκαγιών. Αισθητήρες με θερμόμετρα που μπορούν να γνωρίζουν την γεωγραφική τους θέση (ο κάθε ένας σχετικά με τους υπόλοιπους) ρίχνονται από ένα αεροπλάνο πάνω από μια πυρκαγιά. Δημιουργούν ένα χάρτη θερμοκρασιών ή καθορίζουν την περίμετρο των πυρκαγιών και περιοχές μεγάλης θερμοκρασίας που μπορούν να πάνε οι πυροσβέστες οι οποίοι θα ενημερωθούν από υπολογιστές χειρός. Παρόμοιες εφαρμογές βρίσκουμε στην περίπτωση των στρατιωτικών παρακολούθησεων περιοχών. Τέτοιου είδους εφαρμογές απαιτούν φθηνούς κόμβους που δεν μας απασχολεί να τους χάσουμε μετά την χρήση τους. Ο χρόνος ζωής χωρίς καλωδίωση δεν είναι πολύ σημαντικός.

Έλεγχος περιβάλλοντος και καταγραφή βιοποικιλότητας Μέτρησεις σε υγρασία, μόλυνση του περιβάλλοντος, θερμοκρασίες ή ακόμα και η μελέτη περιοχών μέσα στο νερό (π.χ. στον πάτο μίας λίμνης) είναι χρήσιμες και για πρακτικούς λόγους αλλά και για ερευνητικούς. Ο αριθμός των ζώων και των φυτών σε μία περιοχή μπορεί να βοηθηθεί από δίκτυα αισθητήρων.

Σε αντίθεση με τα προηγούμενα παραδείγματα, εδώ χρειαζόμαστε ανεξάρτητους από τον ανθρώπινο παράγοντα κόμβους με μεγάλη αντοχή στον χρόνο και μικρή κατανάλωση ενέργειας. Επιπλέον χρειάζονται πολλοί κόμβοι, μεγάλης ακρίβειας και μικροί σε μέγεθος για να μην ενοχλούν το περιβάλλον που πρέπει να καταγράψουν.

Έξυπνα κτήρια Έχει παρατηρηθεί ότι τα περισσότερα κτήρια καταναλώνουν τεράστιες ποσότητες ενέργειας χωρίς αυτό να είναι απαραίτητο. Η κατανάλωση σε ρεύμα εξαιτίας κακής διαχείρισης του κλιματισμού, υγρασίας και αερισμού είναι πολύ σύννηθες φαινόμενο ακόμα και σε μικρά κτήρια όπως είναι τα σπίτια. Καλύτερη διαχείριση των παραπάνω από δίκτυα αισθητήρων μπορεί να οδηγήσει σε μείωση της σπατάλης ενέργειας αλλά και σε μεγάλη ακρίβεια στη ρύθμιση των συνθηκών και κατ' επέκταση στη βελτίωση τους για τους ανθρώπους που ζούνε μέσα σε αυτά.

Μπορούμε ακόμα να τα χρησιμοποιήσουμε και για την βελτίωση της ασφάλειας στα κτήρια. Πέραν του εντοπισμού πυρκαγιών, υπάρχουν συστήματα που ελέγχουν την πίεση του νερού ή ακόμα και την αντοχή των κτηρίων σε περιπτώσεις σεισμού. Είναι χρήσιμα ακόμα και όταν ένα κτήριο είναι ετοιμόρροπο για να ειδοποιούν τους διασώστες, που είναι ασφαλές να πάνε και που βρίσκονται εγκλωβισμένοι άνθρωποι.

Εδώ οι απαιτήσεις είναι συνήθως μικρές. Στις περισσότερες περιπτώσεις μπορούμε να τροφοδοτούμε με ενέργεια τους κόμβους (που δεν χρειάζεται να είναι πολλοί σε αριθμό). Διαφορετικά όμως πρέπει να έχουν ενέργεια για μεγάλο αριθμό χρόνων.

Ιατρική Στον τομέα αυτό υπάρχει από την μια πλευρά μεγάλο όφελος από την χρήση των ασυρμάτων δικτύων αισθητήρων και από την άλλη μεγάλη διαμάχη για το αν η χρήση τους εδώ είναι ηθικά σωστή. Οι γιατροί μπορούν να ελέγχουν από μακριά την κατάσταση των ασθενών τους, να μαθαίνουν έγκαιρα για τυχόν προβλήματα και να επεμβαίνουν πολύ πιο γρήγορα από ότι μπορούν τώρα. Μπορεί να προσφερθεί και η υποδομή για αυτόματη παροχή φαρμάκων σε ακριβή ποσότητα και όταν μόνο χρειάζεται. Η απουσία καλωδίων φυσικά οδηγεί σε μεγαλύτερη άνεση για τους ασθενείς ειδικά σε περιπτώσεις που η παρακολούθηση είναι για μεγάλο χρονικό διάστημα. Φυσικά όλα αυτά πρέπει να γίνονται με σεβασμό στην ιδιωτική ζωή κάθε ανθρώπου, πράγμα που προκαλεί και την διαμάχη σε θέματα ηθικής.

Εργοστασιακή διαχείριση και έλεγχος μηχανημάτων Από τις πιο απαιτητικές διεργασίες που μπορούν να κάνουν τα δίκτυα αισθητήρων με μεγάλο κόστος και εύρος όμως εφαρμογών, είναι η διαχείριση μεγάλων μονάδων, είτε αυτές είναι εργοστάσια παραγωγής ή στρατιωτικές μονάδες με απαγορευμένες περιοχές. Εδώ η προσφορά μπορεί να γίνει για την εύρεση εισβολέων σε χώρους που απαγορεύεται να βρίσκονται, άνοιγμα εισόδων για αυτούς που μπορούν να μπουν και φέρουν το κατάλληλο σήμα που επιτρέπει την είσοδο και έλεγχο για διαρροές, προβλήματα κλπ. Αισθητήρες τοποθετημένοι κατάλληλα σε εργοστασιακά μηχανήματα, μπορούν να καθορίσουν αριθμούς εισόδων για αυτές, να κάνουν συνεχή έλεγχο για προβλήματα και να ειδοποιούν τους τεχνικούς για επιδιορθώσεις, να κάνουν έλεγχο ποιότητας στα παραγόμενα προϊόντα και πολλά άλλα. Φυσικά η διαφορετικότητα στις εφαρμογές δημιουργεί και διαφορετικές απαιτήσεις, από απλούς φθηνούς, εύκολα αντικαταστήσιμων αισθητήρων, μέχρι πολύ ακριβών, ανθεκτικών και με μεγάλη μπαταρία αισθητήρων. Η ενέργεια βέβαια πολλές φορές μπορεί να δίνεται και με καλωδιώσεις οπότε δεν χρειάζεται καθόλου μπαταρία. Η θέσπιση standards εδώ είναι μεγάλη πρόκληση.

Καλλιέργειες ακριβείας Τόσο στις καλλιέργειες όσο και στην κτηνοτροφία, μπορούμε να βρούμε πολλές εφαρμογές των αισθητήρων. Με έναν μικρό αριθμό από αυτούς (έχει υπολογιστεί ένας ανά περιοχή 100 X 100 μέτρα) μπορούμε

να ελέγχουμε όλες τις ποσότητες που μας ενδιαφέρουν στο έδαφος, και να αυτοματοποιήσουμε από το πότισμα μέχρι και την ρίψη λιπασμάτων. Στην κτηνοτροφία ελέγχουμε την υγεία και την κατάσταση των ζώων αλλά και του περιβάλλοντος μέσα στο οποίο ζούνε αυτά αυξάνοντας την παραγωγικότητά τους και αποφεύγουμε έτσι ασθένειες που μπορούν να εξαπλωθούν πολύ γρήγορα.

Άλλες εφαρμογές Είναι πολλές ακόμα οι περιπτώσεις που μπορούν να εφαρμοστούν και επιγραμματικά θα αναφέρουμε την διαχείριση πόρων ώστε να εντοπίζονται άμεσα σε τεράστιες αποθήκες τα αντικείμενα, ή να εντοπίζεται η ακριβής θέση κατά την μετακίνηση από το ένα μέρος στο άλλο, καθώς και τον τομέα της τηλεματικής όπου σε διάφορους δρόμους μπορούν να τοποθετηθούν αισθητήρες που θα ειδοποιούν τα διερχόμενα οχήματα για προβλήματα στον δρόμο τους ή θα χρησιμοποιούνται για στατιστικές μετρήσεις και έλεγχο φαναριών κ.λ.π.

3.3.2 Προκλήσεις στον τομέα των Wireless Sensor Networks (WSN)

Το μεγάλο εύρος των εφαρμογών καθιστά δύσκολη την δημιουργία ή και την ύπαρξη ενός δικτύου γενικού σκοπού του οποίου οι κόμβοι να είναι ικανοί να τις ικανοποιήσουν όλες. Ακόμα και έτσι όμως υπάρχουν μερικά κοινά χαρακτηριστικά και μηχανισμοί σε όλες σχεδόν τις περιπτώσεις. Η κατανόηση και εύρεση των χαρακτηριστικών αυτών και η δημιουργία μηχανισμών είναι η μεγαλύτερη πρόκληση στον τομέα των WSN .

Χαρακτηριστικές απαιτήσεις Οι απαιτήσεις αυτές πρέπει να καθορίζονται σχεδόν σε όλες τις πρακτικές εφαρμογές. Επιγραμματικά είναι οι παρακάτω:

- **Τύπος υπηρεσίας:** Τα τυπικά δίκτυα έχουν να προσφέρουν μόνο μία υπηρεσία. Την μεταφορά δεδομένων από το ένα μέρος στο άλλο. Στα δίκτυα αισθητήρων η διαδικασία αυτή είναι ένα από τα μέσα που προσφέρουν την υπηρεσία. Ένα WSN αναμένεται να κάνει κάτι πρακτικό ή να δίνει πληροφορία που να είναι κατανοητή και όχι απλός αριθμός. Συνεπώς απαιτείται η θέσπιση νέων ευρύτερων παραδειγμάτων και νέος τρόπος σκέψης ως προς την λειτουργία των δικτύων.
- **Ποιότητα υπηρεσίας:** Σχετικό πολύ με την υπηρεσία είναι η ποιότητά της. Σε εφαρμογές για παράδειγμα πολυμέσων στους κλασικούς υπολογιστές, το ζήτημα είναι η καθυστέρηση και το ελάχιστο εύρος ζώνης. Στα WSN σπάνια είναι αυτές οι απαιτήσεις. Το εύρος ζώνης που ζητάμε είναι μικρό και στις περισσότερες περιπτώσεις το μόνο θέμα που έχουμε είναι να φτάσει το πακέτο στον προορισμό του. Φυσικά σε κάποιες άλλες μπορεί να χρειαζόμαστε και ταχύτητα. Το τι ακριβώς είναι απαραίτητο είναι ζήτημα που πρέπει να διευθετηθεί.
- **Αντοχή στα σφάλματα:** Είναι σχεδόν πάντοτε απαραίτητο να χρησιμοποιούμε περισσότερους κόμβους από αυτούς που χρειάζονται για μια υπηρεσία. Επειδή πολλοί κόμβοι μπορεί να ξεμένουν από ενέργεια ή να καταστρέφονται από περιβαλλοντικούς λόγους ενώ δεν πρέπει ποτέ να χάνεται η επικοινωνία μεταξύ τους, ανάλογα με την περίπτωση, πρέπει να καθορίζεται ο ασφαλής αριθμός τους.

- **Χρόνος ζωής:** Η αποτελεσματική διαχείριση και ο καθορισμός της ενέργειας των κόμβων είναι κύρια πρόκληση στον σχεδιαστή του δικτύου. Τις περισσότερες φορές θα χρειαστεί να βασιστούμε στην μπαταρία των κόμβων. Η παροχή ενέργειας θα πρέπει να είναι αρκετή ώστε να τελειώσει η εργασία ή όσο πιο πολύ γίνεται. Υπάρχουν βέβαια και εναλλακτικοί τρόποι, όπως είναι ηλιακή ή η κινητική ενέργεια αν για παράδειγμα οι κόμβοι είναι μικροί και ελαφροί ώστε να κινούνται. Σε τέτοιες περιπτώσεις το ιδανικό είναι ο χρόνος ζωής να μην τελειώνει όμως οι κόμβοι να λειτουργούν περιστασιακά και μετά να επαναφορτίζονται (π.χ. μόνο τις ημέρες που έχει ήλιο). Ο ακριβής καθορισμός του χρόνου ζωής είναι άμεσα συνδεδεμένος με την εφαρμογή και ως μετρική για την ώρα έχουμε τον χρόνο που αποτυγχάνει ο πρώτος κόμβος ή ένα ποσοστό τους, ανάλογα την περίπτωση.
- **Επεκτασιμότητα:** Ένα δίκτυο αισθητήρων μπορεί εν δυνάμει να αποτελείται από πολύ μεγάλο αριθμό κόμβων. Μεγάλη πρόκληση αποτελεί αυτό γιατί τα πρωτόκολλα που εφαρμόζονται και οι αλγόριθμοι θα πρέπει να τους υποστηρίξουν με ικανοποιητικό τρόπο.
- **Μεγάλο εύρος πυκνότητας:** Ανάλογα με την υπηρεσία που προσφέρεται απαιτούνται διαφορετικοί αριθμοί ανά επιφάνεια ή μονάδα όγκου. Ακόμα και στην ίδια εφαρμογή για το ίδιο στιγμιότυπο μπορεί η πυκνότητα να διαφέρει. Το δίκτυο κρίνεται απαραίτητο να μπορεί να αντεπεξέλθει σε τέτοιες περιπτώσεις.
- **Ικανότητα προγραμματισμού:** Ένας προκαθορισμένος τρόπος λειτουργίας κρίνεται από τους ερευνητές ανίκανος να ανταποκριθεί στις απαιτήσεις σε πραγματικές εφαρμογές. Συνεπώς αποτελεί πρόκληση η ύπαρξη δυνατότητας αλλαγής της λειτουργίας και πριν την εκκίνηση του δικτύου αλλά και κατά την διάρκεια της εφαρμογής, ή ακόμα και της αλλαγής τελείως του κώδικα που εκτελείται.
- **Ικανότητα αυτό-διατήρησης:** Το δίκτυο στα πλαίσια της αντοχής στα σφάλματα χρειάζεται την δυνατότητα να αναγνωρίζει την προσωπική του υγεία και να πραγματοποιεί κινήσεις που σε περίπτωση αστοχίας να επαναφέρεται όσο το δυνατόν πιο αποδοτικά στην κανονική κατάσταση λειτουργίας.

Απαιτούμενοι μηχανισμοί Οι παραπάνω απαιτήσεις χρειάζεται σε κάθε πρακτική εφαρμογή να καθοριστούν, και για να καλυφθούν, νέοι καινοτόμοι μηχανισμοί πρέπει να βρεθούν μαζί με νέες αρχιτεκτονικές και πρωτόκολλα. Αυτό από μόνο του είναι μια μεγάλη πρόκληση. Αφενός οι μηχανισμοί αυτοί είναι απαραίτητο να καλύπτουν τις απαιτήσεις για τις οποίες χρησιμοποιούνται μαζί με όλες τις προκλήσεις ανά εφαρμογή για να λειτουργεί αυτή όπως σχεδιάστηκε, αφετέρου δεν πρέπει να είναι ικανοί να λειτουργούν μόνο σε πολύ μικρό αριθμό εφαρμογών με αποτέλεσμα να χρειάζεται υλοποίηση εξαρχής κάθε φορά που σχεδιάζουμε μία υπηρεσία, γιατί αυτό θα καθιστούσε τα δίκτυα αισθητήρων πολύ ακριβά και θα αποφεύγαμε την χρήση τους. Τυπικοί μηχανισμοί σχεδόν σε όλα τα δίκτυα είναι οι παρακάτω:

- **Δρομολόγηση:** Η επικοινωνία των κόμβων είναι κύριο χαρακτηριστικό των δικτύων αυτών. Υπάρχει όμως το πρόβλημα ότι η ακτίνα στην οποία είναι δυνατή περιορίζεται από το ίδιο το υλικό κατασκευαστικά από θέματα

30ΚΕΦΑΛΑΙΟ 3. ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ-ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

έντασης ακτινοβολίας και ενέργειας. Απαιτείται για κάθε σχεδόν περίπτωση, για αποστολή μηνυμάτων σε μεγάλη απόσταση ή ύπαρξη αλγορίθμων που να είναι ικανά με λίγα μηνύματα να δρομολογούν από ενδιαμέσους κόμβους τα δεδομένα.

- **Εξοικονόμηση ενέργειας:** Λειτουργίες που να εξοικονομούν ενέργεια πρέπει να σχεδιάζονται και η λειτουργία του κόμβου χρειάζεται να καταναλώνει όσο το δυνατόν λιγότερη. Ικανή αποστολή δεδομένων με λίγη ενέργεια είναι σχεδιαστικό ζητούμενο ενώ υπάρχουν και θέματα μη ομογενούς κατανάλωσης ενέργειας από συγκεκριμένους κόμβους.
- **Αυτοματοποιημένες ρυθμίσεις:** Τις περισσότερες φορές ο κάθε κόμβος θα χρειαστεί να ρυθμίσει τις παραμέτρους του από μόνος του χωρίς εξωτερική παρέμβαση. Στο παράδειγμα με την πυρκαγιά θα ήταν δύσκολο κάποιος να ρυθμίσει που ακριβώς θα πέσουν οι αισθητήρες από το αεροπλάνο ή να τους στείλει πληροφορία από μακριά. Συνεπώς για τέτοιες περιπτώσεις δημιουργούνται μηχανισμοί που καθιστούν τους κόμβους ικανούς να βρίσκουν μόνοι τους την σχετική τους θέση σε σχέση με άλλους κόμβους του δικτύου, να μπορούν να αντεπεξέλθουν με αυτόματο τρόπο στην απώλεια γειτόνων ή στην προσθήκη νέων σε οποιαδήποτε στιγμή κ.α.
- **Συνεργασία στους υπολογισμούς:** Ανοιχτό ερώτημα παραμένει ο τρόπος με τον οποίο μπορούν αν ομαδοποιούνται δεδομένα μέσα σε ένα δίκτυο ώστε να φτάνει η απαραίτητη πληροφορία στον προορισμό της αλλά και να ελαχιστοποιούνται τα δεδομένα που στέλνονται. Αυτό γιατί υπάρχουν σενάρια στα οποία ένας κόμβος μόνος του δεν μπορεί να πάρει όλη την πληροφορία αλλά να την μαζέψει από πολλούς γείτονές του. Ένα παράδειγμα τέτοιο είναι όταν θέλουμε να μάθουμε την μεγαλύτερη θερμοκρασία σε μία περιοχή μεγάλη όπου ένας κόμβος δεν είναι δυνατόν να κάνει μετρήσεις για όλες τις τοποθεσίες ταυτόχρονα.
- **Διατήρηση δεδομένων:** Το ζήτημα στα δίκτυα χρησιμοποιούμε σήμερα, είναι να φτάνουν τα δεδομένα από μία διεύθυνση σε κάποια άλλη. Στα WSN είναι τελείως διαφορετικά τα πράγματα. Τα δεδομένα που περνάνε από τον έναν κόμβο σε κάποιον άλλο είναι σημαντικό να φτάσουν, και κάποιες φορές παραπάνω από μια φορά. Εδώ ζητάμε να μην χαθούν και το από που ήρθαν (πέραν ίσως των συντεταγμένων τους) είναι πληροφορία που δεν χρειάζεται. Είναι πολύ συγγενές ζήτημα με τα δύο παραπάνω αλλά και την ανάκαμψη από προβλήματα που μπορεί να εμφανιστούν από την ξαφνική βλάβη σε κάποιον κόμβο.
- **Τοπικότητα:** Κόμβοι με περιορισμένους πόρους, είναι καλό να επικοινωνούν μόνο με τους γείτονες κόμβους τους αλλά και να περιορίζουν τους υπολογισμούς όσο το δυνατόν περισσότερο. Ελπίζουμε έτσι ότι θα βοηθηθεί η επεκτασιμότητα σε πολύ μεγάλους αριθμούς κόμβων. Το πως θα σχεδιαστούν πρωτόκολλα για να γίνει κάτι τέτοιο εφικτό είναι πολύ μεγάλη πρόκληση στον τομέα των WSN .
- **Διαχείριση κέρδους:** Σχεδιαστικά είναι πολύ δύσκολο να καθορίσουμε κέρδη και απώλειες. Πολλές φορές κέρδος σε κάποιον τομέα είναι απώλεια για κάποιον άλλο. Για παράδειγμα η μεγάλη ακρίβεια στις μετρήσεις είναι σίγουρο πως θα οδηγήσει σε μεγάλη κατανάλωση ή όπως για παράδειγμα

στην περίπτωση του χρωματισμού γράφων, μεγάλη απόδοση σε σχέση με τον χρωματικό αριθμό, απαιτεί περισσότερα μηνύματα ή/και χρόνο επεξεργασίας. Το θέμα εδώ είναι ο σχεδιασμός πρωτοκόλλων που να μπορούν να εφαρμοστούν σε μεγάλο εύρος εφαρμογών με διαφορετικές απαιτήσεις σε κέρδη στους διαφόρους τομείς.

3.3.3 Διαφορές με άλλα ad-hoc δίκτυα

Η χρησιμότητα και τα παραδείγματα στα οποία εφαρμόζονται τα WSN δίκτυα θυμίζει πολύ παραδείγματα και πρακτικές που βρίσκουμε σε ένα άλλο είδος δικτύων τα Mobile Ad hoc (MANET) . Τα δίκτυα αυτά έχουν το χαρακτηριστικό να στήνονται για έναν σκοπό σε μικρό χρονικό διάστημα με στόχο να ικανοποιήσουν την ανάγκη της επικοινωνίας. Παράδειγμα τέτοιο είναι ένα τοπικό δίκτυο από φορητούς υπολογιστές σε μία συνάντηση. Πιο συχνά όμως τα συναντάμε σε περιπτώσεις που απαιτείται ασύρματη επικοινωνία και φυσικά όταν απαιτείται να είναι μετακινήσιμοι οι κόμβοι. Οι πυροσβέστες σε μία φωτιά που πρέπει να επικοινωνήσουν είναι τέτοιο δίκτυο, όπως και κτυριακές δομές, που απαγορεύουν την ύπαρξη σταθερών κόμβων και καλωδίων. Φυσικά και εδώ η αυτόματη αρχικοποίηση και ρύθμιση των παραμέτρων χωρίς την ανθρώπινη παρέμβαση είναι κύριο ζητούμενο όπως επίσης και η μεταφορά δεδομένων με χρήση ενδιάμεσων σταθμών.

Παρά της πολλής ομοιότητας υπάρχουν και κάποιες βασικές διαφορές ως προς την έρευνα που απαιτείται για το καθένα ξεχωριστά.

- **Εφαρμογές και εξοπλισμός:** Στα MANETs πολύ συχνά έχουμε σε κάποιο βαθμό και ανθρώπινη παρέμβαση. Είναι σύνηθες τα δίκτυα αυτά να έχουν και μεγάλα συστήματα (laptops ,desktop PCs) μεταξύ των κόμβων που τα απαρτίζουν. Οι εφαρμογές που μπορεί να είναι για παράδειγμα μεταφορά ήχου από μέρος σε μέρος έχουν τέτοιες ανάγκες.
- **Συγκεκριμένες εφαρμογές:** Αν και στα MANETs υπάρχει διαφοροποίηση στις εφαρμογές , το εύρος που καλύπτεται από τα ασύρματα δίκτυα αισθητήρων είναι πολύ μεγαλύτερο παρά το ότι δεν γίνεται να δοθεί ενιαία λύση για όλα αυτά μαζί.
- **Επικοινωνία με το περιβάλλον:** Όπως έχουμε δει μέχρι τώρα, στα WSN έχουμε σχεδόν πάντα επικοινωνία με το περιβάλλον. Αυτό από μόνο του τα διαφοροποιεί με τα MANETs που επικοινωνούν με κάποιον τρόπο με τον άνθρωπο. Είναι λοιπόν λογικό να έχουμε στα MANET τυπική επικοινωνία μεταξύ των κόμβων που δεν βρίσκουμε στα WSN , όπου η ροή δεδομένων είναι συχνά κατά μεγάλα χρονικά διαστήματα ιδιαίτερα μειωμένη, με μικρές περιόδους μεγάλης κίνησης όταν συμβεί κάτι στον περιβάλλοντα χώρο.
- **Αναλογία:** Τα WSN είναι πιο πιθανό να αποτελούνται από έναν μεγάλο αριθμό κόμβων. Η ύπαρξη αναγνωριστικού σε αυτά είναι μια διαδικασία που κοστίζει είτε στην κατασκευή τους ή στην απόδοσή τους και την κατανάλωση ενέργειας και πόρων. Υπάρχουν και περιπτώσεις που δεν είναι η απαραίτητη η ύπαρξή τους οπότε βολεύει η έρευνα και δημιουργία πρωτοκόλλων και εφαρμογών χωρίς αυτόν τον αριθμό. Στα MANETs θεωρούμε εύκολη την ύπαρξή του και μη αναγκαίοι τέτοιοι αλγόριθμοι και πρακτικές.

- **Ενέργεια:** Απαραίτητη και στα δύο είδη δικτύων, με ιδιαίτερη έμφαση στα ασύρματα δίκτυα αισθητήρων όπου μπορεί να μην υπάρχει κανένας τρόπος να τα επανατροφοδοτήσουμε.
- **Ποιότητα υπηρεσιών:** Στα MANET κάθε κόμβος από μόνος του πρέπει να προσφέρει ποιότητα υπηρεσίας και η αστοχία του για οποιονδήποτε λόγο είναι πολύ πιο σημαντική από ότι σε ένα δίκτυο WSN όπου ο κάθε κόμβος από μόνος του δεν είναι μεγάλης σημασίας και πρέπει να υπάρχουν πάνω από μία λύσεις για την αποφυγή προβλημάτων αν αυτός καταρρεύσει.
- **Έμφαση στα δεδομένα:** Είπαμε και παραπάνω ότι στα WSN ,σημασία έχει να φτάσουν τα δεδομένα εκεί που πρέπει. Στα υπόλοιπα δίκτυα το ζήτημα είναι να φτάσουν τα δεδομένα από την μία διεύθυνση στην άλλη.
- **Απλότητα και έλλειψη πόρων:** Μιας και τα WSNs είναι απλά και οι πόροι που έχουν είναι λίγοι, έτσι πρέπει να είναι και οι αλγόριθμοι που εκτελούνται τοπικά. Αν πάρουμε για παράδειγμα την δρομολόγηση σε τέτοια δίκτυα πρέπει να είναι πολύ απλούστερη από τους ενεργοβόρους αντίστοιχους των δικτύων MANET . Επιπλέον πολλές φορές λόγω της απλότητας αυτής, χρειάζεται να αφαιρούμε επίπεδα από το κλασικό OSI μοντέλο δικτύων που εφαρμόζεται ως επί το πλείστον στις περισσότερες περιπτώσεις υπολογιστικών δικτύων.
- **Κινητικότητα:** Η κίνηση των κόμβων στα MANET επιβάλλει την διαχείριση της δρομολόγησης όταν αλλάζει η τοπολογία του δικτύου. Στα WSN υπάρχουν και δύο άλλοι παράγοντες που πρέπει να εξετάσουμε όταν οι κόμβοι είναι κινούμενοι. Πρώτο είναι η σιγουριά που πρέπει να έχουμε ότι μπορούμε ακόμα να παρατηρήσουμε το φαινόμενο που θέλουμε αποδοτικά μιας και αυτός είναι ο στόχος μας. Το δεύτερο είναι ότι η ύπαρξη κόμβων που πρέπει να φτάσει η πληροφορία (αν είναι και αυτοί κινητοί) δημιουργούν άλλες απαιτήσεις και ανάγκες για πρωτόκολλα που δεν υπάρχουν στα MANETs .

3.3.4 Μηχανισμοί και απαιτήσεις στους αλγορίθμους που αναπτύσσουμε

Στα πλαίσια της παρούσας εργασίας υλοποιήθηκαν αλγόριθμοι χρωματισμού. Φυσικά η υλοποίηση έγινε στα πλαίσια των κατανεμημένων συστημάτων και πιο συγκεκριμένα για τα ασύρματα δίκτυα αισθητήρων. Για αυτό τον λόγο έγινε προσπάθεια να αντιμετωπιστούν και οι προκλήσεις και να θεσπιστούν οι μηχανισμοί που αναφέραμε παραπάνω. Κάποιες από αυτές δεν υπάγονται στο σχεδιασμό μας μιας και ο αλγόριθμος χρωματισμού για παράδειγμα

Πιο συγκεκριμένα ως προς τον τύπο υπηρεσίας αυτό που προσφέρουμε είναι έναν ικανοποιητικά καλά χρωματισμένο γράφο από κορυφές σε κάθε περίπτωση. Η ποιότητα της υπηρεσίας ήταν αρκετά πιο δύσκολο ζητούμενο μιας και πάντα χρειαζόμαστε γράφους που ανεξάρτητα με τον χρωματικό αριθμό, να έχει όλους τους κόμβους του κανονικά χρωματισμένους. Αυτό απαιτεί πάντα τα μηνύματα να φτάνουν στον προορισμό τους, πράγμα που εξαιρεί κάποιους αλγορίθμους δρομολόγησης. Η δρομολόγηση ήταν ένα θέμα που μας απασχόλησε ιδιαίτερα και παρακάτω θα δούμε τι συμβαίνει γενικά στα WSN ως προς αυτήν και ποια λύση χρησιμοποιήσαμε εδώ.

Επιπλέον σε επόμενο κεφάλαιο στις περιγραφές των αλγορίθμων θα δούμε ότι αυτοματοποιήσαμε σε μεγάλο βαθμό τις ρυθμίσεις, η συνεργασία μεταξύ των κόμβων επιτεύχθηκε από την κατανομημένη λειτουργία των κόμβων, ενώ έγινε προσπάθεια και στην μεγαλύτερη δυνατή μείωση των μηνυμάτων ακόμα και εις βάρος τους χρωματικού αριθμού για να εξοικονομήσουμε ενέργεια, μιας και η αποστολή μηνυμάτων είναι από τις πιο ενεργοβόρες διαδικασίες. Το κομμάτι αυτό περιλαμβάνει και την διαχείριση κέρδους. Η κινητικότητα δεν είναι κάτι που μας απασχολήσει μιας και θεωρούμε τους κόμβους μας σταθερούς αν και όχι όλους με τις ίδιες δυνατότητες όπως θα δούμε και παρακάτω. Τέλος η αντοχή στα σφάλματα και η επιβάρυνση σε μηνύματα μελετήθηκε με την εισαγωγή σφαλμάτων στο περιβάλλον εξομοίωσης.

Δρομολόγηση στα δίκτυα αισθητήρων

Είδαμε νωρίτερα τον αλγόριθμο πάνω στον οποίο βασιστήκαμε για τις υλοποιήσεις μας. Μία από τις προϋποθέσεις του ήταν η ολική γνώση των αριθμών των χρωμάτων που μοιράζονται οι κόμβοι. Ολική η μερική γνώση απαιτείται και από τους δικούς μας αλγορίθμους που συνεπάγεται έναν τρόπο επικοινωνίας από τον ένα κόμβο στον άλλο, χωρίς αυτοί να είναι απαραίτητα γείτονες στον γράφο μας. Για τον λόγο αυτό έπρεπε να μελετηθούν οι μηχανισμοί τέτοιου είδους επικοινωνίας, η οποία στις περισσότερες περιπτώσεις είναι απαραίτητη.

Στο κλασικό μοντέλο δικτύων κατασκευάζονται πίνακες δρομολόγησης που η δουλειά τους είναι να δίνουν το κατάλληλο γείτονα στον οποίο θα προωθηθεί το μήνυμα που θέλουμε να στείλουμε για να φτάσει στον μη γειτονικό κόμβο. Η κατασκευή και η διατήρηση του πίνακα αυτού είναι δουλειά του πρωτοκόλλου δρομολόγησης. Στο σημείο αυτό θα δούμε περιπτώσεις όπου κόμβος προορισμού ή ένα σύνολο από κόμβους ταυτοποιείται από έναν μοναδικό αριθμό, αλλά και τις περιπτώσεις στις οποίες θέλουμε το μήνυμά μας να φτάσει σε ολόκληρο τον γράφο.

Flooding και Controlled Flooding Ο πιο απλός τρόπος δρομολόγησης πακέτων είναι να πλημμυρίζουμε το δίκτυο μας (flooding). Στην μέθοδο αυτή κάθε κόμβος στέλνει κάθε μήνυμα που λαμβάνει σε όλους τους γειτονικούς του κόμβους. Αν ο κόμβος προορισμού βρίσκεται στο δίκτυο μας το πακέτο είναι σίγουρο ότι θα φτάσει. Ένα πρόβλημα εδώ πέραν του μεγάλου αριθμού μηνυμάτων που δεν χρειάζονται είναι και η παροχή ενός μηχανισμού που να αναγνωρίζει ποια μηνύματα έχει στείλει ήδη ο κόμβος μας. Συνεπώς χρειαζόμαστε μήμη που μπορεί να είναι αρκετά μεγάλη για πολύ μεγάλα δίκτυα και έναν τρόπο να γνωρίζουμε πότε θα είναι ασφαλές να διαγράψουμε τον μοναδικό αριθμό αναγνώρισης του μηνύματός μας.

Το πρωτόκολλο ελεγχόμενης πλημμύρας (controlled flooding) λειτουργεί κάπως διαφορετικά. Ο κόμβος αποστολής στέλνει το μήνυμά του και τυχαία ένας από τους γείτονές του το προωθεί σε έναν άλλο κ.ο.κ. Αυτό έχει ως αποτέλεσμα το πακέτο μας να προχωράει τυχαία διαδρομή μέσα στο δίκτυο με την ελπίδα ότι κάποια στιγμή θα φτάσει στον προορισμό του. Είναι εμφανές ότι η καθυστέρηση εδώ μπορεί να είναι πάρα πολύ μεγάλη, και όσο μεγαλύτερο είναι το δίκτυο τόσο χειρότερο το αποτέλεσμα. Ουσιαστικά για να έχουμε το παραπάνω πρωτόκολλο στέλνουμε στην αρχή παραπάνω από ένα μηνύματα σε διαφορετικά μονοπάτια.

Για τους λόγους αυτούς οι παραπάνω αλγόριθμοι δεν προτιμούνται (αν και για την αρχικοποίησή μας θα χρησιμοποιήσουμε μια εκδοχή τους), και χρησιμοποιούνται πρωτόκολλα που παράγουν πίνακες δρομολόγησης και συγκρίνονται με

διάφορες μετρικές ανάλογα με τα κόστη σε βήματα που πρέπει να διανυθούν για να φτάσει το έναν μήνυμα από τον αποστολέα στον προορισμό, αλλά και με κόστη αρχικοποίησης, μνήμης, υπολογιστικής ισχύς κ.α.

Unicast Routing Στα δίκτυα με καλωδίωση χρησιμοποιούμε τα πρωτόκολλα των Dijkstra [18] και Bellman-Ford [19]. Σε ασύρματες και πιθανόν κινητές δικτυώσεις απαιτούνται για όλους τους λόγους που έχουμε αναφέρει παραπάνω, άλλες προσεγγίσεις. Εδώ τα πρωτόκολλα πρέπει να είναι κατανεμημένα, με μικρό κόστος αρχικοποίησης, να ρυθμίζονται αυτόματα και να μπορούν να ανταπεξέρχονται σε συχνά μεταβαλλόμενες θέσεις και κόμβους. Το ερώτημα του γρήγορου Ad hoc routing χρίζει μεγάλης σημασίας και προσοχής και έχει ερευνηθεί εκτενώς με πολλά αποτελέσματα. Δύο κατηγορίες έχουν προκύψει τα table driven, δηλαδή με προσήλωση στους πίνακες, και τα (on demand). Τα πρώτα βασίζονται στην πρόληψη και φροντίζουν να έχουν πάντοτε ενημερωμένους πίνακες ενώ τα δεύτερα κατασκευάζουν την διαδρομή όταν αυτή ζητηθεί. Φυσικά υπάρχουν και υβριδικές λύσεις. Παραδείγματα από την πρώτη κατηγορία είναι:

- Destination-Sequenced Distance Vector (DSDV)
- Clusterhead Gateway Switched Routing (CGSR)
- Wireless Routing Protocol (WRP)

ενώ στην δεύτερη έχουμε τους παρακάτω:

- Dynamic Source Routing (DSR)
- AODV
- Temporally Ordered Routing Algorithm (TORA)
- Associativity-Based Routing (ABR)
- Signal Stability Routing (SSR)

Απαιτήσεις ενέργειας Unicast δρομολόγησης Έχουμε ήδη αναφέρει το ζήτημα της ενέργειας στα ασύρματα δίκτυα αισθητήρων. Όπως και στους υπόλοιπους αλγορίθμους έτσι και εδώ πρέπει να λαμβάνουμε υπόψιν μας την ενέργεια μόνο που δεν είναι τόσο απλό. Για παράδειγμα υπάρχουν δίκτυα στα οποία μια προώθηση είναι μεγαλύτερου ενεργειακού κόστους από κάποια άλλη. Στα δίκτυα αυτά ο στόχος δεν είναι να ελαχιστοποιήσουμε τον αριθμό των βημάτων, αλλά να βρούμε το μονοπάτι με την μικρότερη απαίτηση σε ενέργεια. Άλλος στόχος μπορεί να είναι η μεγιστοποίηση της ζωής του δικτύου. Εδώ πρέπει να ξέρουμε τι σημαίνει αυτό γιατί ανάλογα με την περίπτωση ως ζωή μπορεί να αναφερόμαστε στον χρόνο αποτυχίας του πρώτου κόμβου, του τελευταίου ή όταν δεν θα καλύπτεται η περιοχή που μελετάμε.

Τέλος υπάρχουν και μελέτες σε σχέση με την υπολειπόμενη ενέργεια στους κόμβους. Για παράδειγμα ένα πρωτόκολλο επιλέγει την διαδρομή που έχει την μεγαλύτερη υπολειπόμενη μπαταρία, άλλος κοιτάει την δυσκολία με την οποία μεταφέρονται τα δεδομένα και θεωρεί ότι όσο πιο δύσκολο τόσο λιγότερη η ενέργεια που απομένει. Υπάρχουν κι άλλες ιδέες όπως το να χρησιμοποιούμε δρόμους όπου όλοι οι κόμβοι έχουν από ένα ποσοστό ενέργειας και παραπάνω κ.α.

Κεφάλαιο 4

Βασικά εργαλεία υλοποίησης

4.1 Εισαγωγή

Πριν την παρουσίαση των αλγορίθμων που σχεδιάστηκαν και υλοποιήθηκαν στα πλαίσια της εργασίας μας, θα παρουσιάσουμε τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίησή τους. Θα δούμε στο κεφάλαιο αυτό αρχικά το περιβάλλον εξομοίωσης shawn , που χρησιμοποιήθηκε αφενός για να πάρουμε τα αποτελέσματα αξιολόγησης και αφετέρου για να αντικαταστήσει το λειτουργικό σύστημα και τους πόρους που απαιτεί ο κώδικάς μας για να λειτουργήσει.

Επιπρόσθετα, παραθέτουμε και μία ανάλυση των δυνατοτήτων της βιβλιοθήκης-περιβάλλοντος δημιουργίας εφαρμογών, Wiselib , καθώς και παρουσίαση των πιο σημαντικών κομματιών που την απαρτίζουν. Τόσο για το shawn όσο και για την βιβλιοθήκη βλέπουμε και τους λόγους για τους οποίους τα επιλέξαμε.

Τέλος παρουσιάζουμε κάποιες από τις σχεδιαστικές επιλογές που έγιναν ως προς την υλοποίηση, όπως είναι καινούργιες δομές δεδομένων κατάλληλες για τους αλγορίθμους μας, το πρωτόκολλο δρομολόγησης, η γλώσσα προγραμματισμού κ.α.

4.2 Περιβάλλοντα ανάπτυξης

Εξομοιώσεις στην επιστήμη των υπολογιστών

4.2.1 Ο Εξομοιωτής shawn .

Στον τομέα των ασυρμάτων δικτύων αισθητήρων, απαιτείται ο ενδελεχής έλεγχος των αλγορίθμων που θα χρησιμοποιηθούν για τις διάφορες εργασίες, πριν την εφαρμογή τους σε πραγματικό περιβάλλον. Η εγκατάσταση των αλγορίθμων στους αισθητήρες δεν είναι ιδιαίτερα εύκολη ώστε να είναι δυνατόν να κάνουμε τις εγκαταστάσεις/απ' εγκαταστάσεις των προγραμμάτων όσες φορές χρειάζεται κατά την διάρκεια των δοκιμών. Ακόμα, δεν παρέχονται περιβάλλοντα για τον εντοπισμό και την επίλυση των σφαλμάτων σε πραγματικές δοκιμές.

Επιπλέον σπάνια έχουμε την δυνατότητα να κάνουμε δοκιμές για μεγάλο αριθμό από αισθητήρες στην πραγματικότητα, ενώ για να μπορέσουμε να δούμε την αποτελεσματικότητα ενός αλγορίθμου θα πρέπει να δούμε τι συμβαίνει ακόμα και

σε ακραίες περιπτώσεις που μπορεί να μην χρειαστεί ποτέ να δημιουργηθούν στην πράξη. Παρακάτω, θα παρουσιάσουμε τον εξομοιωτή shawn που ήταν η επιλογή μας για την ανάπτυξη και τις δοκιμές των αλγορίθμων που δημιουργήθηκαν κατά την διαδικασία υλοποίησης αλλά και για την αρχική αξιολόγηση. Μετά θα δούμε επιγραμματικά και την άλλη επιλογή για περιβάλλον εξομοίωσης (NS2) που είχαμε και θα αναφερθούν οι λόγοι για τους οποίους δεν τον προτιμήσαμε.

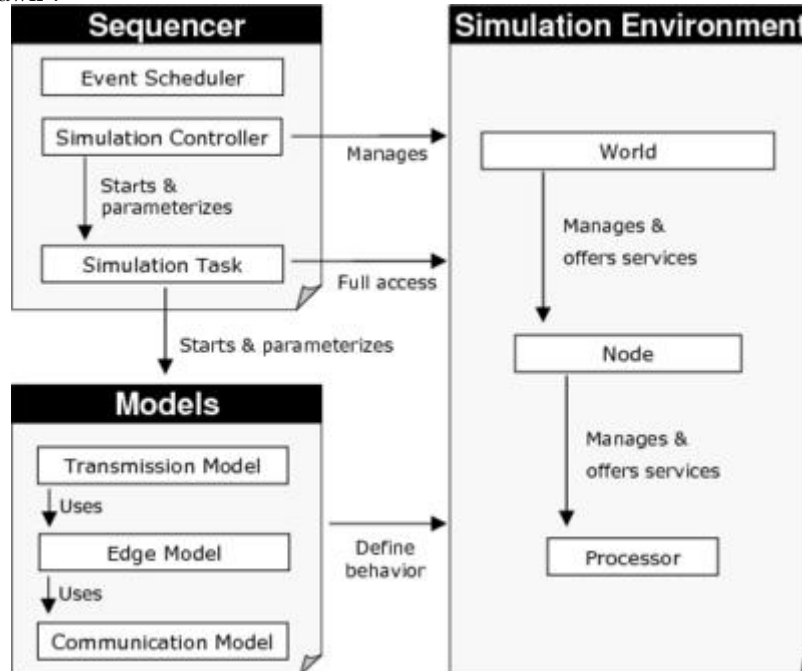
Τα κύρια χαρακτηριστικά του εξομοιωτή: Η κεντρική ιδέα του Shawn που είναι και το κυριότερο χαρακτηριστικό του, είναι ότι δεν προσπαθεί να εξομοιώσει την αιτία των φαινομένων που προκαλούν ένα αποτέλεσμα αλλά το ίδιο το αποτέλεσμα. Αυτό θα το καταλάβουμε καλύτερα με ένα απλό παράδειγμα. Όλοι γνωρίζουμε ότι σε ένα δίκτυο μπορεί να έχουμε αστοχίες υλικού (τα προβλήματα hardware ειδικά σε νέες τεχνολογίες είναι πολύ συχνά), προβλήματα σε διάφορα layers της αρχιτεκτονικής κ.α. Αυτό που συμβαίνει είναι στην πραγματικότητα να χάνονται εξαιτίας των παραπάνω προβλημάτων πακέτα, ή ακόμα και να καθυστερούν να φτάσουν. Οι περισσότεροι εξομοιωτές προσπαθούν να εξομοιώσουν τα προβλήματα αυτά. Το shawn σε αντίθεση χρησιμοποιεί αφηρημένα και εύκολα εναλλασσόμενα μοντέλα για να προσδώσουν στο περιβάλλον την απώλεια πακέτων και τις καθυστερήσεις.

Αυτό κάνει το shawn πιο «ελαφρύ» υπολογιστικά, και δίνει την δυνατότητα σε όσους αναπτύσσουν αλγορίθμους να μην ασχοληθούν με τα προβλήματα σε άλλα επίπεδα πέραν από αυτό που ασχολούνται και να επικεντρωθούν στον αλγόριθμο που ερευνούν και μόνο. Επιπλέον μπορούν να δοκιμάζουν πολύ μεγαλύτερα δίκτυα στα πειράματά τους αφού δεν είναι υποχρεωμένοι να μελετούν κάθε στιγμή πιθανές αστοχίες υλικού σε κάθε κόμβο του συστήματος. Στα μεγάλα δίκτυα τέτοιες διεργασίες θα καθυστερούσαν πάρα πολύ την εξομοίωση. Από την άλλη δεν μπορεί να προσδώσει τόσο μεγάλη προσοχή στην λεπτομέρεια όπως κάνουν άλλοι εξομοιωτές (π.χ. ο NS-2).

Επίσης σημαντικό χαρακτηριστικό του Shawn είναι η δυνατότητα που δίνει στον προγραμματιστή, να ξεκινήσει την εργασία του με οποιοδήποτε μοντέλο θέλει εκείνος. Το shawn μας δίνει την δυνατότητα να εργαστούμε σε οποιαδήποτε φάση της ανάπτυξης αλγορίθμου θέλουμε. Για παράδειγμα όταν έχουμε την ιδέα για τον αλγόριθμο που θέλουμε να φτιάξουμε δεν είναι απαραίτητο να προβούμε αμέσως στην ανάπτυξη της κατανεμημένης έκδοσής του. Έχουμε την δυνατότητα από αυτόν τον εξομοιωτή να «τρέξουμε» αλγορίθμους που να έχουν πλήρη γνώση του δικτύου που μελετάμε. Συνεπώς μπορούμε τμηματικά να φτιάξουμε τον κεντρικοποιημένο αλγόριθμο και σταδιακά να πάμε στον κατανεμημένο. Αυτό δεν φαίνεται πολύ σημαντικό με την πρώτη ματιά, αλλά είναι κάτι που βολεύει ιδιαίτερα του προγραμματιστές.

Η αρχιτεκτονική του εξομοιωτή:

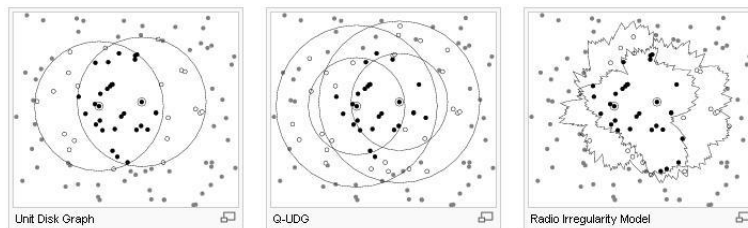
Στην παρακάτω φωτογραφία βλέπουμε τα τρία κύρια μέρη της αρχιτεκτονικής του shawn .



Το πρώτο μέρος είναι ο Ταξινομητής Διαδικασιών. Είναι η κεντρική συντονιστική μηχανή του εξομοιωτή, αφού τοποθετεί τα γεγονότα με την σειρά, και ελέγχει την εξομοίωση. Το δεύτερο κομμάτι είναι τα μοντέλα τα οποία χρησιμοποιούνται για να διαφοροποιούν το περιβάλλον που δημιουργείται κάθε φορά ανάλογα με τις εκάστοτε απαιτήσεις. Τέλος έχουμε και το περιβάλλον εξομοίωσης όπου υπάρχουν οι εικονικοί κόμβοι του δικτύου που προσπαθούμε να εξομοιώσουμε. Ας δούμε όμως αναλυτικά κάθε ένα από αυτά, μιας και θα τα χρησιμοποιήσουμε παρακάτω για την περιγραφή των δικών μας εξομοιώσεων και δοκιμών.

Τα μοντέλα του shawn :

Μοντέλο επικοινωνίας Καθορίζει τον τρόπο με τον οποίο μπορούν να επικοινωνήσουν δύο εικονικοί αισθητήρες μεταξύ τους. Για να καταλάβουμε ακριβώς θα δούμε τα παραδείγματα μοντέλων που είναι υλοποιημένα. Εδώ όπως και στα υπόλοιπα μοντέλα πρέπει να σημειώσουμε ότι ο προγραμματιστής έχει την δυνατότητα να υλοποιήσει τους δικούς του τρόπους. Το shawn όμως έχει φροντίσει από μόνο του να έχει τα πιο διαδεδομένα μοντέλα από πριν έτοιμα.



Στην πρώτη φωτογραφία έχουμε το πιο απλό και πολυχρησιμοποιημένο μοντέλο. Στο μοντέλο αυτό βλέπουμε 2 κόμβους που επικοινωνούν μεταξύ τους με την λογική της απόστασης που λέει ότι η ένταση του σήματος είναι αντιστρόφως ανάλογη του τετραγώνου της απόστασης. Αν λοιπόν δύο κόμβοι έχουν μεταξύ τους την κατάλληλη απόσταση επικοινωνούν, διαφορετικά δεν είναι εφικτό να μιλήσουν απευθείας.

Στην δεύτερη περίπτωση έχουμε σχεδόν το ίδιο μόνο που προστίθεται και η ιδέα της εξασθένησης του σήματος όσο αυξάνεται η απόσταση από κάποια τιμή ρ_1 μέχρι κάποια άλλη τιμή ρ_2 . Μέχρι την τιμή ρ_1 το σήμα είναι τέτοιο που θα έχουμε σίγουρα επιτυχία αποστολής.

Στην τρίτη περίπτωση έχουμε ανομοιόμορφο τρόπο αποστολής μηνυμάτων. Εδώ υπάρχει και η πιθανότητα να μπορεί ο ένας κόμβος να στείλει μήνυμα σε κάποιον άλλο ο οποίος όμως να αδυνατεί να του απαντήσει απευθείας. Υπάρχουν και άλλα μοντέλα όπως είναι το μόνιμης σύνδεσης στο οποίο είναι σαν να έχουμε καλωδιώσει τον κόμβο στο gateway και το chain στο οποίο συνδυάζουμε μερικά ή και όλα τα παραπάνω μοντέλα επικοινωνίας.

Μοντέλο ακμών Δίνει στον προγραμματιστή την δυνατότητα να έχει εποπτική εικόνα του δικτύου που εξομοιώνει. Είναι η αναπαράσταση του δικτύου αισθητήρων σε γραφο-θεωρητική μορφή. Για την σωστή αναπαράσταση, το μοντέλο ακμών ρωτάει συνέχεια το μοντέλο επικοινωνίας για να γνωρίζει πως θα φτιάξει τις ακμές. Τα διαφορετικά μοντέλα εδώ αναφέρονται στον τρόπο με τον οποίο αποθηκεύονται οι γράφοι (την μορφή δηλαδή).

Επιγραμματικά κάποια από τα μοντέλα που χρησιμοποιούνται είναι:

- **Lazy Edge Model** . Το μοντέλο ελάχιστης εργασίας, χρησιμοποιείται μόνο για μικρό αριθμό από κόμβους με μικρή κινητικότητα και επικοινωνία.
- **Grid Edge Model**. Το μοντέλο με δίκτυα, χρησιμοποιεί δύο δίκτυα για να βρίσκει την θέση των κόμβων.
- **List Edge Model**. Κρατάει όλα το δίκτυο στην μνήμη του, και έχει μεγάλο χρόνο κατασκευής που γίνεται μόνο μια φορά στην αρχή. Συνεπώς υποστηρίζει μόνο στατικά δίκτυα.
- **Fast List Model**. Είναι συνδυασμός των δύο παραπάνω.

Μοντέλο αποστολής/παραλαβής Η ύπαρξη του μοντέλου επικοινωνίας δεν αποτελεί από μόνη της καλή εξομοίωση όλων των λόγων για τους οποίους μπορεί να αποτύχει η επικοινωνία. Ακόμα και αν υπάρχει αμφίδρομη επικοινωνία μεταξύ δυο κόμβων και πάλι είναι δυνατό να έχουμε αποτυχίες στην αποστολή και παραλαβή μηνυμάτων.

Μια περίπτωση κατά την οποία μπορεί να συμβεί κάτι τέτοιο είναι για παράδειγμα όταν έχουμε ταυτόχρονη αποστολή μηνυμάτων. Το μηνύματα αυτά μπορεί να επηρεάσουν το ένα το άλλο με αποτέλεσμα να έχουμε εξασθένηση της ποιότητας του σήματος, απώλεια και καθυστέρηση πακέτων.

Η επιλογή του μοντέλου και εδώ έχει να κάνει με το τι θέλουμε να εξομοιώσουμε. Αν το θέμα μας είναι φτιάχνουμε έναν αλγόριθμο που καταπολεμάει το φαινόμενο της λανθασμένης αποστολής, φυσικά και θα επιλέξουμε μεγάλο βαθμό απωλειών και φθοράς. Αν ο στόχος μας είναι να μετρήσουμε την ταχύτητα του αλγορίθμου μας, ακόμα και ένα απλό μοντέλο χωρίς απώλειες μπορεί να μας κάνει. Αρκεί να έχουμε υπόψη μας ότι στην πραγματικότητα θα καθυστερήσει παραπάνω (λόγο error correction και επαναλαμβανόμενων αποστολών μηνυμάτων).

Επιπλέον μοντέλα Στο shawn δίνεται η δυνατότητα ρύθμισης της εξομοίωσης και με την χρήση κάποιων άλλων μοντέλων που μοντελοποιούν για παράδειγμα την κίνηση των αισθητήρων, εισαγωγή τυχαίων μεταβλητών και αποστάσεων των κόμβων.

Ο ταξινομητής διαδικασιών:

Ο sequencer είναι το κέντρο ελέγχου της εξομοίωσής μας. Από αυτόν ετοιμάζεται ο κόσμος της εξομοίωσης, αρχικοποιούνται και παραμετροποιούνται τα μοντέλα που θα χρησιμοποιηθούν σε σχέση με την είσοδο που δίνει ο χρήστης για την εξομοίωση. **Αποτελείται από τρία βασικά μέρη, τις διαδικασίες εξομοίωσης, τον ελεγκτή εξομοίωσης τον χρονιστή γεγονότων.**

Διαδικασίες εξομοίωσης Είναι κομμάτια κώδικα τα οποία γράφει ο χρήστης του εξομοιωτή και τα παρέχει στο shawn . Δεν είναι άμεσα συνδεδεμένα με την εξομοίωση του αλγορίθμου (δεν επηρεάζουν το πώς εκτελείται ο αλγόριθμος), αλλά έχουν πρόσβαση σε ολόκληρη την εξομοίωση και στα γεγονότα που συμβαίνουν σε οποιοδήποτε σημείο της και σε οποιοδήποτε κόμβο. Παραδείγματα τέτοιων διαδικασιών είναι η διαχείριση της εξομοίωσης, η δημιουργία κεντρικοποιημένων αλγορίθμων και η συγκέντρωση δεδομένων από ανεξάρτητους κόμβους. Τις διαδικασίες αυτές τις διαχειρίζεται ο ελεγκτής εξομοίωσης.

Ο ελεγκτής εξομοίωσης Ενεργεί σαν μία κεντρική αποθήκη όλων των υλοποιήσεων των μοντέλων και των διαδικασιών εξομοίωσης. Είναι ο συνδεδετικός κρίκος μεταξύ του πυρήνα του shawn και του χρήστη. Στην πράξη αυτό που κάνει είναι να παραμετροποιεί τις διαδικασίες σε σχέση με το αρχείο παραμετροποίησης που παρέχει ο χρήστης και μετά να τις τρέχει. Όπως και τα περισσότερα κομμάτια του shawn μπορεί να παρ αλλαχτεί με στόχο ακόμα μεγαλύτερο ή μικρότερο αυτοματοποιημένο έλεγχο. Στην αρχική του δομή όμως παίρνει παραμέτρους είτε από αρχείο ή από την εκτέλεση προγράμματος.

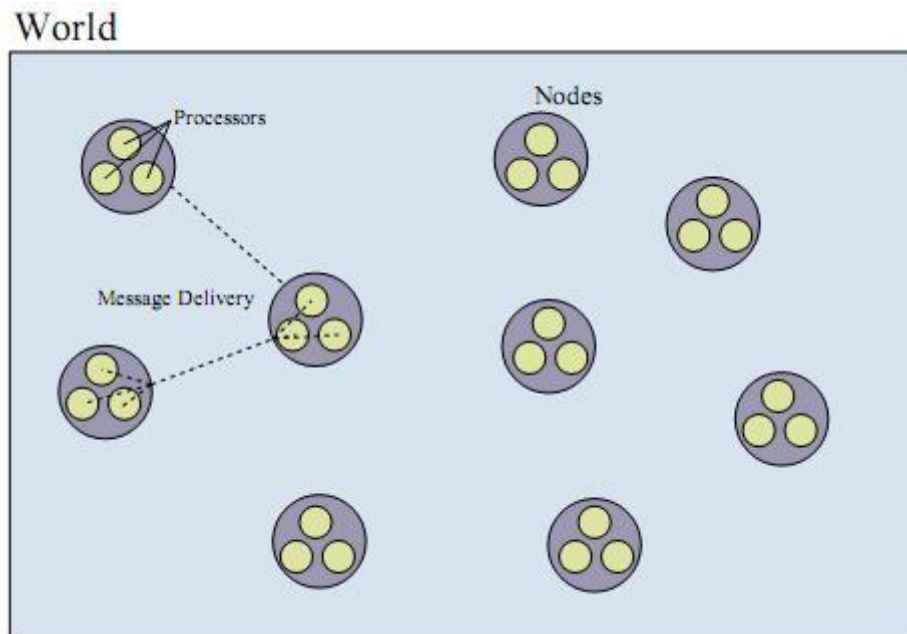
Μια δεύτερη υλοποίηση, το jshawn δίνει την δυνατότητα μέσω scripts να παραμετροποιήσουμε την εκτέλεση της με πιο περίπλοκες κατασκευές από δεδομένα εισόδου.

Χρονιστής γεγονότων Είναι ο ελεγκτής του χρόνου για το shawn . Μοντελοποιεί τον χρόνο που περνάει μεταξύ των γεγονότων και παρέχει σε οποιοδήποτε αντικείμενο της εξομοίωσης την δυνατότητα να δηλώσει ότι θέλει να κάνει κάτι σε μια χρονική στιγμή και ο χρονιστής αναλαμβάνει να τον ειδοποιήσει. Η εξομοίωση προχωράει από το ένα χρονικό γεγονός στο επόμενο και σε κάθε ένα από αυτά ενημερώνει όσα αντικείμενα ενδιαφέρονται. Η διαδικασία σταματάει όταν όλα τα αντικείμενα είναι εκτός λειτουργίας, ή με το πέρας του συνολικού χρόνου που επιτρέπεται σε μία εξομοίωση να διαρκήσει.

Με τον τρόπο αυτό που διαχειριζόμαστε τα γεγονότα, αρχικά κερδίζουμε από το γεγονός ότι δεν κάνουμε κλασικό χώρισμα του χρόνου σε αντιστοίχιση με κάτι άλλο (πχ cpu ticks) πράγμα που μας δίνει μεγαλύτερη ακρίβεια (κινήτης υποδιαστολής) στην υλοποίησή μας. Επιπλέον δεν γίνονται αχρείαστες ενέργειες απέναντι σε μη ενεργούς κόμβους συνέχεια, για να ελέγχουμε αν χρειάζονται να εκτελέσουν κάτι. Τέλος μας δίνεται η δυνατότητα απλού τρόπου εκτέλεσης μέσα στα προκαθορισμένα rounds (γύρους) όταν η ακρίβεια δεν είναι ιδιαίτερης σημασίας.

Το περιβάλλον εξομοίωσης:

Εδώ έχουμε τον ουσιαστικό κόσμο της εξομοίωσής μας. Στον κόσμο αυτό βρίσκονται όλες οι κορυφές και οι επεξεργαστές. Εδώ θα δούμε ξεχωριστά τι είναι κάθε ένα από αυτά και πως τα χρησιμοποιούμε για να φτιάξουμε την εξομοίωσή μας.



Ο κόσμος περιγράφεται πέραν από τους κόμβους, από το μέγεθός του αλλά και από τα μοντέλα τα οποία τον χαρακτηρίζουν σε σχέση με την λειτουργία του. Στην πράξη παρέχει πληροφορίες για την εξομοίωση, όπως είναι ο χρόνος της και οι κόμβοι που έχουν τελειώσει την εργασία τους. Μέσω του αντικειμένου κόσμου έχουμε επίσης πρόσβαση σε όλους τους κόμβους, πληροφορία ως προς το ποιοι κόμβοι απαρτίζουν τις γειτονιές κ.α.

Οι κόμβοι. Περιέχουν τους επεξεργαστές που θα τρέξουν τους αλγόριθμους μας. Μπορούμε να δούμε τον κόμβο σαν την φυσική υπόσταση ενός γενικού σκοπού αισθητήρα με πολλές δυνατότητες.

Οι κόμβοι αναλαμβάνουν την αποστολή και παραλαβή μηνυμάτων από τον υπόλοιπο πληθυσμό του κόσμου, και ο κάθε ένας έχει ένα μοναδικό χαρακτηριστικό αλφαριθμητικό το οποίο όμως είναι δυνατόν να αλλάξει κατά την διάρκεια της εξομοίωσης. Το shawn παρέχει επιπλέον έναν κόμβο «ειδικό» που λειτουργεί διαφορετικά από τους υπόλοιπους.

Ο επεξεργαστής. Αυτός είναι ο πυρήνας και ο λόγος ύπαρξης του shawn. Ενώ μέχρι τώρα είδαμε εργαλεία που ρυθμίζουν την εξομοίωση και το περιβάλλον της, στον επεξεργαστή θα δούμε τον ίδιο τον αλγόριθμο που θέλουμε να δοκιμάσουμε. Μπορεί να θέλουμε να εκτελέσουμε παραπάνω από έναν ταυτόχρονα (π.χ. έναν αλγόριθμο και μια εφαρμογή για στατιστικά δεδομένα στο συγκεκριμένο επεξεργαστή) και το shawn μας δίνει αυτή την δυνατότητα. Ο επεξεργαστής αναλαμβάνει μερικός και την αποστολή των μηνυμάτων. Μπορεί να έχει τρεις φάσεις, ενεργός, κοιμισμένος και ανενεργός. Αν όλοι οι επεξεργαστές είναι ανενεργοί τότε

και ο κόμβος είναι ανενεργός. Αν όλοι οι κόμβοι είναι ανενεργοί τότε ο κόσμος είναι ανενεργός και η εξομοίωση έχει τελειώσει.

Ο Εξομοιωτής NS-2 και λόγοι επιλογής του shawn. Ο παραπάνω εξομοιωτής είναι ίσως ο πιο πολυ-χρησιμοποιημένος για την διερεύνηση των ad-hoc δικτύων. Παρέχει έναν τεράστιο αριθμό από δυνατότητες για οποιοδήποτε είδος δικτύου, είναι και αυτός ανοιχτού κώδικα με μεγάλη κοινότητα υποστήριξης που καθημερινά αυξάνει τις δυνατότητές του. Εξαιτίας της μεγάλης χρησιμοποίησής του υπάρχει αρκετή τεκμηρίωση πράγμα που τον κάνει ακόμα πιο εύκολο στην χρήση του.

Στα αρνητικά του έχουμε την δυσκολία στην εξομοίωση ακόμα και απλών περιπτώσεων λόγω του μεγάλου βαθμού ανάλυσης των πραγματικών καταστάσεων. Πολλά από τα έτοιμα μοντέλα που υπάρχουν στο shawn εδώ πρέπει να τα φτιάξει ο χρήστης, ενώ δεν υποστηρίζονται μεγάλοι αριθμοί από κόμβους. Από την άλλη βέβαια υπάρχει γραφικό περιβάλλον στο οποίο μπορεί ο χρήστης να βλέπει σε «πραγματικό χρόνο» την διαδικασία.

Όπως είπαμε και νωρίτερα ένας ακόμα λόγος για την χρήση του shawn είναι η ειδικεισή του σε δίκτυα αισθητήρων και ο μεγάλος αριθμός κόμβων που υποστηρίζει λόγω της απλότητας μοντέλων που χρησιμοποιεί. Υπάρχει και ένας ακόμα λόγος, ο οποίος είναι η δυνατότητα χρησιμοποίησης κώδικα από απευθείας υλοποίηση αλγορίθμου για πραγματικές δοκιμές. Χρησιμοποιήσαμε λοιπόν για την ανάπτυξη του αλγορίθμου την βιβλιοθήκη wiselib για την οποία θα μιλήσουμε παρακάτω, η οποία συνδυάζεται άριστα με το shawn αλλά όχι και με το ns-2.

4.2.2 Η βιβλιοθήκη wiselib

Wisebed (Wireless Sensor Network Testbeds) . Είναι ένα έργο που αποτελεί μια συντονισμένη προσπάθεια από εννέα διαφορετικά ακαδημαϊκά και ερευνητικά ιδρύματα, που ξεκίνησε το 2008 και αναμένεται να ολοκληρωθεί το 2011. Ο στόχος του έργου είναι η παροχή μιας δομής διασυνδεδεμένων περιβαλλόντων από μεγάλα δίκτυα αισθητήρων με την λογική της κοινής αντιμετώπισης θεμάτων υλικού, προγραμμάτων, αλγορίθμων κλπ. Σκοπεύει στο να δείξει πως ετερογενή δίκτυα μπορούν να συνθέσουν καλά δομημένες αρχιτεκτονικές. Αυτό θα επιτρέψει έρευνα σε πολύ μεγαλύτερη έκταση από αυτήν που γίνεται τώρα, η οποία θα υποστηρίζει διαφορετικών επιπέδων ποιότητας που θα καθορίζεται από το υλικό.

Η κίνηση αυτή ευελπιστούμε να οδηγήσει σε έναν ενιαίο αυτό τρόπο λειτουργίας για τέτοια δίκτυα, σε αντίθεση με την κατάσταση τώρα που βλέπουμε να αποτελείται από ατομικές προσπάθειες, αν και όπως φαίνεται οδηγούμαστε προς αυτήν την κατεύθυνση. Για τον λόγο αυτό υλοποιούνται στα πλαίσια του Wisebed όλοι οι αλγόριθμοι, οι μηχανισμοί και τα πρωτόκολλα που προέκυψαν από σύγχρονες έρευνες ώστε να υπάρχουν οι κατάλληλες δομές και να επιλέγονται από τους ερευνητές στον τομέα που τις χρειάζονται για τις δικές τους έρευνες.

Ο κώδικας που προκύπτει δοκιμάζεται εντατικά σε μεγάλες εξομοιώσεις, από τις οποίες οι ομάδες του wisebed παίρνουν τα κατάλληλα αποτελέσματα τα οποία πολλές φορές οδηγούν την έρευνας προς νέους τομείς. Όταν το έργο προχωρήσει οι δομές και τα δίκτυα θα δοθούν στην ευρωπαϊκή επιστημονική κοινότητα για περαιτέρω έρευνα και αξιολόγηση.

Στα πλαίσια του wisebed και από την προσπάθεια να δημιουργηθούν κοινές δομές στα ασύρματα δίκτυα αισθητήρων, δημιουργήθηκε και εξελίσσεται και σήμερα μία βιβλιοθήκη αλγορίθμων που λειτουργεί ανεξάρτητα με το περιβάλλον στο οποίο

εκτελείται (για να μπορεί να αντεπεξέλθει στην διαφορετικότητα ανάμεσα στα πολλά είδη υλικού στα δίκτυα αισθητήρων). Η βιβλιοθήκη αυτή ονομάζεται wiselib (Wireless Sensor Network Library), και ήταν και η επιλογή μας για την υλοποίηση των αλγορίθμων χρωματισμού στην εργασία αυτή.

Wiselib

Στο σημείο αυτό θα δούμε τα βασικά στοιχεία της βιβλιοθήκης, τον τρόπο με τον οποίο λειτουργεί αυτή και καταφέρνει να είναι λειτουργική σε διαφορετικά περιβάλλοντα και την αρχιτεκτονική της. Πέραν από τον κυριότερο στόχο της, να λύσει το πρόβλημα της απουσίας πρακτικών εφαρμογών παρά την ύπαρξη πολλών στην θεωρία, η Wiselib στοχεύει και στα παρακάτω:

Στόχοι της βιβλιοθήκης

- **Ανεξαρτησία από πλατφόρμες.** Η βιβλιοθήκη μπορεί να χρησιμοποιηθεί σε έναν μεγάλο αριθμό από διαφορετικές πλατφόρμες υλικού.
- **Ανεξαρτησία από λειτουργικά συστήματα.** Μπορεί επίσης να λειτουργήσει σε διάφορα λειτουργικά συστήματα είτε βασίζονται σε γλώσσα προγραμματισμού C όπως το Contiki είτε σε C++ .
- **Δυνατότητα αλλαγής κομματιών.** Οι εφαρμογές μπορεί να αποτελούνται από διάφορα κομμάτια τα οποία συνομιλούν μέσω καλά ορισμένων διεπαφών. Αυτά τα κομμάτια μπορούν εύκολα να αλλάξουν στην περίπτωση που βρεθούν άλλα καλύτερα χωρίς να επηρεάσουν τα υπόλοιπα μέρη.
- **Ύπαρξη μεγάλου εύρους αλγορίθμων.** Η Wiselib αποτελείται από μεγάλο αριθμό αλγορίθμων όπως δρομολόγησης, συγχρονισμού, ομαδοποίησης τοπολογιών κ.α.
- **Αλγόριθμοι πολλών επιπέδων.** Η βιβλιοθήκη παρέχει ακόμα την δυνατότητα να χρησιμοποιηθούν άλλοι, έτοιμοι αλγόριθμοι για την υλοποίηση μίας ολοκληρωμένης εφαρμογής. Τα πρωτόκολλα που υλοποιούνται χρησιμοποιούνται συγχρονισμένα επεκτείνοντας τις δυνατότητές τους.
- **Συμβατότητα με τα Standards .** Η Wiselib έχει υλοποιηθεί σε ένα καλά ορισμένο υποσύνολο της ISO C++ .Αυτό έχει συγκριτικά πλεονεκτήματα σε σχέση με τροποποιημένες εκδόσεις της γλώσσας όπως είναι η καλύτερη υποστήριξη σε ανανεώσεις και τεκμηρίωση.
- **Επεκτασιμότητα και αποτελεσματικότητα.** Τέλος είναι ικανή να εκτελείται σε μία μεγάλη ποικιλία από υλικό - από 8-bitεπεξεργαστές με ελάχιστη μνήμη, μέχρι καινούργιες πλατφόρμες με πολλές δυνατότητες. Οι αλγόριθμοι πρέπει να είναι δυνατόν να εκτελούνται σε αδύναμα τερματικά αλλά αν υπάρχει η δυνατότητα για μεγάλη υπολογιστική ισχύ να την εκμεταλλεύονται.

Η αρχιτεκτονική της Wiselib Η κεντρική ιδέα στον σχεδιασμό της βιβλιοθήκης, ήταν βασισμένη σε μοντέλα και ιδέες. Η γενική αρχιτεκτονική που προέκυψε φαίνεται και στην παρακάτω φωτογραφία. Σαν ιδέα αναφερόμαστε σε ένα στοιχειώδες κομμάτι που από μόνο του δεν αποτελεί αλγόριθμο αλλά ένα βασικό αφηρημένο μέλος όλων των αλγορίθμων αυτού του είδους. Είναι εμφανές λοιπόν το ότι υπάρχει μία τέτοια ιδέα για κάθε σύνολο από αλγορίθμους είτε αυτοί είναι δρομολόγησης ή οποιασδήποτε άλλης κατηγορίας.

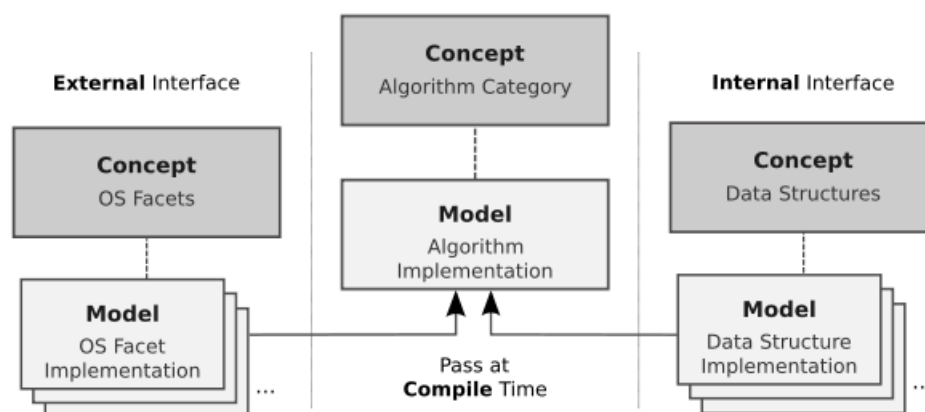


Fig. 1. Wiselib Architecture

Κάθε αλγοριθμικό μοντέλο αποτελείται από μία ή περισσότερες ιδέες ενώ το τελικό αποτέλεσμα είναι ένα πρότυπο που αναμένει διάφορες παραμέτρους για να εκτελεστεί στην πράξη. Οι παράμετροι αυτοί μπορεί να είναι είτε δομές δεδομένων ή όψεις λειτουργικών συστημάτων (OS Facet).

Η όψη του λειτουργικού συστήματος στην ουσία αντιπροσωπεύει την σύνδεση με το πραγματικό λογισμικό που έχει το υλικό στο οποίο θα χρησιμοποιήσουμε την βιβλιοθήκη. Είναι αφηρημένοι τύποι και συναρτήσεις που δεν υλοποιούνται αλλά πραγματοποιούν ένα επίπεδο το οποίο χρησιμοποιούμε στον προγραμματισμό. Το επίπεδο αυτό αντικαθίσταται με πραγματικές συναρτήσεις όταν αυτές δοθούν σαν είσοδος πριν την εκτέλεση του προγράμματος. Επιπλέον με τις δομές δεδομένων ορισμένες με παρόμοιο τρόπο μπορούμε να τροφοδοτούμε από τα ετερογενή περιβάλλοντα με διάφορες δομές που εκμεταλλεύονται τις εκάστοτε δυνατότητες.

Η διεπαφή με τον έξω κόσμο Η διεπαφή αυτή μας δίνει πρόσβαση στο πραγματικό λειτουργικό σύστημα. Αυτό μπορεί να είναι κάποιο λειτουργικό σύστημα αισθητήρων όπως τα CONTIKI ή το iSense , εξομοιωτές όπως είναι το shawn που χρησιμοποιούμε κ.α. Η ιδέα είναι να δίνονται βασικά στοιχεία που περιέχουν βασικούς τύπους και συναρτήσεις, που στην συνέχεια θα πρέπει να υλοποιηθούν για το κάθε λειτουργικό σύστημα ξεχωριστά. Στην συνέχεια οι υλοποιήσεις αυτές μπορούν να δίνονται στα πρότυπα ως είσοδοι. Με τον τρόπο αυτό ο μεταγλωττιστής μπορεί να καθορίσει ποια είναι η διασύνδεση που θα χρησιμοποιήσει κατά την μεταγλώττιση.

Στην πράξη έπρεπε να επιλυθούν δύο προβλήματα. Το πρώτο είναι ότι το πραγματικό λειτουργικό σύστημα πρέπει να είναι υλοποιημένο σε κάποια γλώσσα

προγραμματισμού συμβατή με τις C/C++ . Αυτό οδηγεί σε πολλούς περιορισμούς. Το δεύτερο είναι ότι η διασύνδεση με τις λειτουργίες του λογισμικού του υλικού, μπορεί να γίνεται με διαφορετικό τρόπο από τα διαφορετικά λειτουργικά συστήματα.

Το πρώτο πρόβλημα επιλύεται με την χρήση μόνο ορισμών τύπων και στατικών συναρτήσεων στην διεπαφή με τον έξω κόσμο. Αυτή η λύση έχει δύο πλεονεκτήματα. Το πρώτο είναι ότι οι στατικές συναρτήσεις μπορούν να δοθούν σαν παράμετροι όπου αναμένονται κανονικές C συναρτήσεις. Δεύτερον δεν είναι αναγκαίο οποιοδήποτε στιγμιότυπο της κλάσεις της διεπαφής και συνεπώς δεν απαιτείται μνήμη για δείκτες.

Το δεύτερο πρόβλημα επιλύεται με την εισαγωγή ενός επιπλέον τύπου λειτουργικού συστήματος. Παρόλα αυτά απαιτείται ένας δείκτης σε αυτά σε κάθε στατική συνάρτηση της διεπαφής.

Κάποια από τα κομμάτια που την απαρτίζουν είναι:

- **Το μοντέλο OS** . Παρέχει πληροφορία για το λειτουργικό σύστημα, και καθορίζει μια τιμή για τον χρονισμό την αποστολή μηνυμάτων κ.α.
- **Radio** . Επιτρέπει την αποστολή και την παραλαβή μηνυμάτων.
- **Χρονιστής**. Επιτρέπει την δήλωση γεγονότων και κλήσεων στον κατάλληλο χρόνο.
- **Ρολόι**. Δίνει και θέτει την κανονική ώρα συστήματος.

Η διεπαφή με τον έξω κόσμο είναι αυτή που δίνει την δυνατότητα να υλοποιηθεί μία φορά ο αλγόριθμος και να μεταγλωττίζεται για διαφορετικές εξωτερικές διεπαφές. Συνεπώς το λειτουργικό σύστημα μπορεί να έχει γραφτεί σε οποιαδήποτε γλώσσα βασισμένη στην C/C++ .

Εσωτερική διεπαφή Η διεπαφή αυτή περιέχει τις δομές δεδομένων που αναφέραμε νωρίτερα και χρησιμοποιούνται από τους αλγορίθμους. Παραδείγματα τέτοιων δομών αποτελούν οι πίνακες δρομολόγησης που αποτελούνται από δυάδες "συνήθως" κόμβων και καθορίζουν πιο θα είναι το επόμενο βήμα στο δίκτυο για τον συγκεκριμένο προορισμό ή πίνακες με συγκεκριμένες ιδιότητες. Επιπλέον εδώ βρίσκουμε και οδηγίες για την υλοποίηση δομών μηνυμάτων με ανεξαρτησία σε σχέση με το περιβάλλον στο οποίο θα χρησιμοποιηθούν.

Αλγόριθμοι Στο επίκεντρο της βιβλιοθήκης βρίσκονται οι αλγόριθμοι. Προαναφέρθηκε ότι χρειάζεται μόνο μία φορά να υλοποιηθούν για να χρησιμοποιηθούν σε πραγματικά περιβάλλοντα. Μεταγλωττίζονται για διαφορετικές πλατφόρμες με την χρήση των διαφορετικών διεπαφών σύνδεσης με τον έξω κόσμο. Η Wiselib προσφέρει αυτή την στιγμή τους παρακάτω αλγορίθμους:

- Δρομολόγησης
- Ομαδοποίησης
- Συγχρονισμού
- Έρευνας τοποθεσίας
- Διαχείρισης ενέργειας
- Επιπέδου MAX

Εργαλεία Στα εργαλεία βρίσκουμε διάφορα εύχρηστα κομμάτια κώδικα που μας βοηθάνε να δημιουργήσουμε ανεξάρτητο σε σχέση με τις πλατφόρμες κώδικα. Παράδειγμα είναι η pSTL που χρησιμοποιείται για την στατική μόνο δέσμευση μνήμης.

4.3 Χαρακτηριστικά υλοποίησης

4.3.1 Tora Routing

Σχεδόν οποιοσδήποτε αλγόριθμος, απαιτεί την αποστολή μηνυμάτων σε κόμβους οι οποίοι δεν βρίσκονται μέσα στην περιοχή την οποία μπορεί να τον δει. Σε τέτοιες περιπτώσεις απαιτείται η αποστολή από τον έναν κόμβο σε κάποιον άλλο μέχρι το μήνυμα να φτάσει στον σωστό αποδέκτη. Μιλήσαμε νωρίτερα για τεχνικές πλημμύρας, οι οποίες όμως έχουν εκθετικά μεγάλο αριθμό από μηνύματα που απαιτούνται για την σωστή παράδοση μηνυμάτων. Για τον λόγο αυτό έπρεπε και εμείς να χρησιμοποιήσουμε έναν αλγόριθμο που να καθορίζει τις διαδρομές για τους παραλήπτες μας σωστά και γρήγορα με τον κατά δύναμη ελάχιστο αριθμό αποστολών.

Οι αρχικές υλοποιήσεις έγιναν με έναν αλγόριθμο που προσφέρει η βιβλιοθήκη Wiselib και ονομάζεται δρομολόγηση δέντρου. Το πρόβλημα με τον αλγόριθμο αυτό είναι η αβεβαιότητα σε κάθε σημείο ότι το μήνυμα θα φτάσει στον παραλήπτη. Για να επιλυθεί υπήρχε η απαίτηση για κάθε μήνυμα που στέλνουμε να πρέπει να ξανα-δημιουργηθούν οι πίνακες δρομολόγησης όταν το μήνυμα δεν έφτανε με την ελπίδα κάποια στιγμή να υπάρχει η διαδρομή. Αν και η πιθανότητα αυτή ήταν μεγάλη τελικά προέκυψε ότι δεν ήταν καθόλου ικανοποιητική η απόδοση του αλγορίθμου και κατ' επέκταση έπρεπε να καταφύγουμε σε άλλες λύσεις.

Η λύση στην οποία καταφύγαμε ήταν ο αλγόριθμος Tora που σχεδιάστηκε από τον Vincent Park του πανεπιστημίου Maryland και υλοποιήθηκε για την βιβλιοθήκη Wiselib στο ερευνητικό και ακαδημαϊκό ινστιτούτο του πανεπιστημίου Πατρών.

Λειτουργία Η λειτουργία του Tora δεν μοιάζει με άλλους αλγόριθμους του είδους του. Αναφέραμε νωρίτερα δύο κατηγορίες αλγορίθμων δρομολόγησης. Ο TORA ανήκει στους on-demand πράγμα που τον καθιστά ιδιαίτερα χρήσιμο στην περίπτωση μας λόγω της μη ύπαρξης αναγκαιότητας για δρόμους ανάμεσα στους κόμβους. Η μεγάλη όμως διαφορά με άλλους αλγορίθμους είναι ότι ο συγκεκριμένος δεν ακολουθεί την λογική του μικρότερου μονοπατιού. Σύμφωνα με αυτήν ο αλγόριθμος ψάχνει την όσο πιο δυνατότερο μικρή διαδρομή για να ταξιδεύουν τα δεδομένα.

Κατασκευάζει και διατηρεί ένα κατευθυνόμενο άκυκλο γράφο με βάση τον προορισμό. Φροντίζει να μην υπάρχουν τρεις διαφορετικοί κόμβοι με το ίδιο ύψος. Η πληροφορία κινείται από το μεγαλύτερο στο μικρότερο ύψος και με μία έννοια η πληροφορία κατεβαίνει από ψηλότερα ύψη στο χαμηλότερα. Διατηρώντας σύνολο με ταξινομημένα ύψη μόνιμα ο TORA εγγυάται κίνηση χωρίς κύκλους, αφού δεν μπορεί η πληροφορία να ταξιδέψει προς τα πίσω.

Η κεντρική σχεδιαστική ιδέα ήταν να διατηρηθούν τα μηνύματα ελέγχου στο ελάχιστο με ένα πολύ μικρό σύνολο κόμβων να επηρεάζονται από μια τοπική αλλαγή στην τοπολογία. Για να το καταφέρει αυτό οι κόμβοι διατηρούν πληροφορία για

όλους τους κόμβους σε απόσταση ενός βήματος. Το πρωτόκολλο πραγματοποιεί τις εξής λειτουργίες.

- Δημιουργία διαδρομής
- Διατήρηση-διόρθωση πληροφορίας διαδρομής
- Διαγραφή διαδρομής

Κατά την διάρκεια των δύο πρώτων οι κόμβοι χρησιμοποιούν μία μετρική ύψους για να δημιουργήσουν τον γράφο που προαναφέραμε. Στο επόμενο βήμα οι σύνδεσμοι αποδίδονται στους κόμβους με βάση το σχετικό ύψος των γειτόνων. Όταν κινούνται οι κόμβοι ο γράφος σπάει και πρέπει να ξανα-δημιουργηθεί. Η διαγραφή γίνεται με ένα πακέτο CLR το οποίο πλημμυρίζει το δίκτυο.

4.3.2 C/C++, Templates, delegates

Εξορισμού από την χρήση της βιβλιοθήκης Wiselib χρησιμοποιούμε την γλώσσα προγραμματισμού C/C++ για την υλοποίηση των αλγορίθμων μας. Ο εξομοιωτής shawn επίσης ρυθμίζεται, πέραν από τα αρχεία ρυθμίσεων, με χρήση της ίδιας γλώσσας, ενώ για την δημιουργία πολλαπλών εξομοιώσεων χρησιμοποιήσαμε unix scripting, python και τέλος το εργαλείο gnuplot για την δημιουργία των γραφικών μας παραστάσεων.

Στο σημείο αυτό θα παραθέσουμε την ιδιαίτερη αξία που έχουν η χρήση των templates-προτύπων και τον delegates . Τα πρότυπα είναι ένα χαρακτηριστικό της C++ που επιτρέπει στις συναρτήσεις και στις κλάσεις να λειτουργούν με γενικευμένους τύπους. Αυτό επιτρέπει λειτουργία με διαφορετικούς τύπους δεδομένων χωρίς να είναι απαραίτητο να ξαναγραφτούν οι κώδικες όταν αυτοί μετατρέπονται. Εύκολα φαίνεται πως είναι ένα εργαλείο με μεγάλη αξία για τους προγραμματιστές, ιδιαίτερα όταν μιλάμε για υπερφόρτωση τελεστών και κληρονομικότητα.

Ένα απλό παράδειγμα φαίνεται στο παρακάτω κομμάτι κώδικα:

```
template <typename T>
const T& max(const T& x, const T& y)
{
    if(y < x)
        return x;
    return y;
}
```

Η παραπάνω απλή συνάρτηση μας δίνει την μεγαλύτερη τιμή από δύο παραμέτρους. Το τι είναι αυτές είναι αδιάφορο για την γλώσσα προγραμματισμού. Το μόνο που δηλώνουμε είναι ότι ανήκει στον τύπο T που θα τον καθορίσουμε αργότερα όταν θα χρησιμοποιήσουμε την συνάρτηση για τους δικούς μας σκοπούς.

Delegation ή αλλιώς *ανάθεση* είναι άλλο ένα πολύ χρήσιμο εργαλείο που μας προσφέρει η C++. Μπορούμε να αντιμετωπίσουμε την ανάθεση ως δυναμική κληρονομικότητα σε πραγματικό χρόνο. Ουσιαστικά αναφέρεται σε ένα αντικείμενο το οποίο βασίζεται σε κάποιο άλλο για να κάνει κάποιες λειτουργίες. Χρησιμοποιείται εκτενώς σε λειτουργικά συστήματα όπως είναι αυτό της Apple όπου η διαχείριση των παραθύρων γίνεται από την συνάρτηση που αναθέτει το κάθε παράθυρο ξεχωριστά. Στην περίπτωση της wiselib για παράδειγμα η συνάρτηση αποστολής μηνυμάτων ανατίθεται στην εκτέλεση με βάση το λογισμικό που χρησιμοποιεί το κάθε υλικό. Ετερογενή περιβάλλοντα θα ήταν σχεδόν αδύνατον να υλοποιούν

αλγορίθμους με μεγάλα κομμάτια κώδικα ίδια αν δεν υπήρχε η ανάθεση και τα templates . Χωρίς την ανάθεση δεν θα μπορούσαμε να ορίζουμε γενικευμένες συναρτήσεις και χωρίς τα πρότυπα οι συναρτήσεις αυτές δεν θα ήταν δυνατόν να λειτουργούν με γενικευμένους τύπους χωρίς να είναι απαραίτητο να ξαναγραφτεί ο κώδικας.

4.3.3 Δομές δεδομένων

Σε κάθε υλοποίηση αλγορίθμων είναι πολύ σημαντική η επιλογή των τύπων δεδομένων οι οποίες θα χρησιμοποιηθούν. Πέραν από τους τυπικούς γνωστούς τύπους όπως είναι οι ακέραιοι διαφόρων μεγεθών, οι χαρακτήρες και οι τύποι κινητής υποδιαστολής, υπάρχουν δομές όπως οι πίνακες από τέτοιους τύπους που είναι και η πιο απλή δομή. Πέραν όμως από τους πίνακες υπάρχουν στην C/C++ και άλλες δομές που προσφέρουν μεγάλη ευκολία και δυνατότητες σε συγκεκριμένες διεργασίες. Φυσικά, σε συστήματα όπως είναι τα δίκτυα αισθητήρων με τις περιορισμένες υπολογιστικές δυνατότητες απαιτείται μεγάλη προσοχή στην επιλογή τους, μιας και πολλές φορές απαιτούν μεγάλη ισχύ και χώρο για να λειτουργήσουν απρόσκοπτα.

Μια δομή που χρησιμοποιήθηκε και αξίζει να αναφέρουμε είναι οι πίνακες δρομολόγησης που υπήρχαν ήδη υλοποιημένοι στην βιβλιοθήκη wiselib . Προφανώς χρησιμοποιούνται από τον TORA και είναι ασφαλείς στην χρήση τους για όλους τους διαφορετικούς αισθητήρες που υποστηρίζονται.

Πέραν αυτών μεγάλη χρησιμότητα για εμάς είχαν οι χάρτες (maps) της C/C++ . Οι χάρτες έχουν πολύ καλή συμπεριφορά όταν χρειαζόμαστε να αποθηκεύσουμε πληροφορία με βάση κάποιο κλειδί (για παράδειγμα μία τιμή με βάση τον κόμβο). Δυστυχώς όμως δεν είναι καλοί αν πρέπει να έχουμε ταξινομημένα δεδομένα. Μαζί με τους χάρτες χρησιμοποιήσαμε ειδικά στο περιβάλλον εξομίωσης τα διανύσματα vectors για να έχουμε ουσιαστικά μια δομή που λειτουργεί σαν την δομή στοίβας. Τα διανύσματα φάνηκαν χρήσιμα όταν έπρεπε να υπολογίσουμε την διάμετρο του δικτύου μας και την πυκνότητα (όχι την γραφο-θεωρητική / βλέπε κεφάλαιο 6).

Παρά τις αρκετές επιλογές από δομές που είχαμε δημιουργήσαμε μία επιπλέον δομή. Είναι ουσιαστικά ένα πίνακας δύο διαστάσεων ο οποίος παραμένει πάντα ταξινομημένος σε σχέση με την δεύτερή του διάσταση. Αποθηκεύουμε σε αυτόν δυάδες (χρωμάτων-αριθμών κόμβων) και κάθε φορά που προσθέτουμε ένα χρώμα ή αυξομειώνουμε την αντίστοιχη τιμή τους ο πίνακας διορθώνεται αυτόματα και γρήγορα.

Κεφάλαιο 5

Αλγόριθμοι και Εξομοίωση

5.1 Εισαγωγή

Αφού δόθηκαν τα περιβάλλοντα στα οποία υλοποιήσαμε τους αλγορίθμους μας και δόθηκαν κάποια από τα κυριότερα τεχνικά χαρακτηριστικά τους θα προχωρήσουμε στο κεφάλαιο αυτό με την παρουσίαση τους.

Ξεκινάμε από τον απλό προσεγγιστικό αλγόριθμο που σχεδιάστηκε από τους Σπυράκη-Παναγοπούλου, καθώς και κάποιους άλλους που υπάρχουν ήδη. Μετά βλέπουμε τους δικούς μας με την σειρά την οποία υλοποιήθηκαν, που ουσιαστικά είναι και η σειρά πολυπλοκότητας τους ως προς την πραγματοποίησή τους.

Πρώτος είναι ο απλός αλγόριθμος με τον δικαστή, μία απλή μεταβολή του προσεγγιστικού, ώστε να παρέχεται με έναν απλό τρόπο η πληροφορία που απαιτούν οι κόμβοι. Ακολουθεί η εκδοχή του με πολλαπλούς δικαστές, ο οποίος είναι και αυτός σειριακός, αποσκοπεί όμως σε λιγότερα μηνύματα και κατέπέκταση σε μικρότερους χρόνους εκτέλεσης με το μειονέκτημα του χειρότερου χρωματικού αριθμού. Ο τρίτος αλγόριθμός μας αποτελείται από πολλαπλούς δικαστές οι οποίοι λειτουργούν ταυτόχρονα. Το πόσο βελτιώνονται οι χρόνοι και πόσο χειροτερεύει το αριθμός σε χρώματα θα το δούμε σε επόμενο κεφάλαιο.

Πέραν των αλγορίθμων που δημιουργήσαμε βασιζόμενοι στον προσεγγιστικό αλγόριθμο, υλοποιήσαμε και δύο αλγορίθμους πλήρως κατανεμημένους και παράλληλους που διαφοροποιούν την αρχική λύση. Ο ένας από αυτούς ονομάζεται αλγόριθμος δυο βημάτων, ο οποίος θεωρεί σαν δίκτυο στο οποίο ανήκει ένας κόμβος μόνο όσους κόμβους απέχουν το πολύ δύο βήματα από τον ίδιο, και επεκτείνεται σε όσα βήματα θέλουμε, ενώ ο δεύτερος επιλέγει τυχαία μέσα στο δίκτυο από $\log(n)$ κόμβους για να φτιάξει την γειτονιά του.

Τέλος πριν να προχωρήσουμε στην αξιολόγηση, αξίζει να παρουσιαστεί ο τρόπος με τον οποίον δημιουργήθηκαν τα πειραματικά δεδομένα. Θα δούμε πως φτιάξαμε τυχαίους γράφους αυξανόμενης πυκνότητας και σταθερής διαμέτρου και αντίστροφα, σταθερής πυκνότητας και αυξανόμενης διαμέτρου. Κλείνοντας το κεφάλαιο παρουσιάζεται επίσης και ο τρόπος ομαδοποίησης των κόμβων κάτω από έναν δικαστή.

5.2 Προτεινόμενοι αλγόριθμοι

5.2.1 Υπάρχοντες αλγόριθμοι

Αναφερθήκαμε σε προηγούμενα κεφάλαια σε διάφορους αλγορίθμους χρωματισμού που έχουν μελετηθεί. Πιο συγκεκριμένα στο δεύτερο κεφάλαιο αναφέραμε την επίλυση με χρήση της θεωρίας παιγνίων, ενώ στο τρίτο, στο κομμάτι που περιγράφηκαν τα καταναμεμένα συστήματα αναφέραμε κάποιες άλλες καταναμεμένες λύσεις. Ο αναγνώστης, προτείνεται να ανατρέξει στο συγκεκριμένα κεφάλαια, μιας και παρακάτω θα παρουσιάσουμε αλγορίθμους που σχετίζονται με τους προηγούμενους σε έναν βαθμό και ιδιαίτερα με αυτόν του πρώτου κεφαλαίου.

5.2.2 Ο Αλγόριθμος με δικαστή

Με βάση την δουλειά σε θεωρητικό επίπεδο που έχει πραγματοποιηθεί μέχρι σήμερα και ιδιαίτερα με βάση τον αλγόριθμο των Σπυράκη-Παναγοπούλου παρουσιάζουμε εδώ έναν πρακτικό τρόπο για την υλοποίηση του. Αφορά τα ασύρματα δίκτυα αισθητήρων αλλά και οποιοδήποτε άλλο καταναμεμένο σύστημα. Η περιγραφή θα γίνει σε δύο φάσεις: Την αρχικοποίηση του αλγορίθμου και κατ' επέκταση του ίδιου του δικτύου και την φάση που εξελίσσεται το παιχνίδι. Στην πρώτη φάση περιγράφουμε τον τρόπο με τον οποίο η πληροφορία ταξιδεύει μέσα στο δίκτυο έτσι ώστε κάθε κόμβος να έχει πρόσβαση στα χρώματα που υπάρχουν στο δίκτυο και είναι ικανά να ανεβάσουν το τελικό τους σκορ.

Αρχικοποίηση Η μόνη γνώση που θεωρούμε ότι έχουν ήδη οι κόμβοι είναι ένας μοναδικός αριθμός ταυτοποίησης. Παρά την θεωρητική παρατήρηση ότι κάτι τέτοιο απαιτεί υπολογιστικούς πόρους για να επιτευχθεί, στην πράξη μπορούμε να βρούμε τέτοιους αριθμούς ειδικά σε μοντέρνα δίκτυα-ένα τέτοιο παράδειγμα είναι η Mac Address που υπάρχει σε κάθε συσκευή σύνδεσης-. Στην περίπτωση της εξομοίωσής μας το Shawn παρέχει αυτοματοποιημένο μηχανισμό απόδοσης τέτοιων αριθμών.

Το πρώτο βήμα των κόμβων είναι να δημιουργήσουν γνωριμίες με τους γύρω τους. Οι κόμβοι πρέπει να γνωρίζουν ποιος θα διευθύνει τις διεργασίες (αυτός που θα είναι ο δικαστής), αλλά και τους γείτονές τους. Ένα μήνυμα αρχικοποίησης αποστέλλεται από κάθε κόμβο γνωστοποιώντας έτσι την ύπαρξή τους σε κάθε άλλον που μπορεί να τους ακούσει, δηλαδή σε όποιον βρίσκεται στην εμβέλεια τους ή αλλιώς στους γείτονές τους. Το μήνυμα αυτό το στέλνουν όλοι οι κόμβοι για να γνωρίζουν όλοι τον ακριβή αριθμό των γειτόνων ο οποίος αποθηκεύεται από τον αλγόριθμο για όλη την διάρκεια της εκτέλεσής του.

Από την πλευρά του ο δικαστής στέλνει το μήνυμα αρχικοποίησης δικαστή. Αυτό έχει την ιδιότητα να εξαπλώνεται μέσα στο δίκτυο με την παρακάτω λογική: Κάθε κόμβος που λαμβάνει ένα μήνυμα αρχικοποίησης δικαστή για πρώτη φορά το ξαναστέλνει μέσα στο δίκτυο για να το λάβουν οι δικοί του γείτονες. Αυτοί με την σειρά τους το ξαναστέλνουν κ.ο.κ. Με αυτόν τον τρόπο πιστοποιούμε ότι το μήνυμα θα φτάσει σε όλους τους κόμβους που είναι συνδεδεμένοι στο δίκτυό μας. Φανερό είναι ότι το μήνυμα αυτό το στέλνουν οι κόμβοι μόνο μία φορά κατά την είσοδό τους. Στην περίπτωση που μπου καινούργιοι κόμβοι στο δίκτυο, οι ήδη υπάρχοντες γείτονές τους μπορούν αν τους ενημερώσουν για τον δικαστή.

Στην δεύτερη φάση του αλγορίθμου έχουμε δύο εργασίες. Οποιοσδήποτε κόμβος λάβει το μήνυμα αρχικοποίησης δικαστή και αποθηκεύσει τον αριθμό του (μο-

ναδικό αριθμό αναγνώρισης) στέλνει μέσω του δικτύου (με τον αλγόριθμο TORA) ένα μήνυμα αιτήματος χρώματος. Με αυτόν τον τρόπο ο δικαστής αναγνωρίζει που θα πρέπει να στείλει το πρώτο μήνυμα απόδοσης χρώματος, αλλά και ποιοι είναι οι κόμβοι του δικτύου (οι κόμβοι που έχει αναλάβει να χρωματίσει στις επόμενες υλοποιήσεις). Ο ίδιος αποδίδει στον εαυτό του το χρώμα μηδέν (0). Προφανώς ο αλγόριθμος χρωματισμού δεν έχει σκοπό να χρωματίσει το δίκτυο αλλά να αποδώσει κοινές ιδιότητες. Συνεπώς το χρώμα αντικαθίσταται στην περίπτωση μας από αριθμούς. Ο δικαστής διατηρεί μία μεταβλητή τύπου ακεραίου αριθμού στην οποία αποθηκεύει το επόμενο από το τελευταίο χρώμα που έδωσε. Την τιμή της μεταβλητής αυτής την στέλνει τώρα στον κόμβο που έκανε το αίτημα. Μόλις παραλάβει ο κόμβος που ζήτησε το χρώμα το μήνυμα αυτό, δημιουργεί ένα νέο μήνυμα που επιβεβαιώνει την παραλαβή. Εδώ τελειώνει το βήμα αυτό με τον δικαστή να έχει μία λίστα με κόμβους και τα χρώματά τους. Υπάρχει εδώ ένας μικρός χρόνος αναμονής από τον δικαστή για να είμαστε σίγουροι ότι όλοι οι κόμβοι του δικτύου έχουν ζητήσει και πάρει χρώμα.

Με την παραπάνω μέθοδο έχουμε αποκτήσει ένα δίκτυο χρωματισμένο με τον χειρότερο όμως δυνατό χρωματικό αριθμό. Κανένας κόμβος όμως δεν έχει χρώμα ίδιο με τον γείτονά του. Αυτό φυσικά θα μπορούσαμε να το καταφέρουμε παίρνοντας σαν χρώμα τον μοναδικό αριθμό ταυτοποίησης που αποδίδεται από πριν. Αυτό όμως δεν θα μας γλίτωνε τα μηνύματα και τον φόρτο μιας και σε κάθε περίπτωση ο δικαστής πρέπει να έχει γνώση χρωμάτων και γνώση κόμβων, αλλά και οι κόμβοι πρέπει να έχουν μερική γνώση της γειτονιάς τους.

Εκτέλεση Μία δεύτερη υπόθεση που θα κάνουμε σε αυτό το σημείο είναι ότι γνωρίζουμε από πριν πιο είναι το κέντρο του δικτύου, και μάλιστα έχουμε τοποθετήσει τον δικαστή μας στο κέντρο του. Στην εξομοίωσή μας φυσικά αυτό το βρίσκουμε μέσω ενός αλγορίθμου που υπολογίζει τις πραγματικές αποστάσεις από το κέντρο του δικτύου και αποδίδει την ιδιότητα του δικαστή στον πιο κοντινό. Στην πραγματικότητα όμως μπορεί να χρειαστεί ο δικαστής να είναι πιο ισχυρός σε υπολογιστική δύναμη από τους υπόλοιπους κόμβους, και συνεπώς να μην είναι δυνατόν να επιλεγεί μέσω αλγορίθμου.

Στην φάση της εκτέλεσης ο δικαστής θα διαχειριστεί το παιχνίδι μέχρι αυτό να φτάσει στην θεμιτή για εμάς ισορροπία Nash. Ξεκινώντας από έναν τυχαίο κόμβο ο δικαστής εκτελεί τις παρακάτω ενέργειες:

- Σημειώνουμε στην αρχή το χρώμα με το μεγαλύτερο σκορ και το αποστέλλουμε στον τυχαία επιλεγμένο κόμβο. Την στιγμή που το μήνυμα καταφτάνει στον κόμβο αυτός πρέπει να αποφασίσει αν θα το κρατήσει ή όχι. Η απόφαση αυτή λαμβάνεται αφού ο κόμβος στείλει με την σειρά του ένα μήνυμα σε όλους τους γείτονές του ζητώντας την άδειά τους. Αυτοί θα ελέγξουν άμεσα το χρώμα τους και αν είναι κοινό θα στείλουν το μήνυμα απόρριψης, ενώ διαφορετικά θα στείλουν επιβεβαίωση. Όταν τελικά ο κόμβος που είναι υποψήφιος για αλλαγή χρώματος λάβει την απαντήσεις ίσες σε αριθμό με τον αριθμό των γειτόνων του, αποφασίζει για το αν θα το κρατήσει ή όχι. Ένα μόνο μήνυμα απόρριψης αρκεί για να μην το κρατήσει μιας και κάτι τέτοιο οδηγεί σε λανθασμένο χρωματισμό. Όποια και να είναι η απόφασή του την στέλνει τελικά στον δικαστή.
- Μόλις παραλαμβάνει την απάντηση του κόμβου αποφασίζει την επόμενη κίνησή του. Είτε αποθηκεύει την αλλαγή και προχωράει σε άλλον κόμβο αν πάρει

θετική απάντηση αποδοχής, ή στέλνει το επόμενο χρώμα από την λίστα του. Αλλαγή σε κόμβο αν δεν λάβει θετική απάντηση, θα πραγματοποιηθεί μόνο αν εξαντληθούν τα χρώματα που έχει στην λίστα του και είναι ικανά να ανεβάσουν το τοπικό σκορ του υπό εξέταση κόμβου.

- Πιστοποιεί ότι δεν υπάρχει κόμβος που να μπορεί να βελτιώσει το σκορ του περαιτέρω. Ο μόνος τρόπος που υπάρχει για να συμβεί αυτό, είναι κάθε φορά που ο δικαστής λαμβάνει ένα μήνυμα αποδοχής αλλαγής χρώματος, να ελέγχουμε όλους τους άλλους κόμβους με την μέθοδο του δεύτερου βήματος αν είναι δυνατόν να αλλάξει τον χρώμα τους. Αν όχι τότε ο αλγόριθμος ολοκληρώνεται και αποδίδει έναν χρωματισμό που ικανοποιεί τα θεωρητικά όρια που δόθηκαν και αποδέχτηκαν από τους Σπυράκη-Παναγοπούλου.

Ο λόγος που πρέπει να ξανακοιτάξουμε όλους τους κόμβους κάθε φορά που κάποιος αποδέχεται ένα χρώμα χρειάζεται λίγο καλύτερη εξήγηση. Ένα παράδειγμα που το κάνει πιο κατανοητό είναι το παρακάτω. Ας φανταστούμε ότι ο κόμβος k του δικτύου μας αποδέχεται το χρώμα κόκκινο το οποίο το έχουν n άλλοι κόμβοι. Το χρώμα πράσινο το έχουν έστω $n-2$ και θα μπορούσε να το πάρει και αυτό. Τρεις άλλοι κόμβοι μη γειτονικοί του k αλλάζουν το χρώμα τους σε μπλέ. Ότι και να έχει κάνει νωρίτερα ο k τώρα μπορεί να βελτιώσει το σκορ του. Οι αλλαγές λοιπόν που γίνονται σε οποιονδήποτε κόμβο μπορεί να προκαλέσουν (το πιθανότερο) αλλαγές σε κόμβους που έχουν ήδη επιλέξει κάποιο χρώμα.

5.2.3 Σειριακός αλγόριθμος πολλαπλών δικαστών

Όπως και στην προηγούμενη επίλυση με τον έναν δικαστή έτσι και εδώ έχουμε πάλι τις δύο φάσεις της αρχικοποίησης και της εκτέλεσης του παιχνιδιού. Δεν κρίνεται απαραίτητη η εκτενής παρουσίαση του αλγορίθμου πολλαπλών δικαστών εδώ μιας και μοιράζεται πολλά από τα χαρακτηριστικά του με τον αλγόριθμο με έναν. Εδώ θα δούμε μόνο τις αλλαγές που έγιναν στον προηγούμενο για να προκύψει ο αλγόριθμος αυτός.

Πριν όμως μπούμε στην φάση της αρχικοποίησης πρέπει να πάρουμε κάποιες αποφάσεις. Προκύπτουν εδώ ερωτήματα όπως το πόσοι θα είναι οι δικαστές για τα καλύτερα δυνατά αποτελέσματα, αλλά και ποια θα είναι η θέση τους. Τα δύο αυτά ερωτήματα είναι ακόμα υπό διερεύνηση και αποφασίσαμε να διερευνήσουμε το πρόβλημα ως προς τον βέλτιστο αριθμό δικαστών τοποθετώντας τους δικαστές με έναν λογικά καλό τρόπο αν όχι τον βέλτιστο.

Στην υλοποίησή μας του αλγορίθμου υποστηρίζουμε μέχρι πέντε διαφορετικούς δικαστές. Στην πραγματικότητα θα μπορούσαμε εύκολα να επεκτείνουμε την λύση μας αφού ο μόνος περιορισμός είναι στην ομαδοποίηση που γίνεται από πριν και στην απόδοση περιοχών στους δικαστές και όχι στον κώδικα που εκτελείται ο οποίος υποστηρίζει όσους δικαστές θέλουμε.

Ο τρόπος ομαδοποίησης και απόδοσης κόμβων είναι επιγραμματικά ο παρακάτω:

- Στον έναν δικαστή επιλέγουμε όπως και στον προηγούμενο αλγόριθμο αφού είναι ουσιαστικά ο ίδιος.
- Στους δύο παίρνουμε τους κόμβους στο κέντρο του πάνω αριστερά τεταρτημόριου του κύβου που δημιουργεί ο κόσμος μας, και του κάτω δεξιά. Μπορεί να πάρουμε τους αντίστοιχους δύο αν έχουν «καλύτερες αποστάσεις»

- Στους τρεις επιλέγουμε την δυάδα όπως στους δύο και προσθέτουμε τον κεντρικό.
- Στους τέσσερις παίρνουμε όλους τους κόμβους που αναφέραμε χωρίς τον κεντρικό τον οποίο συμπεριλαμβάνουμε στους πέντε.

Αρχικοποίηση Στο ζήτημα ποιοι δικαστές αναλαμβάνουν ποιους κόμβους, η απάντηση δίνεται κατά την εκτέλεση. Μιας και είναι μεγάλη γενίκευση που δεν ανταποκρίνεται στην πραγματικότητα, να δεχτούμε ισοβαρή τα βήματα από κόμβο σε κόμβο σε έναν ασύρματο δίκτυο αισθητήρων, δεν θεωρήσαμε σωστό να αντι-στοιχούμε κόμβους μόνο με μόνο κριτήριο την απόσταση αυτή.

Αφήνουμε λοιπόν το δίκτυο από μόνο του να πάρει αυτήν την απόφαση επι-λέγοντας στην φάση της αρχικοποίησης ο κάθε κόμβος τον δικαστή του με μόνο κριτήριο την ταχύτητα. Δεν μας ενδιαφέρουν τα βήματα ή η πραγματική απόστα-ση από τους δικαστές αφού αποφασίσαμε σαν δικαστή μας τον πρώτο του οποίου μάθαμε το μοναδικό αναγνωριστικό. Βέβαια για να λειτουργήσει αυτό απαιτείται ταυτόχρονη αρχικοποίηση και ενεργοποίηση των δικαστών.

Η υπόλοιπη διαδικασία είναι ίδια με του απλού αλγορίθμου πέραν μίας διαφο-ράς. Ο προσεγγιστικός αλγόριθμος χρωματισμού προϋποθέτει την εκκίνηση του παιγνίου από έναν σωστό χρωματισμό όσο μεγάλος και να είναι ο χρωματικός α-ριθμός. Αν και εδώ ξεκινήσουν οι δικαστές από το μηδέν σαν δικό τους χρώμα και δίνουν αύξοντα αριθμό σε κάθε έναν που ζητάει χρώμα είναι πολύ πιθανόν να έχουμε γείτονες στο δίκτυό μας με το ίδιο χρώμα. Για να το αποφύγουμε αυτό ακολουθούμε την παρακάτω τακτική:

Ο δικαστής παίρνει σαν προσωπικό του χρώμα τον αριθμό του μοναδικού ανα-γνωριστικού του πολλαπλασιασμένο με έναν πολύ μεγάλο ακέραιο αριθμό, αρκούν-τως μεγαλύτερο ώστε να ξεπερνάει αρκετά τον συνολικό αριθμό των κόμβων(π.χ. για 1000 κόμβους ο αριθμός μας είναι το 10000 ή μεγαλύτερος). Έστω ο δικαστής με αναγνωριστικό αριθμό 1 πολλαπλασιασμένο με το 10000 έχει τώρα αναγνωρι-στικό 10000 και στην τοπική του μεταβλητή του μέγιστου χρώματος αποθηκεύει το 100001. Όποτε ένας κόμβος ζητήσει χρώμα θα του δώσει το 100001,...,10000ν όπου ν το χρώμα του ν-οστού κόμβου που θα αιτηθεί. Με αυτόν τον τρόπο εξα-σφαλίζουμε αποδεκτό αρχικό χρωματισμό.

Εκτέλεση Αρκετές διαφορές βλέπουμε και στην φάση της εκτέλεσης του αλγο-ρίθμου. Κάθε δικαστής έχει πλήρη γνώση της κατάστασης του κομματιού του δι-κτύου που του ανατέθηκε στην αρχικοποίηση. Πράγμα που σημαίνει μερική γνώση της κατάστασης σε ολόκληρο το δίκτυο είτε κατά την διάρκεια της εξομοίωσης ή της εκτέλεσης σε κανονικό hardware . Με βάση αυτήν την γνώση θα προσπαθήσει να χρωματίσει το δίκτυο που του αναλογεί.

Ήταν απαραίτητο στο σημείο αυτό, να βρούμε μία μέθοδο να συντονιστούν οι δικαστές για να διασφαλιστεί ολοκλήρωση του αλγορίθμου αλλά και αποδεκτός χρωματισμός. Για την επίτευξη του στόχου αυτού θέτουμε σε κάθε δικαστή έναν αριθμό προτεραιότητας ίσο με τον αριθμό του αναγνωριστικού του. Οι κόμβοι γνωρίζοντας το αναγνωριστικό του δικαστή τους γνωρίζουν αυτόματα και την προτεραιότητά τους. Έτσι κάθε φορά που κάποιος από τους κόμβους ζητάει να μάθει αν μπορεί να μεταβεί σε ένα χρώμα που θα βελτιώσει το σκορ του στέλνει μαζί με το χρώμα και τον αριθμό προτεραιότητάς του. Αν ο αριθμός αυτός είναι μεγαλύτερος από αυτόν που έχει ο κόμβος που καλείται να απαντήσει, τότε αυτός απαντάει πάντα θετικά και αντιστρόφως φυσικά.

Μόλις ένας δικαστής ολοκληρώσει την λειτουργία του τότε στέλνει την λίστα με τα χρώματα και τα αντίστοιχα σκορ τους στον επόμενο. Εδώ πρέπει να τονίσουμε ότι και αυτός ο αλγόριθμος είναι σειριακός. Ποτέ δυο δικαστές δεν λειτουργούν ταυτόχρονα. Μόνο όταν τελειώσει ο πρώτος προχωράμε στον επόμενο χρησιμοποιώντας πλέον και πληροφορία από τον προηγούμενο.

Ο αλγόριθμος ολοκληρώνεται μόλις όλοι οι δικαστές ολοκληρώσουν την διεργασία τους. Θεωρούμε πως αν έκαναν έστω άλλη μια επανάληψη θα είχαμε και καλύτερα αποτελέσματα, όμως αυτό μάλλον θα αναιρέσει ή θα μειώσει αρκετά την απόδοση σε σχέση με την ταχύτητα που κερδίσαμε από τον πρώτο αλγόριθμο.

5.2.4 Παράλληλος αλγόριθμος πολλαπλών δικαστών

Η τελευταία έκδοση του αλγορίθμου είναι ένας συνδυασμός της πρώτης και τις δεύτερης. Όπως ακριβώς και στον σειριακό αλγόριθμο με του πολλαπλούς δικαστές, θα παρουσιάσουμε μόνο τις διαφορές που έχουν αξία σε σχέση με τους προηγούμενους. Η κεντρική ιδέα πίσω από τον σχεδιασμό του, ήταν να έχουμε τον προηγούμενο αλγόριθμο με την κυριότερη διαφορά να είναι η παράλληλη απόδοση χρωματισμών από όλους τους δικαστές ταυτόχρονα με μόνο ένα πολύ μικρό κομμάτι να είναι "κάποιες" φορές σειριακό. Κατά μία έννοια ο κάθε δικαστής είναι ένα νήμα όπως αυτά που έχουμε στους παράλληλους αλγόριθμους.

Οι μεγάλες διαφορές είναι στο τέλος του χρωματισμού όλων των κόμβων με την μέθοδο που ακολουθήσαμε παραπάνω. Σε αυτήν την περίπτωση μετά το πέρας του χρωματισμού εφαρμόζουμε μία τεχνική επιδιόρθωσης σφαλμάτων στα αποτελέσματά μας. Υπάρχει όμως και μία αλλαγή στην αρχικοποίηση του συστήματος όπου τώρα όλοι οι δικαστές εφαρμόζουν τον αρχικό χρωματισμό ενδιαφερόμενοι για την ορθότητά του μονάχα μέσα στα πλαίσια των κόμβων που τους έχουν ανατεθεί, αδιαφορώντας κατά κάποιον τρόπο για το αποτέλεσμα ως προς τον συνολικό χρωματισμό του δικτύου.

Στην πραγματικότητα ο τρόπος που επιλέγουν τα χρώματα όταν οι κόμβοι ζητάνε για πρώτη φορά και γνωστοποιούν την παρουσία τους στον δικαστή για τον οποίον έλαβαν γρηγορότερα την πληροφορία ύπαρξής τους, σχεδόν πιστοποιεί την ύπαρξη δύο γειτονικών κόμβων με το ίδιο χρώμα.



Παράδειγμα απόδοσης
κόμβων με 4
δικαστές

Στο παραπάνω παράδειγμα βλέπουμε τις ομάδες που δημιουργήθηκαν από την ομαδοποίηση μας με διαφορετικά χρώματα. Φαίνεται ότι υπάρχουν "μοναχικοί" κατά κάποιον τρόπο κόμβοι ανάμεσα σε άλλους διαφορετικής ομάδας. Όπως και στον σειριακό αλγόριθμο, ο αρχικός γενικευμένος χρωματισμός γίνεται αποδίδοντας ο κάθε δικαστής το μηδενικό χρώμα στον εαυτό του και ύστερα έναν αύξοντα αριθμό

σε κάθε άλλον (βλέπε αρχικοποίηση απλού αλγορίθμου με έναν δικαστή). Οποιοδήποτε λοιπόν χρώμα από τα πρώτα και να πάρει κάποιος από τους κόμβους που προαναφέραμε, είναι πολύ πιθανό να συμπίσει με το χρώμα κάποιων από τους γείτονές του που ανήκουν σε άλλη ομάδα. Σίγουρα όσο πιο μεγάλο είναι το δίκτυο η πιθανότητα αυτή δεν μειώνεται γιατί οι αποστάσεις του κόμβου και τον γειτόνων του είναι χρονικά, σχεδόν ίσες από τους δικαστές στους οποίους ανήκουν, και συνεπώς θα πάρουν πιθανότατα χρώματα με μικρή διαφορά ως προς τον αύξοντα αριθμό. Υπάρχουν λύσεις και για αυτό παρακάτω όμως θα δούμε ότι δεν είναι απαραίτητες λόγω της φύσης του αλγορίθμου. Η υπόλοιπη αρχικοποίηση γίνεται με την ίδια μέθοδο που ακολουθήσαμε και στον σειριακό αλγόριθμο πολλαπλών δικαστών.

Η διαδικασία απόδοσης χρωμάτων γίνεται όπως και νωρίτερα με τις άδειες αποδοχής να δίνονται και πάλι με βάση τον αριθμό προτεραιότητας του κάθε κόμβου. Συμπεραίνουμε εδώ ότι πολλά από τα προβλήματα που προέκυψαν στον τρόπο του αρχικού χρωματισμού θα επιλυθούν. Φυσικά στις οριακές συνθήκες ανταγωνισμού όπου παίρνουν ταυτόχρονα το χρώμα τους δύο κόμβοι γειτονικοί που ανήκουν σε διαφορετικές ομάδες μπορεί να προκύψουν σφάλματα, τα οποία αναλαμβάνει το επόμενο βήμα να επιδιορθώσει.

Επιδιόρθωση σφαλμάτων Ο δικαστής με την μεγαλύτερη προτεραιότητα είναι και ο μοναδικός ο οποίος εξαιρείται από τις επιδιορθώσεις. Είναι ο μόνος ο οποίος τελειώνει και άμεσα στέλνει την πληροφορία των σκορ των χρωμάτων που έχει αναθέσει στον δικαστή με την επόμενη προτεραιότητα, αγνοώντας οποιαδήποτε διένεξη μεταξύ των χρωμάτων της ομάδας του και με τα χρώματα των κόμβων των γειτονικών ομάδων. Εδώ τελειώνει πλήρως η λειτουργία του.

Ο αμέσως επόμενος δικαστής λαμβάνει την λίστα των χρωμάτων και την ενσωματώνει στην δική του. Αν ο αλγόριθμος δεν έχει ολοκληρωθεί ακόμα σε αυτόν, τόσο το καλύτερο μιας και εκμεταλλεύεται και την καινούργια πληροφορία ως προς τα ποια χρώματα έχουν χρησιμοποιηθεί περισσότερο στο δίκτυο. Ειδικά σε ανομοιογενή δίκτυα ως προς τον χώρο αυτό μπορεί να παίζει σπουδαίο ρόλο στην ποιότητα των αποτελεσμάτων.

Στην περίπτωση αντίθετα που έχει τελειώσει τον χρωματισμό, ή όταν θα τελειώσει αν έλαβε το μήνυμα από τον προηγούμενο πιο νωρίς, ο δικαστής στέλνει ένα μήνυμα ειδοποίησης για μία τελική επιβεβαίωση ως προς τα σφάλματα. Στο σημείο αυτό ο ίδιος και όλοι του οι κόμβοι όταν θα παραλάβουν το μήνυμα θα ξεκινήσουν μία διαδικασία ερωτημάτων στους γειτονικούς τους κόμβους για να ελέγξουν την ορθότητα του χρώματός τους. Αν κανείς από τους γείτονες του κόμβου που έκανε το ερώτημα δεν έχει το ίδιο χρώμα με τον ίδιο τότε επιστρέφει στον δικαστή ένα μήνυμα επιβεβαίωσης ενώ σε αντίθετη περίπτωση στέλνει ένα μήνυμα λάθους.

Όταν θα λάβει απαντήσεις από όλους τους κόμβους ο δικαστής προχωράει και πάλι με την εκτέλεση του πρώτου (απλού) αλγορίθμου χωρίς φυσικά την αρχικοποίηση και μονάχα για όσους κόμβους έχουν πρόβλημα με το χρώμα τους. Αυτό δεν θα μας δώσει κατά πάσα πιθανότητα τα καλύτερα δυνατά αποτελέσματα, τα οποία θα πετυχαίναμε αν ο αλγόριθμος έτρεχε και πάλι για όλους τους κόμβους, αλλά εδώ το ζητούμενο είναι κυρίως η ταχύτητα. Κατά το δεύτερο αυτό πέρασμα όμως είναι πιθανόν να μην επαρκούν τα χρώματα που υπήρχαν από την αρχική απόδοση χρωμάτων. Για αυτό τον λόγο δίνονται με βάση την τοπική μεταβλητή του μέγιστου χρώματος, νέα χρώματα εκεί που δεν ικανοποιούν τα παλαιότερα. Το ίδιο γίνεται και για το προσωπικό χρώμα του δικαστή. Μόλις τελειώσει και

εδώ ο αλγόριθμος ο δικαστής στέλνει την πληροφορία στον επόμενο ο οποίος επαναλαμβάνει τις ίδιες κινήσεις.

Επεξηγήσεις πάνω στον αλγόριθμο Αυτή η μέθοδος που εφαρμόσαμε για την επιδιόρθωση λαθών, δεν είναι βέβαιο ότι φαίνεται ξεκάθαρα γιατί την εφαρμόζουμε. Στο σημείο αυτό θα προσπαθήσουμε να το εξηγήσουμε με έναν απλό τρόπο.

Ας φανταστούμε έστω μία εφαρμογή του αλγορίθμου πάνω σε έναν γράφο με τέσσερις δικαστές. Έστω ότι μετά το πέρας της εκτέλεσης του πρώτου σε προτεραιότητα δικαστή έχουμε διενέξεις με τους άλλους τρεις δικαστές στους κόμβους 1-2 με τον δεύτερο στους κόμβους 6-9 με τον τρίτο και στους κόμβους 6-9-32 με τον τέταρτο. Όταν τελειώσει ο δεύτερος αλγόριθμος θα διορθώσει την διένεξη αν υπάρχει με τον πρώτο δικαστή καθώς και όποια άλλη διένεξη έχει με αυτόν. Ο πρώτος δεν θα αλλάξει ποτέ ξανά τις τιμές του αφού έχει ολοκληρώσει την διεργασία του άρα δεν θα έχει ποτέ ξανά διένεξη με τον δεύτερο που και αυτός ολοκληρώθηκε. Το πρώτο ζευγάρι έδιωξε τις διενέξεις στον 1 και 2 κόμβους και στους αντίστοιχους του δύο. Με τον ίδιο τρόπο και ο τρίτος θα διορθώσει τους 6-9 με τον πρώτο και όποια άλλα κοινά προβλήματα είχε με τον πρώτο και τον δεύτερο και αντίστοιχα και ο τέταρτος. Σειριακά λύσαμε τα προβλήματα πρώτου-δεύτερου, πρώτου-τρίτου, δεύτερου-τρίτου, πρώτου-τέταρτου, δεύτερου-τέταρτου και τρίτου-τέταρτου και δείξαμε ότι μετά την διόρθωση δεν είναι δυνατόν να υπάρχει διένεξη μεταξύ των δύο που επιδιορθώθηκαν. Αφού λοιπόν επιδιορθώσαμε όλα τα ζευγάρια συνεπάγεται ότι ο χρωματισμός μας είναι σωστός.

5.2.5 Αλγόριθμοι περιορισμένης γνώσης πλήρως κατανεμημένοι

Όλες οι παραπάνω λύσεις στο πρόβλημα του χρωματισμού, απαιτούν την ύπαρξη κάποιου είδους συντονιστή. Ακόμα και στην περίπτωση που όλοι οι κόμβοι έχουν την δυνατότητα να αποφασίζουν για τον εαυτό τους και να διατηρούν ο καθένας ξεχωριστά την κατάσταση του δικτύου στις δομές δεδομένων που διαχειρίζονται, η διαδικασία ενημέρωσής τους κοστίζει πολύ σε μηνύματα, σε τέτοιο βαθμό που η εφαρμογή τους καθίσταται από πολύ ενεργοβόρα. Αν σκεφτούμε ότι για κάθε αλλαγή χρώματος απαιτείται να πλημμυρίζουμε το δίκτυο για να ενημερωθούν όλοι, αλλά και τα μηνύματα που πρέπει να σταλούν για να μπορούν να συντονίζονται οι κόμβοι μεταξύ τους, καταλαβαίνουμε ότι η τάξη μεγέθους των μηνυμάτων είναι πολύ μεγαλύτερη.

Όλα τα παραπάνω μαζί με το γεγονός ότι ο αλγόριθμος με τους δικαστές είναι σχεδόν σειριακός, ή έστω είναι παράλληλος μόνο για παραπάνω από έναν δικαστή μας οδήγησε σε άλλες λύσεις οι οποίες να μπορούν όσο το δυνατόν περισσότεροι κόμβοι να παίρνουν ταυτόχρονα αποφάσεις με τα λιγότερα δυνατά μηνύματα. Στο τμήμα αυτό θα παρουσιάσουμε κάποιες από αυτές.

Κατανεμημένος χρωματισμός δύο βημάτων Όπως και νωρίτερα υποθέτουμε ότι έχουμε έναν μοναδικό αριθμό που αντιστοιχεί σε κάθε κόμβο του δικτύου μας. Επιπλέον θεωρούμε ότι κάθε κόμβος είναι ικανός να διατηρεί μια λίστα με τις αξίες των χρωμάτων, όπως ακριβώς έκανε και ο δικαστής μας στις προηγούμενες εφαρμογές. Η μόνη διαφορά είναι ότι εδώ χρειαζόμαστε πολύ μικρότερες λίστες από ότι πριν όπως θα δούμε.

Ο κάθε κόμβος i διατηρεί με την θεωρία των διανυσμάτων ένα διάνυσμα n μεγέθους $k_i = (k_j)_{j \in [n]} \in N^n$. Για κάθε j , k_j αντιστοιχεί στην πληθικότητα της κλάσης χρώματος j , όπως αυτή υπολογίζεται από τον κόμβο i . Επιπλέον, κάθε κόμβος διατηρεί ένα σύνολο από απαγορευμένα χρώματα $F_i \in 2^{[n]}$, $|F_i| = \deg(i)$, που είναι στην ουσία τα χρώματα των γειτόνων του.

Υπάρχουν τρία είδη μηνυμάτων που μπορούν να λάβουν και να στείλουν οι κόμβοι:

1. $M(x, y, h, s)$, όπου $x, y \in [n] \cup \{o\}$ είναι χρώματα (ή μηδέν), $h \in \{1, 2\}$ δείχνει τον αριθμό των βημάτων, και s δείχνει το αναγνωριστικό του κόμβου. Όταν ένας κόμβος i λαμβάνει ένα τέτοιο μήνυμα, το αγνοεί αν $s = i$ ή $x = 0$, διαφορετικά θέτει $k_x^i := k_x^i + 1$ και $k_y^i := k_y^i - 1$
2. $I(x)$, όπου $x \in [n]$ και είναι χρώμα. Όταν λαμβάνει ο i το μήνυμα αυτό θέτει $F_i := F_i \cup \{x\}$
3. $R(f, i)$, όπου $f \in \{0, 1\}$ και $i \in [n]$. Το f αντικατοπτρίζει το αν έχει την επιθυμία να μεταβεί ο κόμβος σε άλλο χρώμα.

Τα μηνύματα αυτά παραμένουν ίδια και στον επόμενο αλγόριθμο.

Στον αλγόριθμο δυο βημάτων κάθε κόμβος ενημερώνεται για τα χρώματα που υπάρχουν στην γειτονιά του, και ο κόμβος με το μεγαλύτερο αριθμό χαρακτηριστικού που δεν είναι ικανοποιημένος από τον χρωματισμό του, μπορεί να κάνει αλλαγή στο χρωματισμό του, μεταβαίνοντας σε κάποιο άλλο τις γειτονιάς του. Οι κόμβοι που είναι άμεσοι γείτονες του καθώς και οι κόμβοι που είναι άμεσοι γείτονες των γειτόνων του ενημερώνονται για την αλλαγή αυτή. Αυτό είναι στην ουσία τα δύο βήματα. Οι κόμβοι καθορίζουν τον κόσμο τους μόνο από τα χρώματα των κόμβων που απέχουν από αυτούς μέχρι δύο βήματα.

Ακολουθούν δύο συναρτήσεις που χρησιμεύουν για την κατανόηση του αλγορίθμου:

Έλεγχος Ικανοποίησης(i)
Αν $\exists x \in [n]$ τέτοιο ώστε $x \neq c_i$ και $x \in F_i$ και $k_x \geq k_{c_i}^i$ τότε επέστρεψε 0 αλλιώς επέστρεψε 1
Εγλωιστικό Βήμα(i)
Χρώματα $:= x \in [n] : x \neq c_i, x \in F_i, k_x^i \geq k_{c_i}^i$ Βρές κάποιο $x \in$ Χρώματα τέτοιο ώστε $k_x^i \geq k_{c_i}^i$ για όλα τα $y \in$ Χρώματα $k_{c_i}^i := k_{c_i}^i - 1$ $c_i := x$ $k_{c_i}^i := k_{c_i}^i$ επέστρεψε c_i

Δεδομένων των βημάτων αυτών ο κάθε κόμβος πραγματοποιεί επαναλαμβανόμενα τον παρακάτω αλγόριθμο:

Πρωτόκολλο δύο βημάτων(i)
Αρχικοποίηση $c_i = id_i, k_j^i = 1$ για όλα τα $j \in [n]$
$F_i = \emptyset$
$m_i = 0$
Στείλε($I(c_i), t$) για όλα τα $t \in Neigh(i)$
Παρέλαβε($I(c_t), t$) για όλα τα $t \in Neigh(i)$
Ικανοποίηση = Έλεγχος Ικανοποίησης(i)
Στείλε($R(Iκανοποίηση), t$) για όλα τα $t \in Neigh(i)$
Παρέλαβε($R(f_t), t$) για όλα τα $t \in Neigh(i)$
—
Αν $f_t = 1$ για όλα τα $t > i$
$temp_i := c_i$
$c_i := \text{Εγωιστικό Βήμα}(i)$
$m_i := c_i$
—
Στείλε($M(m_i, temp_i, i, 1), t$) για όλα τα $t \in Neigh(i)$
Παρέλαβε($M(m_t, temp_t, 1, n_t), t$) για όλα τα $t \in Neigh(i)$
Στείλε($M(m_t, temp_t, 2, n_t), t$) για όλα τα $t \in Neigh(i)$
Παρέλαβε($M(m_t, temp_t, 2, n_t), t$) για όλα τα $t \in Neigh(i)$

Στην ουσία αυτό που γίνεται είναι η αποστολή ενημέρωσης στους γείτονες για το χρώμα που έχει ο κόμβος, και παραλαβή του ίδιου μηνύματος από τους υπόλοιπους για την δημιουργία της λίστας χρωμάτων που απαγορεύονται. Με βάση τα χρώματα αυτά ελέγχει αν είναι ικανοποιημένος και στέλνει την απόφασή του στην γειτονιά που βρίσκεται (1 και 2 βήματα μακριά). Μόλις ολοκληρωθεί η λήψη αυτών των μηνυμάτων, ελέγχει στην περίπτωση που δεν είναι ικανοποιημένος αν έχει την μεγαλύτερη προτεραιότητα από αυτούς που δεν είναι και αν ισχύει αυτό κάνει εγωιστικό βήμα. Στην τελευταία φάση ενημερώνονται οι γειτονιές για τις αλλαγές που έγιναν.

Υπάρχουν και διάφορες άλλες εκδοχές του αλγορίθμου βασισμένες στον αριθμό των βημάτων από τα οποία καθορίζεται μια γειτονιά. Ο αλγόριθμος τριών βημάτων για παράδειγμα, θεωρεί ότι ο κάθε κόμβος καθορίζει την λίστα των χρωμάτων του από όλους τους κόμβους που απέχουν το πολύ τρία βήματα μακριά του. Δοκιμές έδειξαν ότι δεν βελτιώνει αρκετά το συνολικό χρώμα ώστε να έχει νόημα η καθυστέρηση και η αύξηση των μηνυμάτων που προκαλεί.

Τυχαία-λογαριθμικός καταναεμημένος χρωματισμός Στον τυχαία-λογαριθμικό αλγόριθμο η λογική των γειτονιών μέσα στις οποίες θα δρουν οι κόμβοι παραμένει αλλά με την διαφορά ότι οι γειτονιές δεν θα είναι πλέον 2 βήματα μακριά, αλλά ο κάθε κόμβος θα διαλέγει έναν λογαριθμικό αριθμό σε σχέση με

το μέγεθος του δικτύου από γείτονες. Με βάση το δίκτυο αυτό θα εκτελεί τον αλγόριθμο. Βλέπουμε παρακάτω την φορμαλιστική εκδοχή του.

Για αρχή δίνεται μία υπο-ρουτίνα που περιγράφει την δημιουργία του πίνακα με τις πληθικότητες των χρωμάτων:

Υπολογισμός Πληθικότητας(i)
Δημιούργησε ένα υποσύνολο κόμβων $N_i \subset [n]$ από τυχαία και ανεξάρτητη επιλογή $O(\log n)$ κόμβων στο $[n]$. Για όλα τα $t \in N_i$ $k_{c_i}^i := k_{c_i}^i + 1$

Δεδομένης της ρουτίνας, ο κάθε κόμβος πραγματοποιεί επαναλαμβανόμενα τον παρακάτω αλγόριθμο:

Πρωτόκολλο Τυχαίο-Λογαριθμικά(i)
Αρχικοποίηση $c_i = id_i, k_j^i = 0$ για όλα τα $j \in [n], k_{c_i}^i := 1$ $F_i = \emptyset$ $m_i = 0$ Στείλε($I(c_i), t$) για όλα τα $t \in Neigh(i)$ Παρέλαβε($I(c_t), t$) για όλα τα $t \in Neigh(i)$ Υπολογισμός Πληθικότητας(i) Ικανοποίηση = Έλεγχος-Ικανοποίησης(i) Στείλε($R(I(c_i), t)$) για όλα τα $t \in Neigh(i)$ Παρέλαβε($R(f_t), t$) για όλα τα $t \in Neigh(i)$ Αν $f_t = 1$ για όλα τα $t > i$ $temp_i := c_i$ $c_i := \text{Εγλωιστικό Βήμα}(i)$ $m_i := c_i$ Στείλε($M(m_i, temp_i, i, 1), t$) για όλα τα $t \in Neigh(i)$ Παρέλαβε($M(m_t, temp_t, 1, n_t), t$) για όλα τα $t \in Neigh(i)$

Βλέπουμε τελικά, έναν αλγόριθμο με πολλές ομοιότητες με τον αντίστοιχο των δύο βημάτων. Η μόνη ουσιαστική διαφορά πέραν των κριτηρίων της επιλογής των γειτόνων, είναι ο υπολογισμός της πληθικότητας κάθε φορά (ενώ στον προηγούμενο αλγόριθμο είχαμε μερική ενημέρωση μόνο όταν γινόταν αλλαγή). Το δεύτερο είναι η κατάργηση της ενημέρωσης των γειτόνων που απέχουν παραπάνω από ένα βήμα.

5.3 Εξομοίωση

Πριν πάμε στα αποτελέσματα των αλγορίθμων, οφείλουμε να παρουσιάσουμε ένα μεγάλο μέρος της εργασίας που έγινε για να δημιουργηθεί το περιβάλλον της εξομοίωσης. Θα ξεκινήσουμε με την παρουσίαση του τρόπου δημιουργίας των δικτύων που χρησιμοποιήθηκαν, και τους λόγους που επιλέχτηκαν. Στο δεύτερο κομμάτι θα δούμε τον τρόπο που ετοιμάστηκε ο αλγόριθμος, από την διαδικασία ομαδοποίησης μέχρι και την αρχικοποίηση του TORA.

5.3.1 Δημιουργία των γράφων

Πριν ξεκινήσουμε με την περιγραφή της διαδικασίας πρέπει να δώσουμε δύο βασικούς ορισμούς:

1. **Διάμετρος δικτύου:** Ως διάμετρος ενός συνδεδεμένου δικτύου ορίζεται ο μέγιστος αριθμός συνδέσμων που απαιτείται για τη μετακίνηση από έναν κόμβο σε έναν άλλο, μέσω μιας ελάχιστης διαδρομής.
2. **Πυκνότητα δικτύου:** Είναι ο μέσος βαθμός κορυφής του δικτύου. Βαθμός κορυφής είναι ο αριθμός των ακμών που καταλήγουν στην κορυφή αυτή. Υπολογίζεται ως

$$\frac{2|E|}{|V|}$$

Δεν πρέπει στα πλαίσια της συγκεκριμένης εργασίας να συγχέουμε την πυκνότητα με τον γραφο-θεωρητικό της ορισμό. Ο αγγλικός ορισμός Graph Density ορίζεται ως:

$$D = \frac{2|E|}{|V|(|V|-1)}$$

Και υποδηλώνει τον αριθμό των ακμών ενός γράφου σε σχέση με τον μέγιστο αριθμό από ακμές που θα μπορούσε δυνητικά να έχει.

Η προσπάθειά μας ήταν εξ' αρχής να δημιουργήσουμε δύο μεγάλα σύνολα από δίκτυα πάνω στα οποία να εφαρμόσουμε τους αλγορίθμους μας. Το πρώτο από αυτά θέλαμε να έχει τις παρακάτω ιδιότητες:

1. **Σταθερή πυκνότητα:** Όσο αυξάνονται οι κόμβοι του δικτύου μας προσπαθήσαμε να διατηρήσουμε τον μέσο βαθμό κορυφής σταθερό
2. **Αναλογικά αυξανόμενη διάμετρο:** Όσο αυξάνονται οι κόμβοι του δικτύου η διάμετρος θέλουμε να είναι αυξανόμενη με μία μορφή τύπου $y = ax$. Ουσιαστικά διάμετρο ανάλογη του αριθμού κόμβων.

Το σύνολο αυτό το ονομάζουμε σταθερής πυκνότητας. Η δεύτερη κατηγορία που την ονομάζουμε σταθερής διαμέτρου έχει αντίστοιχα τα χαρακτηριστικά που ακολουθούν.

1. **Σταθερή διάμετρο:** Όσο αυξάνονται οι κόμβοι του δικτύου μας προσπαθήσαμε να διατηρήσουμε την μέγιστη από τις ελάχιστες αποστάσεις σταθερή
2. **Αναλογικά αυξανόμενη πυκνότητα:** Όσο αυξάνονται οι κόμβοι του δικτύου η πυκνότητα θέλουμε να είναι αυξανόμενη με μία μορφή τύπου $y = ax$. Θέλουμε αντίστοιχα την πυκνότητα ανάλογη με τον αριθμό των κόμβων.

Με τα δύο είδη δικτύων που παρουσιάσαμε παραπάνω, θεωρούμε ότι καλύπτουμε τις περισσότερες περιπτώσεις. Με σταθερά αυξανόμενη πυκνότητα μελετάμε την συμπεριφορά του αλγορίθμου όταν οι κόμβοι έχουν μεγάλο αριθμό από γείτονες καθώς και τι συμβαίνει όταν ο αριθμός αυτός αυξάνεται αλλά οι αποστάσεις μένουν σταθερές για να μην αλλάζει δραματικά η επικοινωνία και η επιβάρυνση της απόδοσης από αυτήν.

Το αντίστροφο κατορθώνουμε στην αντίθετη περίπτωση. Διατηρούμε την πυκνότητα σταθερή, έτσι ώστε να μην υπάρχει ζήτημα στα χρώματα και στην δυσκολία απόδοσής τους, αλλάζουν όμως οι αποστάσεις δραματικά αυξάνοντας συνεχώς την διάμετρο του δικτύου επιβαρύνοντας έτσι την επικοινωνία και αν μελετήσουμε προσεκτικά [U+FFFD]ους αλγορίθμους μας που χρησιμοποιούν δικαστή, θα δούμε ότι η μελέτη της συμπεριφοράς το [U+FFFD]ς σε τέτοια δ [U+FFFD]κτυα είναι ιδιαίτερα σημαντική (μηνύματα σε μεγαλύτερες αποστάσεις απαιτούν πολλούς πόρους για την δρομολόγησή τους).

Ακολουθεί ένα δείγμα από τον κώδικα που υπολογίζει τις μετρικές αυτές για κάθε δίκτυο εισόδου.

```

// -----
int ColoringProcessor::
get_net_diameter(void) throw () {
    int n = this->owner().world().node_count();
    int max = 0;
    int dist[n];
    for (int v = 0; v < n; v++) {
        int d = max_distance(v, dist);
        max = d > max ? d : max;
    }
    return max;
}

// -----

int ColoringProcessor::
max_distance(int v, int dist[]) throw () {
    int n = this->owner().world().node_count();
    for (int i = 0; i < n; i++) dist[i] = n; // set to maximum ↔
    distance
    dist[v] = 0;
    int cnt = 1;
    int depth = 1; // next distance in BFS
    vector<int> distList;
    distList.push_back(v);
    //distList.insert(distList.end(), new Integer(v));
    while (distList.size() > 0) {
        vector<int> nextList;
        for (vector<int>::iterator it = distList.begin(); it != ←
            distList.end(); ++it) {
            shawn::Node* nd = this->owner_w().world_w().←
                find_node_by_id_w(*it);
            vector<int> nbrs;
            for (shawn::EdgeModel::adjacency_iterator it2 = this->←
                owner_w().world_w().begin_adjacent_nodes_w(*nd, ←
                shawn::EdgeModel::CD_BIDI); it2 != this->owner_w().←
                world_w().end_adjacent_nodes_w(*nd); ++it2) {
                nbrs.push_back((*it2).id());
            }
            for (vector<int>::iterator it3 = nbrs.begin(); it3 < ←
                nbrs.end(); ++it3) {
                int u = (*it3);
                if (dist[u] == n) // first time reachable?
                {
                    cnt++;
                    dist[u] = depth;
                    nextList.push_back(u);
                }
            }
        }
        distList = nextList; // next try set of vertices further ↔
        away
        depth++;
    }
    return cnt == n ? depth - 2 : n;
}

```

Υπολογισμός Διαμέτρου.

```
//
float ColoringProcessor::
get_graph_density(void) throw () {
    float density = 0;
    edges = 0;
    int nodes_num = this->owner_w().world_w().node_count();
    for (int i = 0; i < nodes_num; i++) {
        edges--;
        shawn::Node* nd = this->owner_w().world_w().←
            find_node_by_id_w(i);
        for (shawn::EdgeModel::adjacency_iterator it2 = this->←
            owner_w().world_w().begin_adjacent_nodes_w(*nd, shawn←
            ::EdgeModel::CD_BIDI); it2 != this->owner_w().world_w←
            ().end_adjacent_nodes_w(*nd); ++it2) {
            edges++;
        }
    }
    edges = edges / 2;
    density = (float) ((2 * (float) edges) / (float) nodes_num);
    return density;
}
```

Υπολογισμός Πυκνότητας.

Στην εικόνα που ακολουθεί βλέπουμε ένα παράδειγμα του βασικού αρχείου ρυθμίσεων του εξομοιωτή shawn .

```
immediate_delivery=true
iterations=20
prepare_world edge_model=list comm_model=disk_graph \
    trans_model=reliable range=2.0
routing_algorithm=tora
mjcoloring=0
pkjcoloring=0
thcoloring=0
rldcoloring=1
num_judges=0
#rect_world width=5 height=5 count=15 processors=coloring
#load_world file=/home/Master/temp/topologies/findval-topology-10-4.←
    xml processors=coloring
load_world file=/home/Master/shawn/buildfiles/scripts/topologies/←
    topology_e50.xml processors=coloring
#save_world file=/home/Master/shawn/buildfiles/scripts/topologies/←
    topology_zk180.xml snapshot=0
simulation max_iterations=20
vis_create
# Create edges:
vis_create_edges source_regex=.* target_regex=.*
vis_simple_camera
vis_single_snapshot
vis=my_visualization
```

Αρχείο Ρυθμίσεων Shawn .

Οι οχτώ πρώτες γραμμές είναι ρυθμίσεις της εξομοίωσης ως [U+FFFD]ρος τις παραμέτρους που παρουσιάστηκαν στο κεφάλαιο 4 στο κομμάτι που δείξαμε τα μοντέλα του shawn ,αλλά και γενικότερες ρυθμίσεις του coloring αλγορίθμου. Για παράδειγμα βλέπουμε αν θέλουμε να εκτελέσουμε το πρόγραμμα μας με πολλούς αλγορίθμους ή έναν (στην περίπτωση μας εδώ έχουμε mjcoloring = 1) αν θέλουμε παράλληλη εκτέλεση και τέλος πόσοι θα είναι οι δικαστές μας.

Στο τέλος των ρυθμίσεων έχουμε κάποιες εντολές για την απεικόνιση του δικτύου μας μέσω του πακέτου VIS που μας δίνει το shawn καθώς και τον αριθμό

των iterations που είναι μία μετρική καθορισμού του χρόνου που μέγιστου χρόνου που επιτρέπουμε στον αλγόριθμο να εκτελεστεί, και αν δεν έχει τελειώσει μέχρι τότε ολοκληρώνεται η εκτέλεση από τον ίδιο τον εξομοιωτή.

Για το συγκεκριμένο όμως κεφάλαιο μας ενδιαφέρει περισσότερο το μεσαίο κομμάτι του αρχείου ρυθμίσεων. Εδώ βλέπουμε και τις τέσσερις γραμμές σχολιασμένες για να δείξουμε ότι δεν τις λειτουργούμε κάθε φορά όλες. Οι δύο μεσαίες μας φορτώνουν έναν ήδη αποθηκευμένο κόσμο και θέτουν σαν επεξεργαστή τον δηλωμένο στον εξομοιωτή ως coloring . Για περισσότερες πληροφορίες σε σχέση με τους επεξεργαστές προτείνουμε να ανατρέξετε στο κεφάλαιο της εργασίας για το shawn .

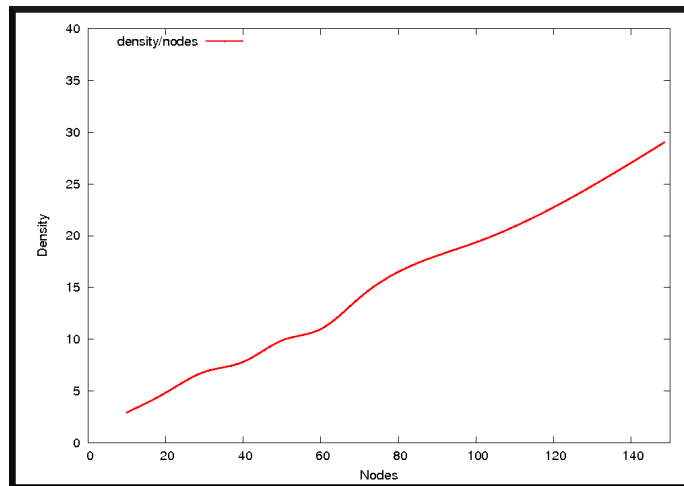
Όταν δεν φορτώνουμε έναν κόσμο που να έχει κατασκευαστεί παλαιότερα σχολιάζουμε τις δύο κεντρικές και ενεργοποιούμε την πρώτη και την τέταρτη γραμμή. Η τέταρτη είναι εδώ για να αποθηκεύσει τον κόσμο στο αρχείο που της δηλώσαμε. Η πρώτη είναι αυτή που θα μας βοηθήσει να κατασκευάσουμε τον κόσμο.

Το σύνολο των γράφων με σταθερή διάμετρο ήταν και το ευκολότερο στην κατασκευή του. Για να δημιουργηθεί πολύ απλά κατασκευάσαμε έναν κόσμο στον οποίο κρατήσαμε σταθερό το ύψος και το πλάτος. Για δική μας ευκολία αργότερα τον κάναμε τετράγωνο. Το shawn στις βασικές του ρυθμίσεις τοποθετεί τους κόμβους τυχαία στον χώρο. Αυτό σημαίνει ότι σε έναν καλό αριθμό από κόμβους σίγουρα ή έστω με μεγάλη βεβαιότητα, θα έχουμε δύο κόμβους με μεγάλη απόσταση.

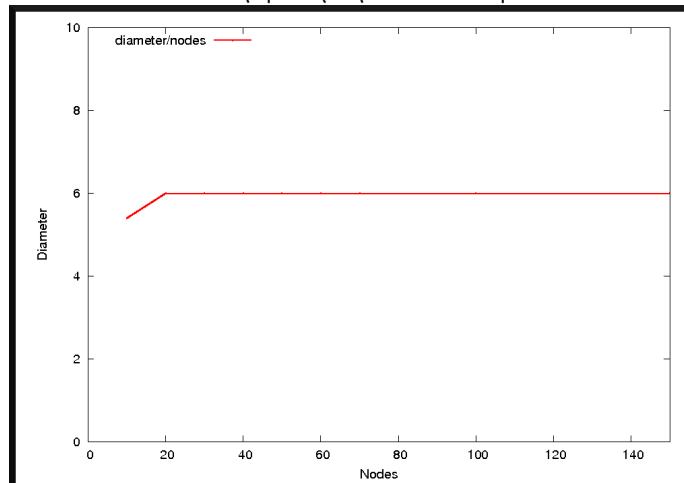
Με βάση την παραπάνω λογική κατασκευάσαμε εύκολα το σύνολο των κόμβων σταθερής διαμέτρου. Στην περίπτωση όμως της σταθερής πυκνότητας έπρεπε να σκεφτούμε λίγο πιο έξυπνα. Για να αυξήσουμε την διάμετρο, ενώ είχαμε αρκετούς κόμβους μέσα στον κόσμο ώστε ο μέσος βαθμός κορυφής να είναι ένας ικανοποιητικός αριθμός, έπρεπε να αυξήσουμε και το μέγεθος του κόσμου. Αυξάναμε λοιπόν τον κόσμο και τους κόμβους με την εξής λογική:

Ξεκινώντας από έναν τετράγωνο κόσμο με μήκος πλευράς α και αριθμό κόμβων n , προκύπτει ένας κόσμος με εμβαδό $E = \alpha^2$ και $\Omega = \frac{n}{\alpha^2}$, όπου Ω οι κόμβοι ανά μονάδα επιφάνειας. Αυξάνοντας συνεπώς την επιφάνεια αυξάνουμε και τους κόμβους με τέτοιο τρόπο ώστε το Ω να παραμένει σταθερό. Εδώ πρέπει να τονίσουμε ότι μια τέτοια τεχνική δεν λειτουργεί για όλα τα δίκτυα. Στα δίκτυα αισθητήρων οι κόμβοι ανά μονάδα επιφάνειας λειτουργούν καλά γιατί οι κόμβοι έχουν μία συγκεκριμένη ακτίνα στην οποία μπορούν να επικοινωνήσουν. Αν δεν ίσχυε αυτό η παραπάνω τεχνική δεν θα ήταν αρκετή για να μας δώσει τα δίκτυα που θέλουμε.

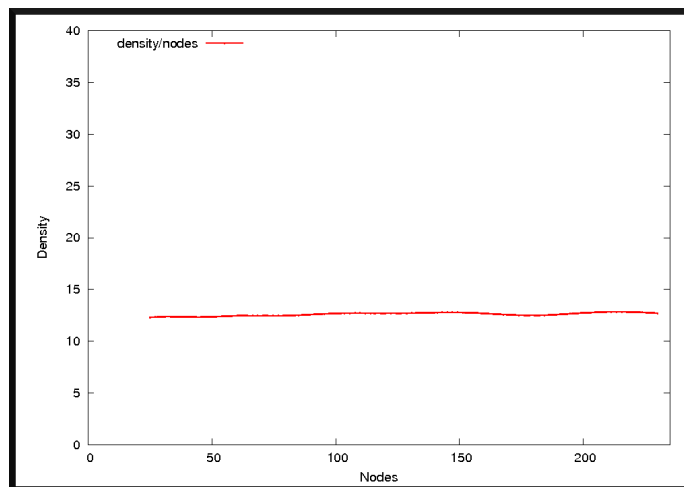
Στις εικόνες που ακολουθούν βλέπετε τα γραφήματα των δύο συνόλων ως προς την πυκνότητα και την διάμετρο που πετύχαμε:



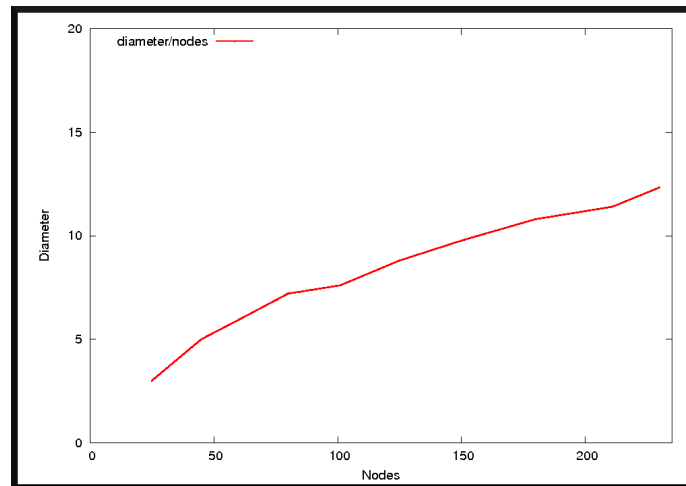
Σταθερή Διάμετρος-Πυκνότητα



Σταθερή Διάμετρος-Διάμετρος



Σταθερή Πυκνότητα-Πυκνότητα



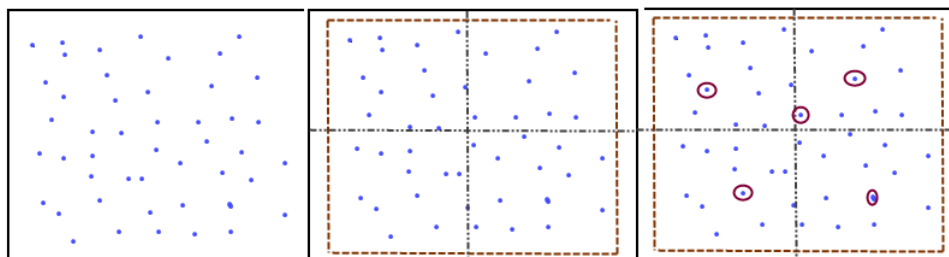
Σταθερή Πυκνότητα-Διάμετρος

5.3.2 Ομαδοποίηση κόμβων και ανάθεση σε δικαστή

Για να ολοκληρωθεί η διαδικασία ρυθμίσεων της εξομοίωσης από το σημείο που έχουμε τα δίκτυα πάνω στα οποία θα πειραματιστούμε το μόνο που απομένει είναι να δώσουμε στο shawn έναν αλγόριθμο για την ομαδοποίηση των κόμβων. Φυσικά τέτοιου είδους ομαδοποίηση θα μπορούσε να γίνει και με έτοιμους αλγόριθμους για κατανεμημένα συστήματα. Τέτοιες προτάσεις έχουν γίνει από τους Ossama Younis, Sonia Fahmy [20], Eshref Januzaj, Hans-Peter Kriegel, Martin Pfeifle [21], Stefano Basagni [22], Hillol Kargupta, Weiyun Huang, Krishnamoorthy Sivakumar, Erik Johnson [23] και πολλούς άλλους.

Η σκέψη για την εφαρμογή ενός από αυτούς τους αλγόριθμους απορρίφθηκε για αρκετούς λόγους. Ο κυριότερος από αυτούς έχει να κάνει με την φύση της επίλυσής μας. Η αυτοματοποιημένη ομαδοποίηση δεν είμαστε σίγουροι πόσες ομάδες θα δημιουργήσει και συνεπώς ποιος θα είναι ο τελικός δικαστής του αλγορίθμου μας. Αυτό δεν είναι θεμιτό, μιας και στην πράξη θέλουμε ο δικαστής να έχει παραπάνω αρμοδιότητες από τους υπόλοιπους κόμβους, πράγμα που οδηγεί στην απαίτηση να έχει παραπάνω υπολογιστικές δυνατότητες, τόσο σε επεξεργαστική ισχύ όσο και σε μνήμη. Επιπλέον αυτό που ζητάμε στην εργασία αυτή είναι να αξιολογήσουμε τον αλγόριθμο χρωματισμού. Δεν κρίθηκε θεμιτό να υλοποιήσουμε μία από τις παραπάνω λύσεις η οποία θα παρέμβαινε σε έναν βαθμό με τα αποτελέσματά μας (π.χ. στην καταμέτρηση απαιτούμενων μηνυμάτων), αλλά και θα καθυστέρουσε την εκτέλεση εξαιτίας της μεγαλύτερης ανάγκης σε μηνύματα δρομολόγησης. Ο αλγόριθμος TORA που χρησιμοποιούμε είναι On Demand και οι δικαστές χρειάζονται μόνο τα μονοπάτια από και προς τους ίδιους. Ένας αλγόριθμος ομαδοποίησης θα απαιτούσε πολύ παραπάνω μονοπάτια. Στην πράξη φυσικά, αν όλοι οι αισθητήρες είναι ικανοί να γίνουν δικαστές ένας τέτοιος αλγόριθμος ομαδοποίησης θα καθιστούσε και τις δικές μας λύσεις πολύ πιο δυναμικές και ικανές να αντεπεξέλθουν σε συνεχείς αλλαγές στα δίκτυα που χρωματίζουν.

Παρά τα όσα είπαμε παραπάνω, απαιτείται ενός είδους ομαδοποίηση για τους κόμβους μας. Θα παρουσιάσουμε για αυτόν το λόγο, τον τρόπο με τον οποίο την πραγματοποιούμε. Στην ακόλουθη εικόνα [5] φαίνονται τα στάδια που περνάει η ομαδοποίηση αυτή.



Εικόνα 5

Στο πρώτο βήμα ουσιαστικά θεσπίζουμε το μέγεθος του κόσμου σαν ένα τετράγωνο. Για να το κάνουμε αυτό παίρνουμε τους τέσσερις κόμβους που έχουν τις μέγιστες και ελάχιστες x, y συντεταγμένες. Από αυτές υπολογίζοντας την διαφορά τους δημιουργούμε το παραλληλόγραμμο που καθιστά τον κόσμο. Πάντα όταν δημιουργούμε τον κόσμο μέσω του εξομοιωτή, φτιάχνουμε ένα τετράγωνο και συνεπώς ο υπολογισμός μας θα είναι πάντα ένα παραλληλόγραμμο με πλευρές σχεδόν ίσες. Χωρίς βλάβη της γενικότητας μπορούμε να πούμε ότι αν νοητά χωρίσουμε το παραλληλόγραμμο σε τέσσερα κομμάτια, όπως φαίνεται και στην εικόνα με τις δύο γραμμές που περνάνε από τα κέντρα των πλευρών πάνω-κάτω και δεξιά-αριστερά, τότε έχουμε τέσσερα ίσα κομμάτια.

Στο σημείο αυτό, παίρνουμε τα κέντρα των τεσσάρων κομματιών. Από όλους τους κόμβους παίρνουμε τις αποστάσεις από τα κέντρα. Όποιος βρίσκεται πιο κοντά στα κέντρα γίνεται αυτόματα για εμάς ο δικαστής. Στην πράξη μπορούμε να τοποθετήσουμε τον δικαστή μας ακριβώς στο κέντρο, στην εξομοίωσή μας όμως κάτι τέτοιο δεν χρειάζεται μιας και όλοι οι κόμβοι έχουν τις ίδιες δυνατότητες.

Αναφέραμε στην προηγούμενη ενότητα ότι οι αλγόριθμοι μας υποστηρίζουν μέχρι πέντε (για την ώρα και για το δεδομένο clustering) δικαστές. Η απόφαση για το ποιός θα είναι δικαστής λαμβάνεται ως εξής.

- **Ένας δικαστής.** Στην περίπτωση αυτή θα πάρουμε τον κόμβο που βρίσκεται πιο κοντά στο κέντρο του δικτύου μας. Η απόφαση αυτή έγινε, για να μην υπάρξουν πολλοί κόμβοι με μεγάλη απόσταση από τον δικαστή, γιατί κάτι τέτοιο θα επιβάρυνε πολύ περισσότερο την δρομολόγησή μας από ότι θα κερδίζαμε από τους λίγους κόμβους που είναι πιο κοντά.
- **Δύο δικαστές.** Υπολογίζουμε και τους τέσσερις δικαστές που βρίσκονται στα κέντρα των περιοχών που ορίσαμε νωρίτερα. Τους ομαδοποιούμε σε κάτω αριστερά-πάνω δεξιά και κάτω δεξιά-πάνω αριστερά. Αποφασίζουμε με βάση την συνολική μέση απόσταση των δύο ομάδων, και όπως είναι φυσικό παίρνουμε αυτήν με την μικρότερη.
- **Τέσσερις δικαστές.** Παίρνουμε και τους τέσσερις κόμβους που βρίσκονται στα κέντρα των περιοχών που ορίσαμε.
- **Τρεις και πέντε.** Όπως και στους δυο ή τέσσερις, μόνο που τώρα δικαστής είναι και αυτός που απέχει λιγότερο από το κέντρο.

Το μόνο που μένει είναι να αποφασίσουμε ποιος κόμβος ανατίθεται σε κάθε δικαστή. Με μία πρώτη ματιά θα λέγαμε πως το πιο λογικό είναι να αναθέσουμε σε κάθε δικαστή τους κόμβους που βρίσκονται πιο κοντά του με την έννοια της

απόστασης. Όμως, όπως είπαμε και στο κεφάλαιο 3.3, η απόσταση δεν είναι το μόνο που καθορίζει την δυσκολία αποστολής μηνυμάτων, ή/και την ταχύτητα με την οποία θα παραδοθούν. Αποφασίσαμε για τον λόγο αυτό να αφήσουμε το δίκτυο από μόνο του να κάνει την ανάθεση αυτή, σε σχέση με την εγγύτητα με την έννοια της ταχύτητας παράδοσης μηνύματος. Ολόκληρη η παραπάνω διαδικασία φαίνεται στον κώδικα που στο τέλος του κεφαλαίου.

Συνοψίζοντας έχουμε φτάσει πλέον σε ένα σημείο στο οποίο υπάρχουν οι αλγόριθμοι έτοιμοι, και η εξομοίωσή μας έχει ρυθμιστεί. Δημιουργήσαμε επιπλέον έναν αυτοματοποιημένο μηχανισμό που παρέχεται από το shawn για να μπορούμε να εκτελέσουμε μαζικά τους αλγορίθμους για όλες τις τοπολογίες που έχουμε κατασκευάσει.

Χρησιμοποιώντας το java αρχείο του shawn ,jshawn-allinone.jar μπορούμε να το τροφοδοτήσουμε με αρχεία ρυθμίσεων παρόμοια με αυτό που είδαμε νωρίτερα. Δημιουργήσαμε για αυτό τον λόγο ένα script το οποίο βρίσει όλα τα αρχεία ρυθμίσεων του αλγορίθμου στον αντίστοιχο φάκελο, και τα εκτελεί ένα προς ένα. Το script αυτό το βλέπουμε εδώ:

```
#!/bin/bash

cd ./jcoloring/Fixed_Diameter/
rm *.out
echo "Evaluating Judged Coloring Algorithm"
for a in *.jshawn
do
    echo "Input File: $a "
    ../../jcoloring "$a"
done

../../fixresults "./out/colors.out.cn" 1
../../gnu.pl ./out/colors.out.cn.final 1
../../fixresults "./out/colors.out.mna.final" 1
../../gnu.pl ./out/colors.out.mna.final.final 2
../../fixresults "./out/colors.out.mnco.final" 1
../../gnu.pl ./out/colors.out.mnco.final.final 3
../../fixresults "./out/colors.out.time" 1
../../gnu.pl ./out/colors.out.time.final 4
../../fixresults "./out/colors.out.diam" 1
../../gnu.pl ./out/colors.out.diam.final 5
../../fixresults "./out/colors.out.dens" 1
../../gnu.pl ./out/colors.out.dens.final 6
```

Η εντολή jcoloring που φαίνεται στις παραπάνω εντολές είναι και αυτή με την σειρά της ένα script σε unix περιβάλλον που χρησιμοποιεί το αρχείο java και καλεί διάφορα εργαλεία και εκτελέσιμα για να μας δημιουργήσει τα αρχεία αποτελεσμάτων που χρειαζόμαστε για την αξιολόγηση των αλγορίθμων.

```
#!/bin/bash
java -jar ../../../../jshawn-allinone.jar -s ../../../../shawn -b ./ $1 |←
grep JC_Colors >> ./out/Colors.out

cat ./out/Colors.out | grep 'JC_Colors-n' | awk '{print $2 "\t" $3 }' |←
>> ./out/colors.out.cn
cat ./out/Colors.out | grep 'JC_Colors-diam' | awk '{print $2 "\t" $3 }' |←
>> ./out/colors.out.diam
cat ./out/Colors.out | grep 'JC_Colors-dens' | awk '{print $2 "\t" $3 }' |←
>> ./out/colors.out.dens
cat ./out/Colors.out | grep 'JC_Colors-time' | awk '{print $2 "\t" $3 }' |←
>> ./out/colors.out.time
cat ./out/Colors.out | grep 'JC_Colors-mna' | awk '{print $2 "\t" $3 }' |←
> ./out/colors.out.mna.tmp
cat ./out/Colors.out | grep 'JC_Colors-mnco' | awk '{print $2 "\t" $3 }' |←
> ./out/colors.out.mnco.tmp
```

```
cat ./out/Colors.out | grep 'JC-Colors-errors' | awk '{print $2 "\t" ←
    $3 }' >> ./out/errors.tmp

./../fixresults "./out/colors.out.mna.tmp" 2
rm "./out/colors.out.mna.tmp";
./../fixresults "./out/colors.out.mnco.tmp" 2
rm "./out/colors.out.mnco.tmp";
rm ./out/Colors.out

mv snapshot.pdf $1.pdf
```

JColoring Script

Για την γραφική απεικόνιση των αποτελεσμάτων από τα αρχεία που δημιουργήσαμε, χρησιμοποιούμε το εργαλείο gnuplot που βρίσκουμε στις περισσότερες διανομές linux. Σε συνδυασμό με την γλώσσα python προκύπτει το αρχείο που ακολουθεί, και ρυθμίζει τον τρόπο με τον οποίο φαίνονται στα γραφήματα τα αποτελέσματα.

```
#!/usr/bin/python

import os
import sys
print "Generating Image File"      # This is an image
print                               # blank line, end of headers
sys.stdout.flush()

gp = os.popen("/usr/bin/gnuplot", 'w')
gp.write('reset\n')
gp.write('set terminal postscript dashed color\n')
gp.write('set output "' + sys.argv[1] + '.eps"\n')
gp.write('unset log xy\n')
gp.write('unset label\n')
gp.write('set key top left\n')
#set xr [0:20]
#set yr [1:10000000]
gp.write('set style line 1 lt 1 lw 3 pointtype 0\n')
gp.write('set style line 2 lt 2 lw 3 pointtype 0\n')
d = int(sys.argv[2])
if d == 1:
    gp.write('set yrange[0:40]\n')
    gp.write('set xrange[0:200]\n')
    gp.write('set xlabel "Nodes"\n')
    gp.write('set ylabel "Colors"\n')
    gp.write('plot "' + sys.argv[1] + '" using 1:2 smooth csplines title \'←
        colors/nodes\' with linepoints linestyle 1')
```

Python Script

```

vector<int> ColoringProcessor::
get_num_judges(int n_judges) throw () {
    vector<int> judges_;
    double world_ux, world_uy, world_lx, world_ly;
    world_ux = this->owner().world().upper_right().x();
    world_uy = this->owner().world().upper_right().y();
    world_lx = this->owner().world().lower_left().x();
    world_ly = this->owner().world().lower_left().y();
    double x_size = world_ux - world_lx;
    double y_size = world_uy - world_ly;
    double dist_from_center = 200;
    int tmp_judge = 0;
    for (int i = 0; i < this->owner().world().node_count(); i++) {
        shawn::Node *nd = this->owner_w().world_w().←
            find_node_by_id_w(i);
        double tmp_dist = get_dist_from_position(nd->real_position←
            ().x(),
        nd->real_position().y(), x_size / 2, y_size / 2);
        if (tmp_dist < dist_from_center) {
            dist_from_center = tmp_dist; tmp_judge = i;
        }
    }
    if ((n_judges % 2) != 0) judges_.push_back(tmp_judge);
    if (n_judges > 1) {
        double dist_from_ul = 200, dist_from_ur = 200;
        double dist_from_ll = 200, dist_from_lr = 200;
        double ul_judge = 0, ur_judge = 0, ll_judge = 0, lr_judge ←
            = 0;
        for (int i = 0; i < this->owner().world().node_count(); i←
            ++){
            shawn::Node *nd = this->owner_w().world_w().←
                find_node_by_id_w(i);
            double tmp_dist = get_dist_from_position(nd->←
                real_position().x(),
        nd->real_position().y(), (x_size / 3), 2 * (y_size / 3));
            if (tmp_dist < dist_from_ul)
                dist_from_ul = tmp_dist; ul_judge = i;
            tmp_dist = get_dist_from_position(nd->real_position().←
                x(),
        nd->real_position().y(), 2 * (x_size / 3), 2 * (y_size / 3));
            if (tmp_dist < dist_from_ur)
                dist_from_ur = tmp_dist; ur_judge = i;
            tmp_dist = get_dist_from_position(nd->real_position().←
                x(),
        nd->real_position().y(), x_size / 3, y_size / 3);
            if (tmp_dist < dist_from_ll)
                dist_from_ll = tmp_dist; ll_judge = i;
            tmp_dist = get_dist_from_position(nd->real_position().←
                x(),
        nd->real_position().y(), 2 * (x_size / 3), y_size / 3);
            if (tmp_dist < dist_from_lr)
                dist_from_lr = tmp_dist; lr_judge = i;
        }
        if ((n_judges == 4) || (n_judges == 5)) {
            judges_.push_back(ul_judge); judges_.push_back(ur_judge←
                );
            judges_.push_back(ll_judge); judges_.push_back(lr_judge←
                );
        } else if ((n_judges == 2) || (n_judges == 3)) {
            double sum1 = (dist_from_ur + dist_from_ll) / 2;
            double sum2 = (dist_from_ul + dist_from_lr) / 2;
            if (sum1 < sum2) {
                judges_.push_back(ur_judge); judges_.push_back(←
                    ll_judge);
            } else {
                judges_.push_back(ul_judge); judges_.push_back(←
                    lr_judge);
            }
        }
    }
    return judges_;
}

```

Αλγόριθμος Ομαδοποίησης

Κεφάλαιο 6

Αξιολόγηση και Συμπεράσματα

6.1 Εισαγωγή

Στο τελευταίο κεφάλαιο θα παρουσιάσουμε τα αποτελέσματα των διαφόρων λύσεων που υλοποιήθηκαν για τον χρωματισμό γράφων. Για την κατανόησή τους θα δούμε αρχικά τις μετρικές που θεσπίσαμε για να μετρήσουμε την απόδοση ως προς τον χρωματικό αριθμό και την πολυπλοκότητά τους. Θα συνεχίσουμε με τις γραφικές παραστάσεις που προέκυψαν από τα τελικά αποτελέσματα και τον σχολιασμό τους, για να καταλήξουμε στις καλύτερες από τις λύσεις αυτές ανάλογα με τις απαιτήσεις που έχουμε σε απόδοση συστημάτων ή χρωματισμών.

Στο δεύτερο σκέλος του κεφαλαίου θα δούμε αναλυτικότερα τις εργασίες που επιθυμούμε να κάνουμε για να βελτιώσουμε παραπάνω τις λύσεις μας, αλλά και την περαιτέρω εργασία πάνω στις ήδη υπάρχοντες υλοποιήσεις. Τέτοιες είναι οι εξομοιώσεις σε ακόμα περισσότερα δίκτυα για πιο ολοκληρωμένη άποψη ως προς τα αποτελέσματα, αλλά και η δημιουργία άλλων ειδών δικτύων πέραν των δύο που είδαμε στο προηγούμενο κεφάλαιο κ.α.

6.2 Μετρικές

6.2.1 Χρωματικός Αριθμός

Μιλήσαμε στο πρώτο κεφάλαιο εκτενώς για το πρόβλημα του χρωματισμού κορυφών γράφων vertex coloring αλλά και για τις προσεγγιστικές λύσεις στα προβλήματα αυτά. Από καθαρά θεωρητική άποψη ένας αλγόριθμος που σε πολυωνυμικό χρόνο μπορεί να λύσει, αναλύεται ως προς την απόδοση του για τον χρωματικό αριθμό. Υπενθυμίζουμε εδώ ότι:

Χρωματικός αριθμός $\chi(G)$ ενός γράφου $G(V,E)$, είναι ο μικρότερος αριθμός από χρώματα που χρειαζόμαστε για να χρωματίσουμε έναν γράφο έτσι ώστε να μην υπάρχουν γειτονικές κορυφές με το ίδιο χρώμα.

Είναι εμφανές λοιπόν ότι και στην πρακτική εφαρμογή των υλοποιήσεών μας, οφείλουμε να κρίνουμε τα αποτελέσματά μας σε σχέση με τον αριθμό αυτό. Φυσικά δεν είναι δυνατόν να υπολογιστεί ο ακριβής αριθμός, όμως ευτυχώς έχουμε την δυνατότητα να συγκρίνουμε με τον αλγόριθμο των Σπυράκη-Παναγοπούλου που μας δίνει ακριβή άνω και κάτω όρια ως προς την απόδοση της σε σύγκριση με την μη ντετερμινιστική λύση.

Ο χρωματικός αριθμός ως μετρική από μόνη της δεν αρκεί για τα ασύρματα δίκτυα αισθητήρων. Εδώ αν και υπάρχει επιτακτική ανάγκη για καλά χρωματισμένα δίκτυα δεν μπορούμε να αγνοήσουμε και την μικρή υπολογιστική ικανότητα και τις απαιτήσεις σε ενέργεια. Αν το δίκτυο μας δεν έχει την ικανότητα να πραγματοποιήσει εργασίες δεν υπάρχει λόγος να χρησιμοποιήσουμε χρωματισμό για να τις χρονο-προγραμματίσουμε.

6.2.2 Αριθμός Μηνυμάτων

Για τους λόγους που αναφέρθηκαν στην προηγούμενη παράγραφο θεσπίσαμε τον αριθμό των μηνυμάτων ως την πολυπλοκότητα των αλγορίθμων. Με μια πρώτη ματιά κάτι τέτοιο δεν φαίνεται επαρκές. Μεγάλο ζήτημα για τα ασύρματα δίκτυα αισθητήρων είναι και η δυνατότητα σε μνήμη και επεξεργαστική ισχύ. Αυτό πιθανόν να δημιουργήσει πρόβλημα στους αλγορίθμους με περιορισμένη γνώση, όπου ζητάμε από όλους τους κόμβους να πραγματοποιούν πράξεις, το μέγεθος των οποίων καθορίζεται από την εκάστοτε τοπολογία, και ειδικότερα από τον μέσο βαθμό του κόμβου υπό εξέταση. Στην περίπτωση όμως του απλού χρωματισμού με τις υλοποιήσεις που βασίστηκαν στην θεωρία παιγνίων, κάτι τέτοιο δεν μας απασχολεί, μιας και οι εργασίες των απλών κόμβων αποτελούνται μόνο από αποστολή και παραλαβή μηνυμάτων, και πολύ απλών πράξεων πάνω σε αυτά. Η επιβάρυνση του συνολικού χρόνου από αυτά θεωρήθηκε αμελητέα σε σχέση με αυτή των μηνυμάτων και μπορούμε με ασφάλεια να την αγνοήσουμε.

Χωρίζουμε τα μηνύματα που μελετάμε σε δύο κατηγορίες:

1. **Συνολικά μηνύματα**
2. **Μηνύματα αλγορίθμου**

Στην πρώτη κατηγορία, μετράμε όλα τα μηνύματα που χρειάστηκε να σταλούν από τους κόμβους στους γείτονές τους. Σε αυτά συμπεριλαμβάνουμε όλα τα

μηνύματα αρχικοποίησης του αλγορίθμου δρομολόγησης, τα μηνύματα αρχικοποίησης και γνωριμιών των δικών μας αλλά και όλα όσα χρειάζεται ο TORA για να διατηρεί και να ανανεώνει τους πίνακες δρομολόγησης.

Αντίθετα στην δεύτερη κατηγορία λειτουργούμε πιο αφαιρετικά. Θεωρούμε ότι η δρομολόγηση έχει γίνει νωρίτερα και είναι έτοιμες όλες οι διαδρομές. Στην περίπτωση αυτή μετράμε μόνο τα μηνύματα που δεδομένων που προωθούνται μεταξύ των ενδιάμεσων κόμβων.

Ο παραπάνω διαχωρισμός γίνεται για να εντοπίσουμε αφενός οποιεσδήποτε ιδιομορφίες στα αποτελέσματά μας οφείλονται στον αλγόριθμο δρομολόγησης και αφετέρου για να μπορέσουμε να κατανοήσουμε σε έναν καλό βαθμό την τάξη μεγέθους μηνυμάτων που χρειάζεται η κάθε λύση. Ακριβή αποτελέσματα είναι αδύνατον να πάρουμε μιας και για να συμβεί αυτό θα πρέπει να βρεθεί ιδανικός αλγόριθμος δρομολόγησης, πράγμα που από όσα γνωρίζουμε την παρούσα χρονική στιγμή, δεν υπάρχει.

6.2.3 Χρόνος εκτέλεσης

Ο εξομοιωτής shawn μας δίνει την δυνατότητα αξιολόγησης αλγορίθμων σε σχέση με τον χρόνο, με την μορφή των γύρων. Οι επεξεργαστές προγραμματίζονται με τρόπο τέτοιο ώστε σε κάθε γύρω να μπορούν να κάνουν συγκεκριμένα πράγματα, όπως για παράδειγμα να επεξεργαστούν και να στείλουν ένα μήνυμα.

Δυστυχώς στην περίπτωση που χρησιμοποιούμε την βιβλιοθήκη wiselib για να κατασκευάσουμε αλγορίθμους δεν μπορούμε να μελετήσουμε τον χρόνο με τον ίδιο τρόπο. Όταν ξεκινάει η εκτέλεση του προγράμματος, ο επεξεργαστής συνδέεται κατά κάποιο τρόπο με την βιβλιοθήκη, δημιουργώντας ένα αντικείμενο της κλάσης του αλγορίθμου που θέλει να υλοποιήσει. Με αυτόν τον τρόπο δεν ορίζουμε συγκεκριμένο σύνολο από εργασίες ανά γύρο. Η δυνατότητα αυτή υπάρχει, όμως απαιτεί μεγάλο μέρος του κώδικα που χρειάζονται οι αλγόριθμοι, όπως είναι ο συγχρονισμός τους να μεταφερθεί στο shawn με κόστος την αναγκαιότητα να γίνει η ίδια δουλειά κάθε φορά που θέλουμε να εκτελέσουμε τον αλγόριθμο σε διαφορετική πλατφόρμα.

Να σημειώσουμε σε αυτό το σημείο, ότι η παραλληλία στις διεργασίες παραμένει. Η διαφορά είναι ότι το shawn αφιερώνει κομμάτια χρόνου σε κάθε κόμβο που μας είναι άγνωστα αν και λαμβάνει υπόψη του τις καθυστερήσεις που επιβάλουμε με χρονόμετρα στον κώδικα της wiselib.

Ανεξάρτητα όμως από την αδυναμία του εξομοιωτή μετρήσει την απόδοση σε γύρους, θεσπίσαμε σαν μετρική για του σειριακούς αλγορίθμους και τον χρόνο εκτέλεσης της εξομοίωσης. Στην σειριακή περίπτωση ελπίζουμε να μας δώσει μια καλή αναλογική προσέγγιση σε σχέση με τον πραγματικό χρόνο που θα χρειάζονταν οι αλγόριθμοι στην πράξη.

6.3 Συγκρίσεις αλγορίθμων

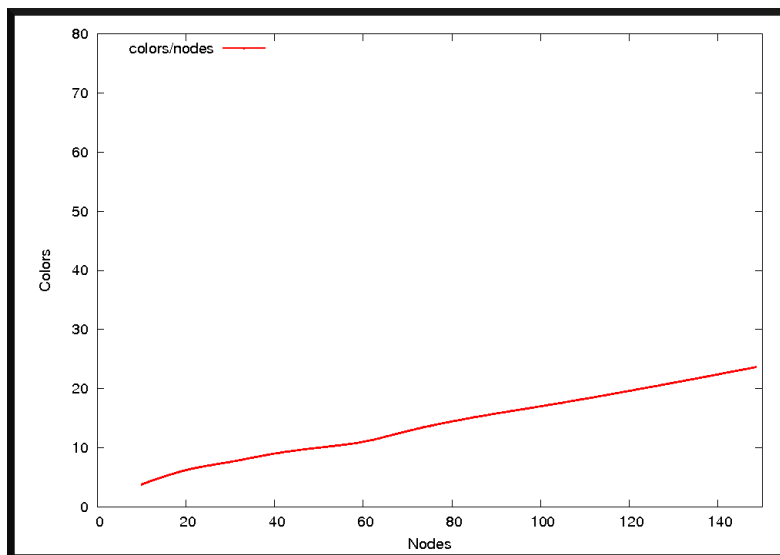
Στο τμήμα αυτό δείχνουμε τα αποτελέσματά μας με τον εξής τρόπο. Για κάθε έναν αλγόριθμο ξεχωριστά έχουμε από ένα υποκεφάλαιο στο οποίο φαίνονται με την σειρά, πρώτα τα αποτελέσματα ως προς τα δίκτυα σταθερής διαμέτρου και μετά τα σταθερής πυκνότητας. Για κάθε μία γραφική παράσταση υπάρχουν σχόλια και όπου χρειάζεται κάποια συμπεράσματα. Στο τέλος υπάρχουν γενικά συμπεράσματα ως προς το σύνολο των αποτελεσμάτων.

6.3.1 Αλγόριθμος με έναν δικαστή

Ο αλγόριθμος με έναν δικαστή θα αναφέρεται και ως JColoring (Judged Coloring) ή πιο απλά JC.

Αποτελέσματα σταθερής διαμέτρου

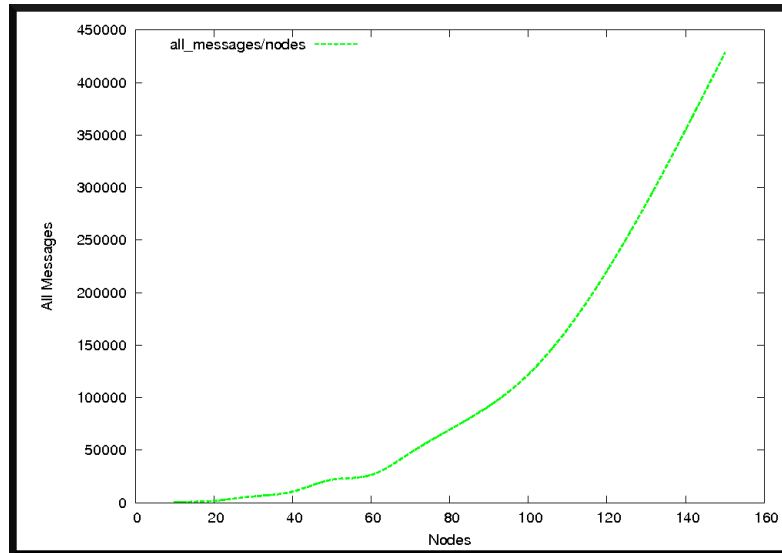
Χρωματικός Αριθμός



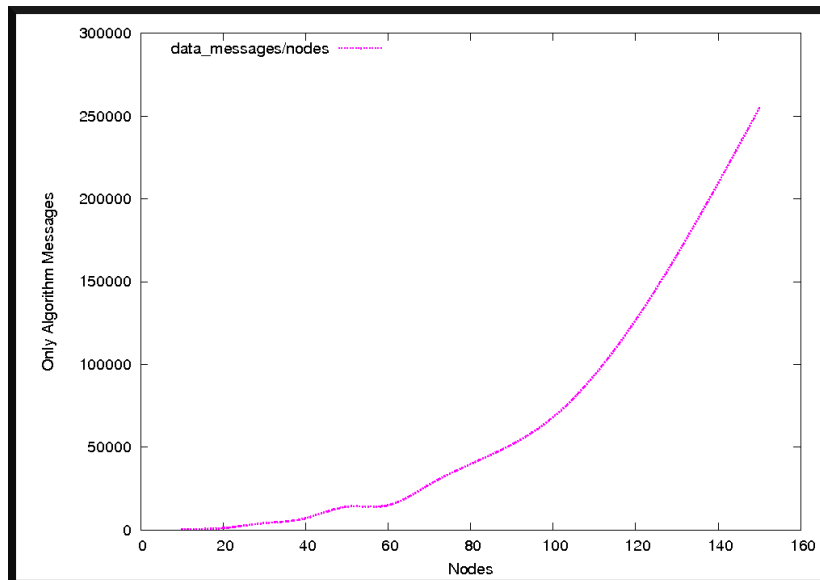
Σταθερή Διάμετρος-Χρώματα

Στην περίπτωση της σταθερής διαμέτρου παρατηρούμε μία συνεχόμενη αναλογική αύξηση των χρωμάτων. Αυτό ήταν κάτι που το περιμέναμε αφού αυξανόμενη πυκνότητα σημαίνει περισσότεροι γείτονες ανά κόμβο. Είδαμε και στις γραφικές ότι στα πειράματα αυτά η πυκνότητα είναι αυξανόμενη σε σχέση με τον αριθμό των κόμβων. Αυτό που έχει μεγάλη σημασία εδώ όμως είναι ότι τα χρώματα αυξάνονται με πολύ μικρότερο ρυθμό από ότι αυξάνεται η πυκνότητα και αυτό φαίνεται από την κλίση της ευθείας. Ακόμα βλέπουμε ότι το ποσοστό $\frac{x(G)}{|N|}$ μικραίνει όσο αυξάνεται η είσοδος.

Αριθμός Μηνυμάτων



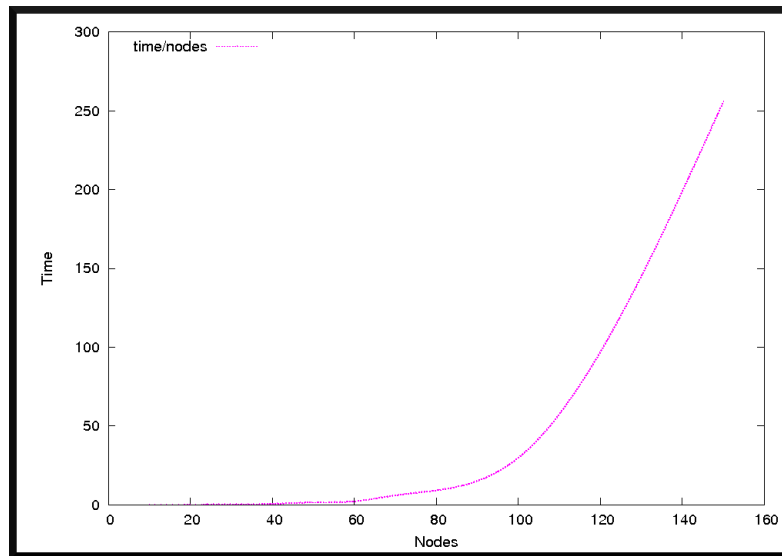
Σταθερή Διάμετρος-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Διάμετρος-Αριθμός Μηνυμάτων Αλγορίθμου

Τόσο στα συνολικά μηνύματα όσο και στα μηνύματα του αλγορίθμου, παρατηρούμε ότι η μορφή της καμπύλης δεν μεταβάλλεται. Η αύξηση όπως βλέπουμε τείνει να γίνει εκθετική όσο αυξάνεται ο αριθμός των κορυφών, πράγμα που καθιστά απαγορευτικές τις δοκιμές με περισσότερες κορυφές. Η καμπύλη είναι όμως αρκετά ομαλή ώστε να έχουμε μία πολύ καλή προσέγγιση ως προς το ποια θα είναι τα μηνύματα για οποιαδήποτε είσοδο.

Χρόνος

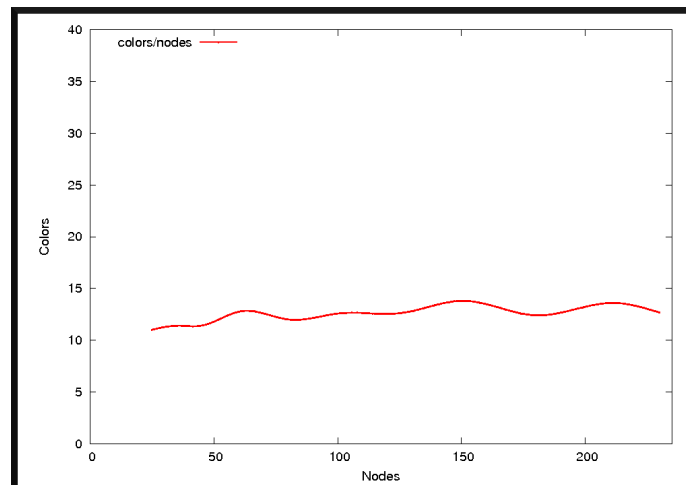


Σταθερή Διάμετρος-Χρόνος

Ο χρόνος που πήρε η εκτέλεση του αλγορίθμου βλέπουμε ότι ακολουθεί την μορφή που έχουν και τα μηνύματα. Αυτό δεν μας εκπλήσσει αφού όπως είπαμε και στην εισαγωγή τα μηνύματα και η επιλογή αποστολής τους με δρομολόγηση είναι η πιο χρονοβόρα διαδικασία που πραγματοποιούν οι αλγόριθμοι μας.

Αποτελέσματα σταθερής πυκνότητας

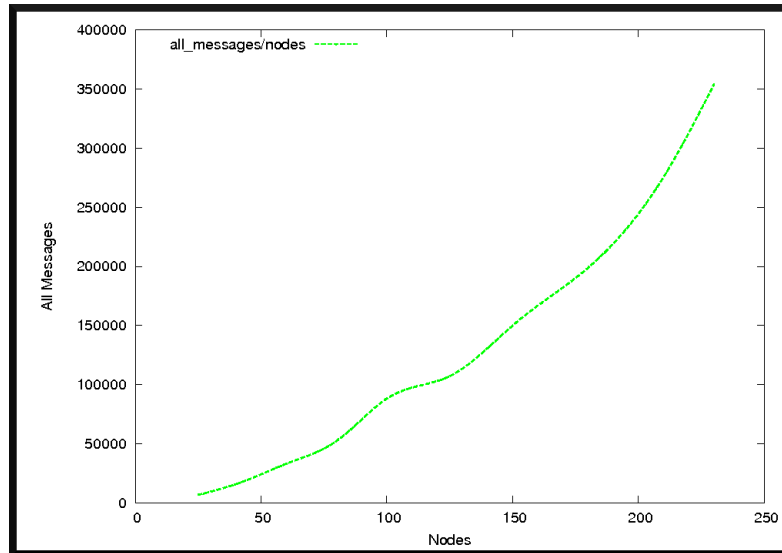
Χρωματικός Αριθμός



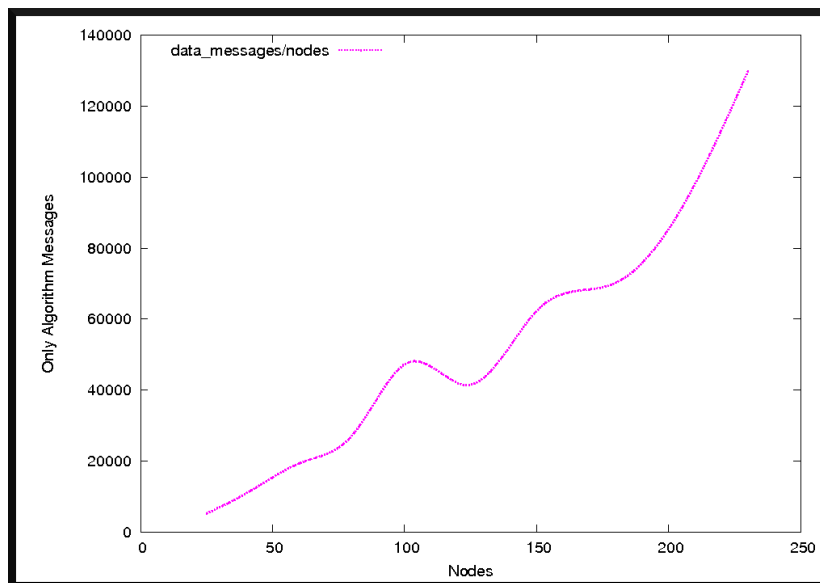
Σταθερή Πυκνότητα-Χρώματα

Στην περίπτωση της σταθερής πυκνότητας τα χρώματα παραμένουν σταθερά με πολύ μικρή απόκλιση από το 12. Η ιδιότητα αυτή αποδίδεται στον σταθερό μέσο βαθμό κορυφής ο οποίος είναι και αυτός ίσος ακριβώς με τον χρωματικό αριθμό. Το αποτέλεσμα αυτό είναι πολύ καλό για τον αλγόριθμο αφού φαίνεται ότι η ποιότητα του αλγορίθμου που δεν χειροτερεύει όσο μεγαλώνουν τα δίκτυα.

Αριθμός Μηνυμάτων



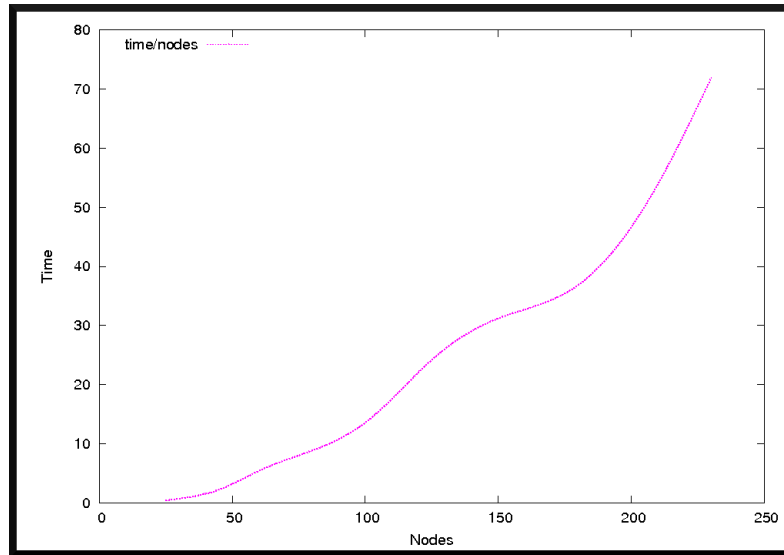
Σταθερή Πυκνότητα-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Πυκνότητα-Αριθμός Μηνυμάτων Αλγορίθμου

Εδώ παρατηρούμε μια διαφοροποίηση ως προς τις δύο κατηγορίες μηνυμάτων. Τα μηνύματα δρομολόγησης είναι πολύ περισσότερα από αυτά του αλγορίθμου. Η πολύ μεγάλη αύξηση της διαμέτρου δυσκολεύει αντίστοιχα και την δρομολόγηση, πράγμα που ελπίζουμε να επιλύσουμε με τις κατανεμημένες μορφές των αλγορίθμων. Είναι όμως θετικό ότι εδώ η αύξηση γίνεται σχεδόν με αναλογικό τρόπο σε σχέση με την είσοδο.

Χρόνος



Σταθερή Πυκνότητα-Χρόνος

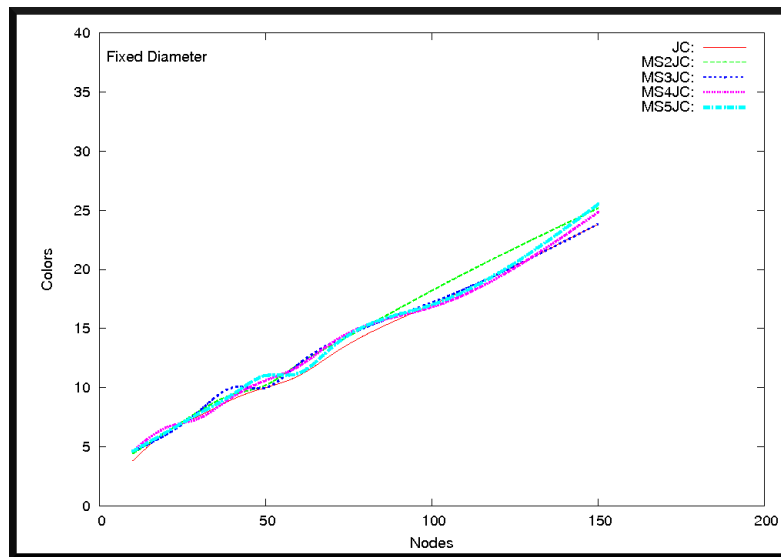
Ο χρόνος της εκτέλεσης ακολουθεί την συμπεριφορά των συνολικών μηνυμάτων πράγμα αναμενόμενο.

6.3.2 Σειριακός αλγόριθμος με πολλούς δικαστές

Ο σειριακός αλγόριθμος με έναν δικαστή θα αναφέρεται και ως MSJColoring (Multi Server Judged Coloring) ή πιο απλά MS-N-JC όπου το N ο αριθμός των δικαστών.

Αποτελέσματα σταθερής διαμέτρου

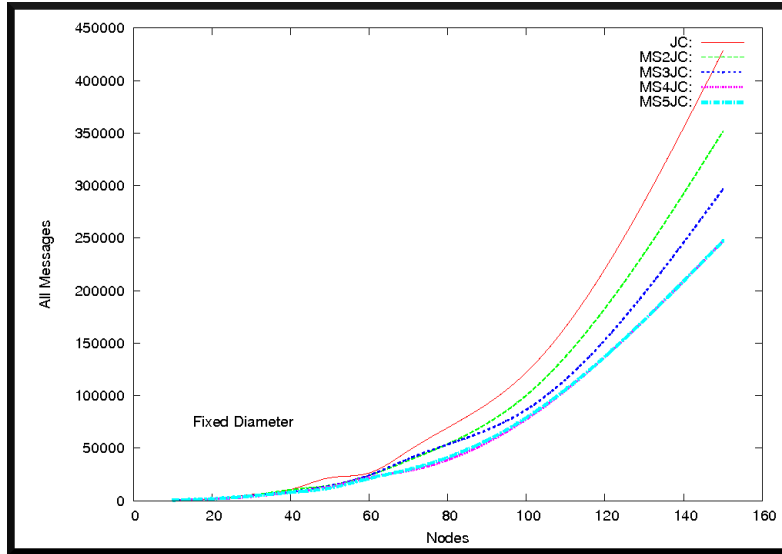
Χρωματικός Αριθμός



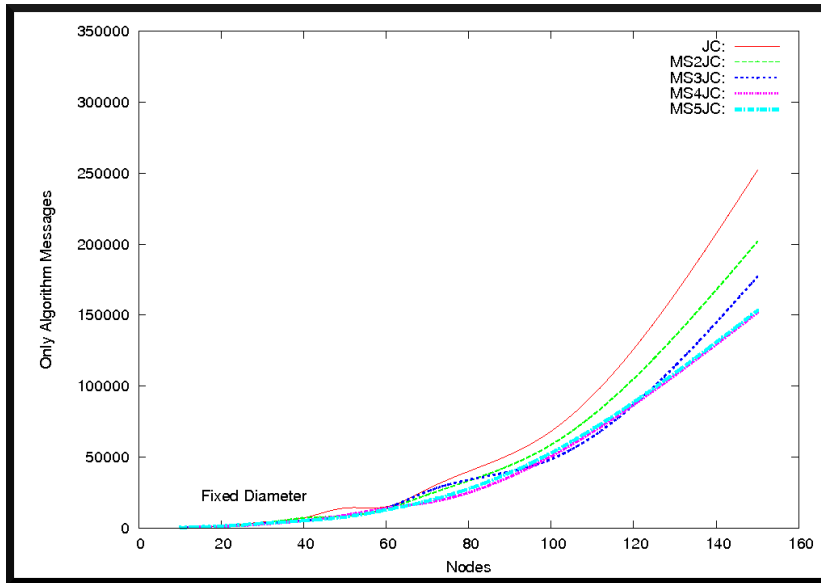
Σταθερή Διάμετρος-Χρώματα

Για τον αλγόριθμο αυτό το πιο σημαντικό είναι να αξιολογήσουμε τα αποτελέσματά μας με βάση τα αντίστοιχα που πήραμε από τον ένα μόνο δικαστή. Κρατώντας σταθερή την διάμετρο, ως προς τα χρώματα βλέπουμε μία μικρή αν και σταθερή διαφορά από τα αντίστοιχα του αλγορίθμου με έναν μόνο δικαστή. Η διαφορά μένει στα 2 με τέσσερα το πολύ χρώματα όσο και αν αυξάνεται η είσοδος αν και προς το τέλος των μετρήσεων βλέπουμε μία τάση να αυξηθούν λίγο περισσότερο αν και με χαμηλούς ρυθμούς.

Αριθμός Μηνυμάτων



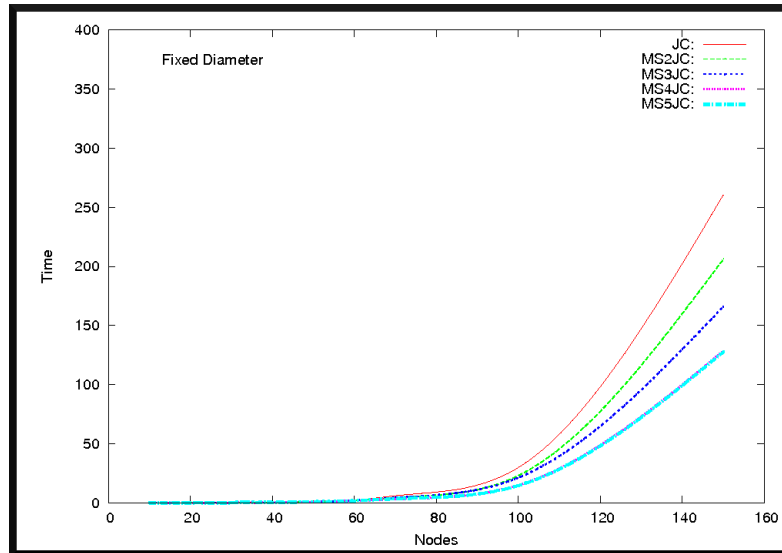
Σταθερή Διάμετρος-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Διάμετρος-Αριθμός Μηνυμάτων Αλγορίθμου

Ως προς τα μηνύματα η ποσοστιαία διαφορά είναι μεγάλη. Στην περίπτωση των πέντε δικαστών βλέπουμε σχεδόν τα μισά μηνύματα να είναι απαραίτητα, τόσο ως προς τα συνολικά (μιας και η δρομολόγηση εδώ είναι αισθητά πιο εύκολη), αλλά και στου αλγορίθμου, αφού ο κάθε δικαστής δεν χρειάζεται να δοκιμάσει χρώματα από τεράστιες παλέτες σε όλους τους κόμβους όπως πρέπει να κάνει με έναν δικαστή που στα αρχικά βήματα μπορεί να χρειαστεί να στείλει και ολόκληρη την παλέτα.

Χρόνος

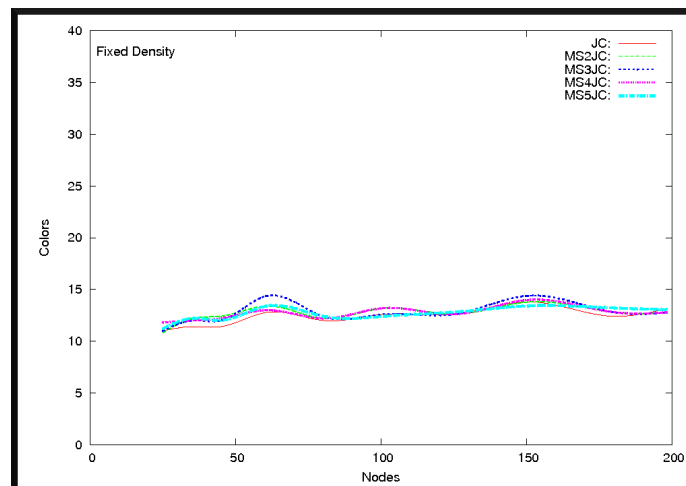


Σταθερή Διάμετρος-Χρόνος

Ως προς τον χρόνο εκτέλεσης ο MS-N-JC υπερισχύει αισθητά. Έχουμε χρόνους εκτέλεσης σχεδόν τρεις φορές πιο γρήγορους.

Αποτελέσματα σταθερής πυκνότητας

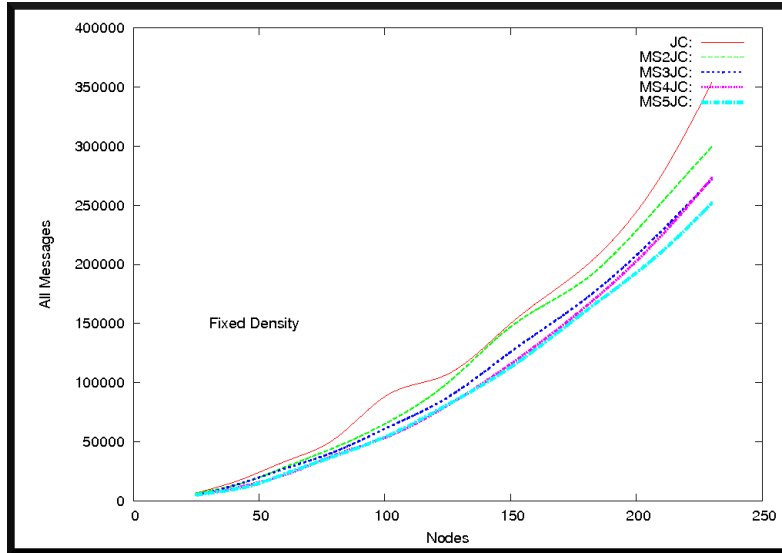
Χρωματικός Αριθμός



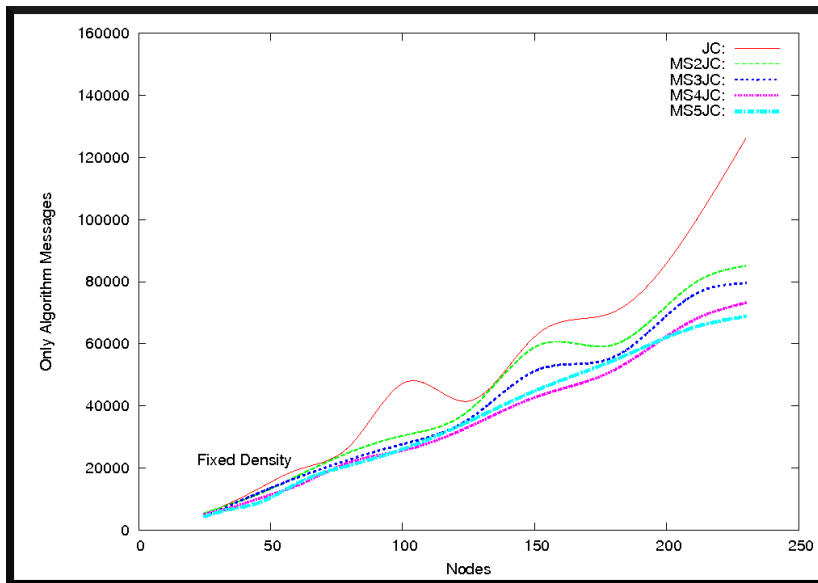
Σταθερή Πυκνότητα-Χρώματα

Στην σταθερή πυκνότητα, οι διαφορές στον χρωματικό αριθμό είναι σχεδόν αμελητέες. Έχουμε περιπτώσεις που οι πολλαπλοί δικαστές δίνουν καλύτερα αποτελέσματα ακόμη και από τον ένα. Πιο σύνηθες βέβαια είναι όπως βλέπουμε να έχουμε 1 με 2 χρώματα παραπάνω.

Αριθμός Μηνυμάτων



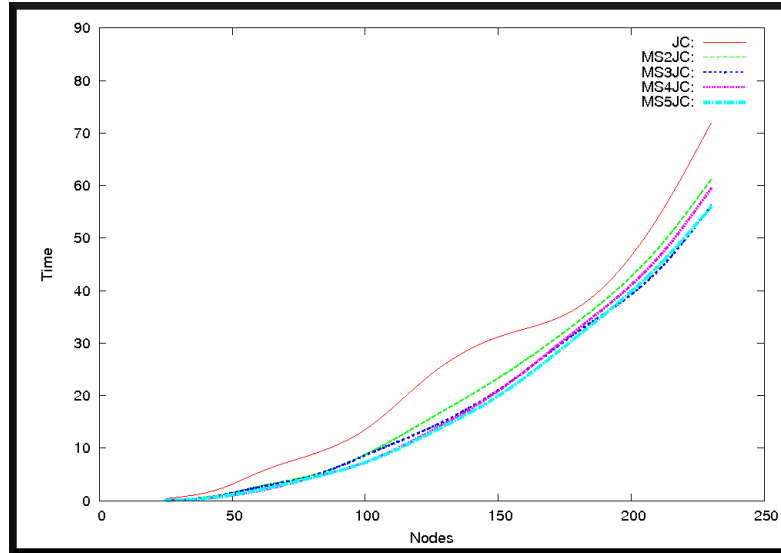
Σταθερή Πυκνότητα-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Πυκνότητα-Αριθμός Μηνυμάτων Αλγορίθμου

Το κέρδος στα μηνύματα δεν είναι όμως το ίδιο καλό με τα πειράματα σταθερής διαμέτρου. Η διαφορά όμως είναι αισθητή και είναι της τάξης των 200.000 μηνυμάτων για μεγάλα δίκτυα. Εδώ να τονίσουμε ότι βλέπουμε για πρώτη φορά διαφορές στα μηνύματα αλγορίθμου και τα συνολικά. Ενώ τα συνολικά αυξάνονται με κανονικό ρυθμό, στα μηνύματα αλγορίθμου παρατηρούνται διαφοροποιήσεις οι οποίες όμως αποσβένονται όσο αυξάνονται οι δικαστές.

Χρόνος



Σταθερή Πυκνότητα-Χρόνος

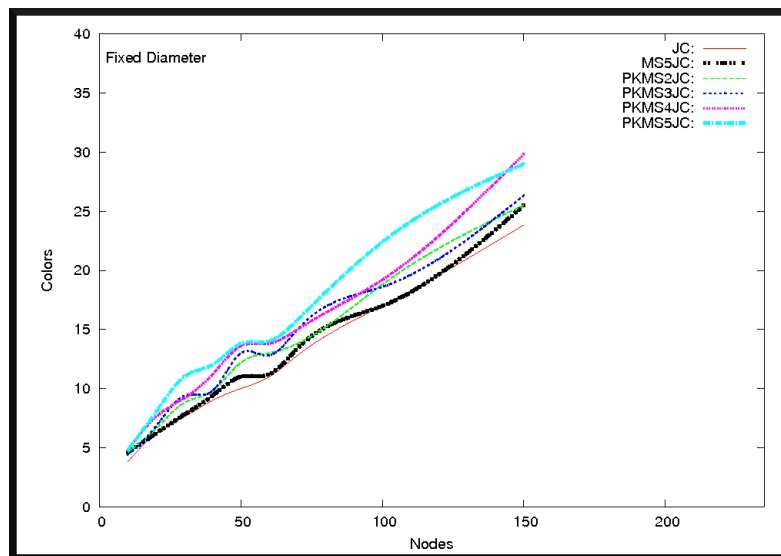
Τα κέρδη στον χρόνο εκτέλεσης είναι αντίστοιχα με τα κέρδη στα μηνύματα.

6.3.3 Παράλληλος αλγόριθμος πολλαπλών δικαστών

Ο Παράλληλος αλγόριθμος πολλαπλών δικαστών θα αναφέρεται και ως PKJColoring (Partial knowledge Judged Coloring) ή πιο απλά PKJC.

Αποτελέσματα σταθερής διαμέτρου

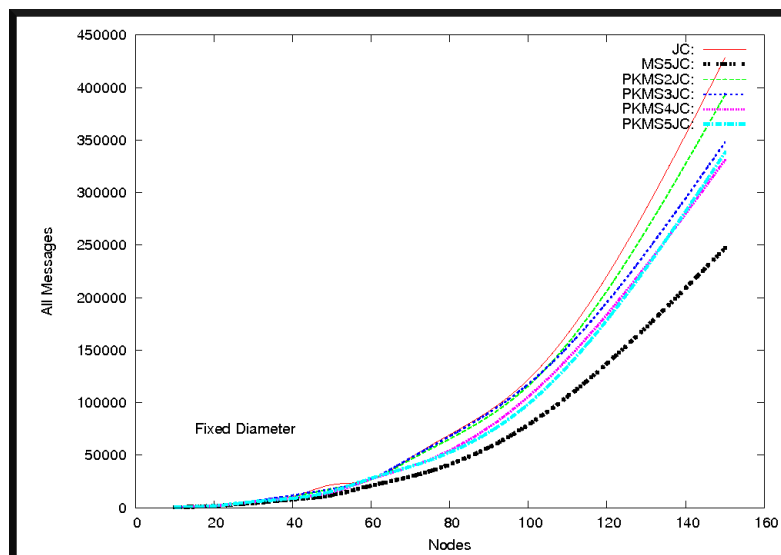
Χρωματικός Αριθμός



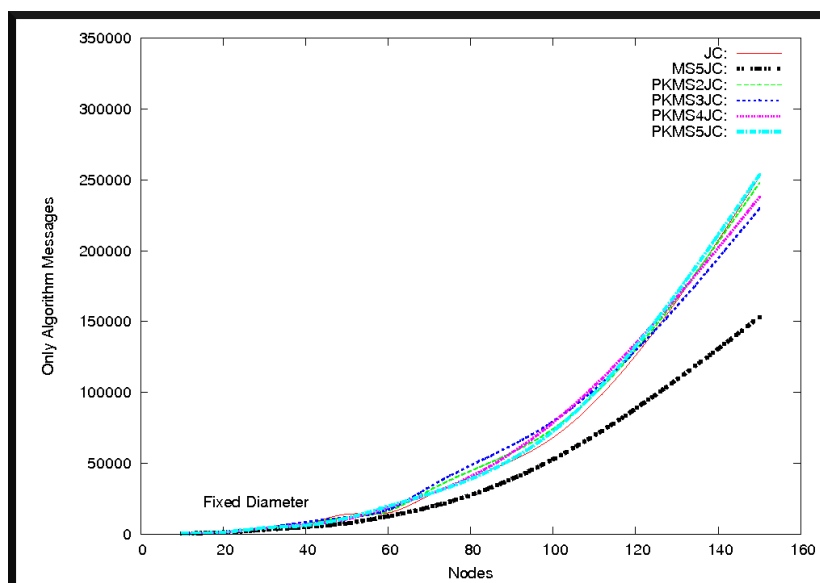
Σταθερή Διάμετρος-Χρώματα

Σε αντίθεση με την σειριακή εκτέλεση πολλαπλών δικαστών, όπου τα χρώματα δεν έχουν σχεδόν καθόλου απόκλιση από τον απλό αλγόριθμο, στους πολλαπλούς δικαστές τα πράγματα είναι λίγο διαφορετικά. Έχουμε περιπτώσεις με αύξηση μέχρι και πέντε χρώματα για μεγάλους αριθμούς κόμβων. Χαρακτηριστικό παράδειγμα είναι η περίπτωση με τέσσερις δικαστές που μας δίνει κατά μέσο όρο στους εκατόν πενήντα κόμβους κατά μέσο όρο 4,5 χρώματα παραπάνω. Η αύξηση αυτή ήταν αναμενόμενη. Στην φάση της διόρθωσης των προβλημάτων, ήταν σύνηθες φαινόμενο να δίνονται χρώματα έξω από την αρχική παλέτα. Τα χρώματα αυτά αναγκαστικά προστίθενται στο σύνολο στην τελική καταμέτρηση. Το θετικό εδώ είναι ότι η κλίσεις των ευθειών παραμένουν ίδιες και τα χρώματα δεν δείχνουν να χειροτερεύουν περισσότερο όσο αυξάνεται η είσοδος.

Αριθμός Μηνυμάτων



Σταθερή Διάμετρος-Αριθμός Συνολικών Μηνυμάτων



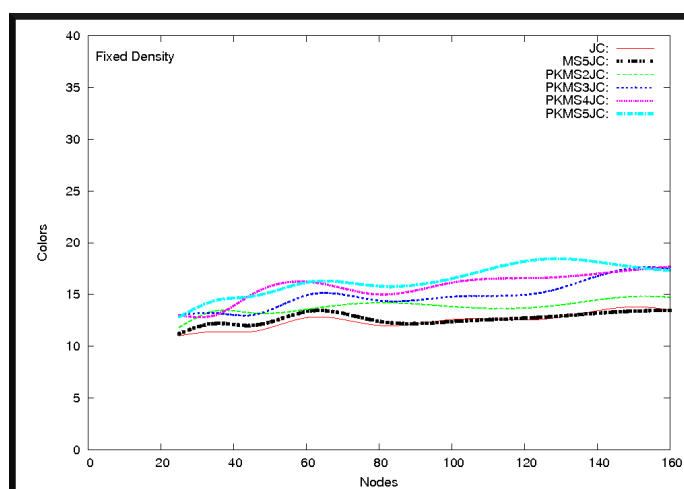
Σταθερή Διάμετρος-Αριθμός Μηνυμάτων Αλγορίθμου

Ακόμα και με την φάση διόρθωσης των κόμβων με σφάλματα, παραμένουν λιγότερα τα συνολικά μηνύματα. Για μέχρι τέσσερις δικαστές τα μηνύματα του αλγορίθμου είναι λιγότερα από τον πρώτο αλγόριθμο, αν και η διαφορά είναι μικρή. Σε σχέση με τον σειριακό πολλαπλών δικαστών βλέπουμε όμως αρκετά μεγάλη διαφορά.

Χρόνος Εδώ ο χρόνος εκτέλεσης δεν έχει μεγάλη σημασία. Μόνο στην πράξη μπορούμε να δούμε τα αποτελέσματα μιας και δεν εξομοιώνεται η παραλληλία τόσο καλά ώστε να δούμε τα αποτελέσματά της από τον χρόνο.

Αποτελέσματα σταθερής πυκνότητας

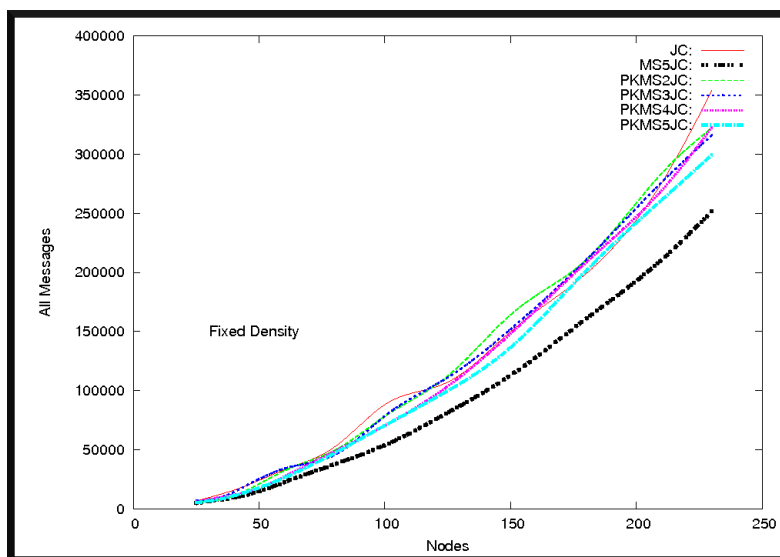
Χρωματικός Αριθμός



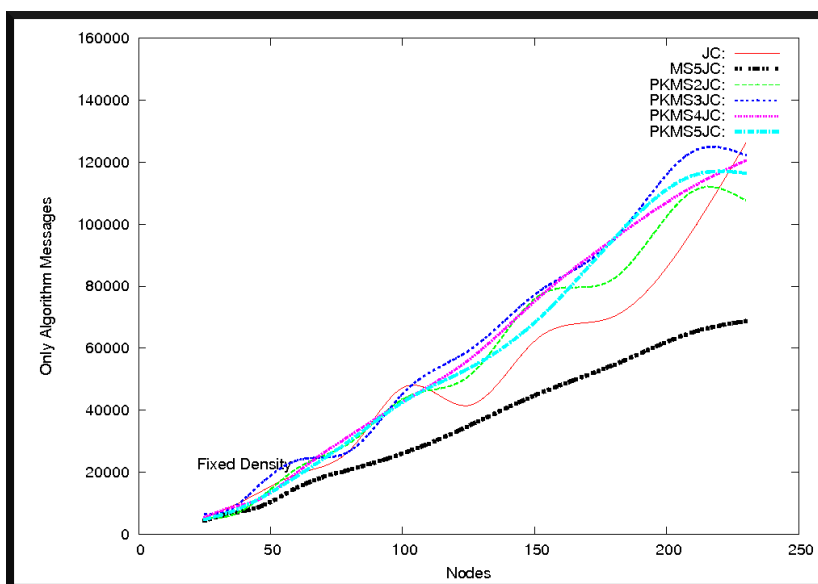
Σταθερή Πυκνότητα-Χρώματα

Όπως και με την σταθερή διάμετρο έτσι και εδώ έχουμε αύξηση των χρωμάτων κατά ένα σταθερό ποσοστό για τους ίδιους λόγους. Και εδώ είναι της τάξης των τεσσάρων με πέντε το πολύ χρωμάτων.

Αριθμός Μηνυμάτων



Σταθερή Πυκνότητα-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Πυκνότητα-Αριθμός Μηνυμάτων Αλγορίθμου

Με σταθερή την πυκνότητα, έχουμε περιπτώσεις όπου τα μηνύματα μας είναι περισσότερα ακόμα και από τον απλό. Λόγο της αύξησης της διαμέτρου, στην τελευταία φάση του ελέγχου των χρωμάτων από κάθε κόμβο, απαιτούνται πολλά

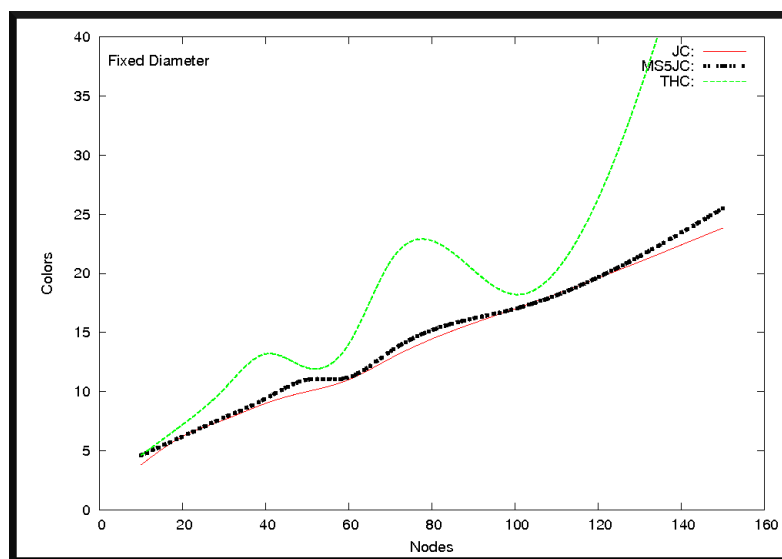
μηνύματα για την επιβεβαίωση σφάλματος ή ορθότητας σε ένα χρώμα. Στο σύνολο έχουμε ελάχιστα λιγότερα μηνύματα.

6.3.4 Αλγόριθμος δύο βημάτων

Ο Αλγόριθμος δύο βημάτων θα αναφέρεται και ως THColoring (Two-Hops Coloring) ή πιο απλά THC.

Αποτελέσματα σταθερής διαμέτρου

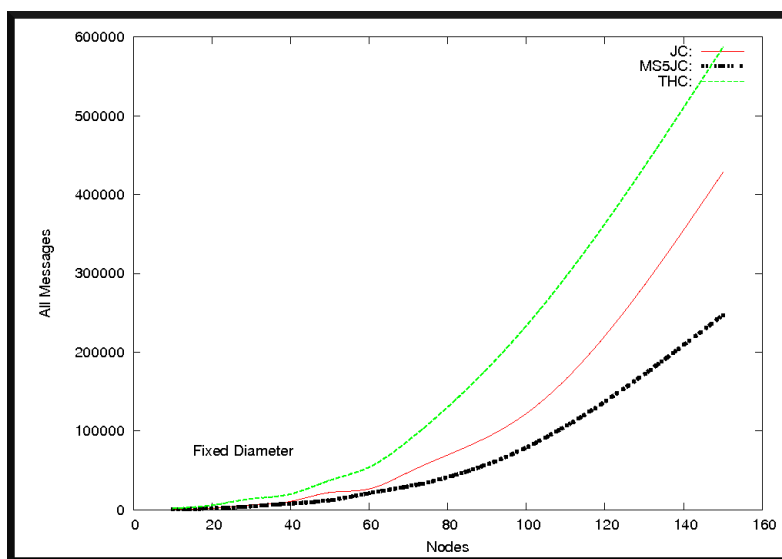
Χρωματικός Αριθμός



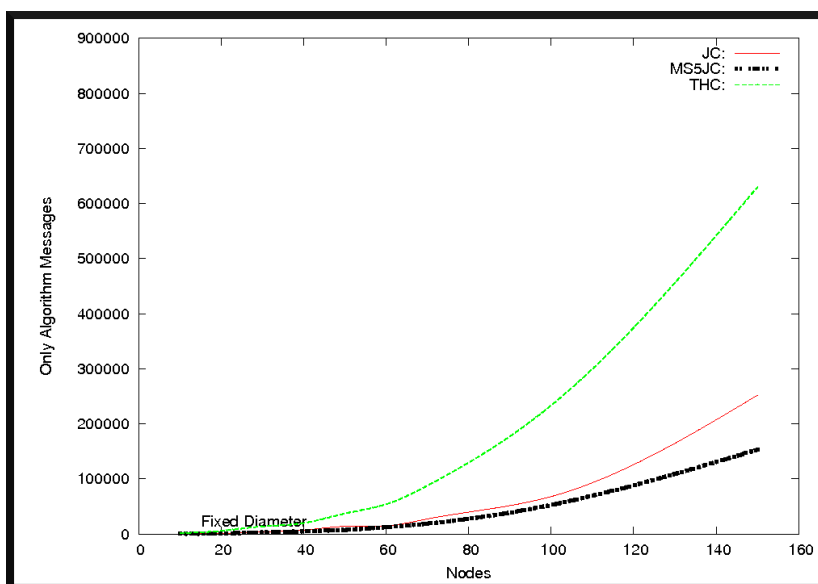
Σταθερή Διάμετρος-Χρώματα

Μέχρι τους εκατό περίπου κόμβους παρατηρούμε ότι τα χρώματα διατηρούν μια σταθερή χαμηλή τιμή σε σχέση με τον αρχικό αλγόριθμο. Δυστυχώς όμως για μεγάλους αριθμούς κόμβων τα πράγματα χειροτερεύουν αρκετά, όχι όμως σε βαθμό που να καθιστούν τον αλγόριθμο ιδιαίτερα κακό.

Αριθμός Μηνυμάτων



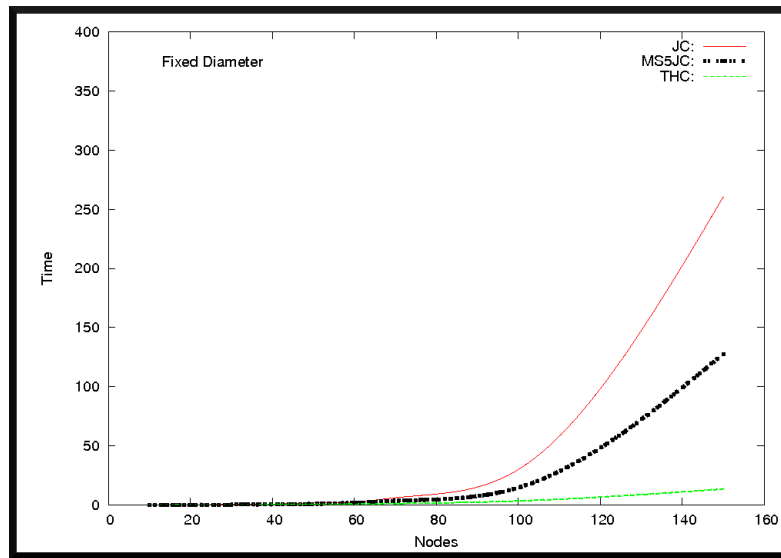
Σταθερή Διάμετρος-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Διάμετρος-Αριθμός Μηνυμάτων Αλγορίθμου

Ως προς τα μηνύματα βλέπουμε ότι ο αλγόριθμος των δυο βημάτων είναι χειρότερος από όλους τους προηγούμενους. Κάτι τέτοιο δικαιολογείται αν σκεφτούμε ότι σε κάθε γύρο γίνεται μία μικρή πλημμύρα από μηνύματα από όλους τους κόμβους σε όλους μέσα σε μία γειτονιά. Οι γειτονιές συνδέονται μεταξύ τους και πρακτικά είναι σαν να πλημμυρίζουμε το δίκτυο συνεχώς.

Χρόνος

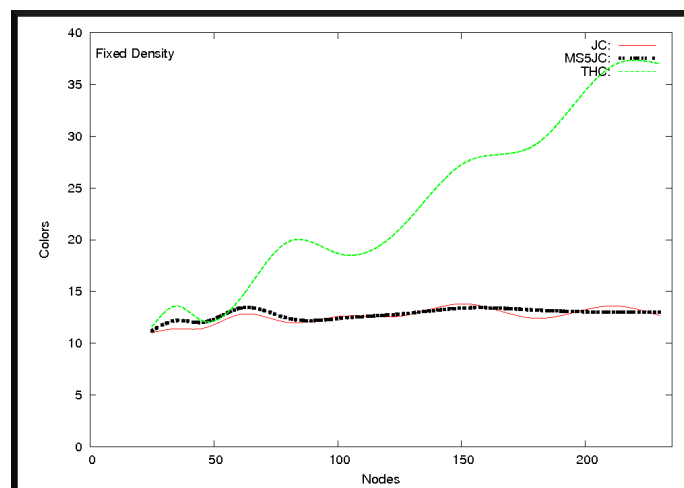


Σταθερή Διάμετρος-Χρόνος

Στον τομέα αυτό είναι που υπερσχύει αισθητά ο αλγόριθμος των δύο βημάτων. Ακόμα και όταν δεν έχουμε μετρήσεις που να λαμβάνουν υπόψη τους την παραλληλία, η απουσία της ανάγκης βημάτων που γίνονται το ένα μετά το άλλο με συγκεκριμένο ρυθμό, δίνει χρόνους ασύγκριτα καλύτερους με οτιδήποτε έχουμε δει μέχρι στιγμής.

Αποτελέσματα σταθερής πυκνότητας

Χρωματικός Αριθμός

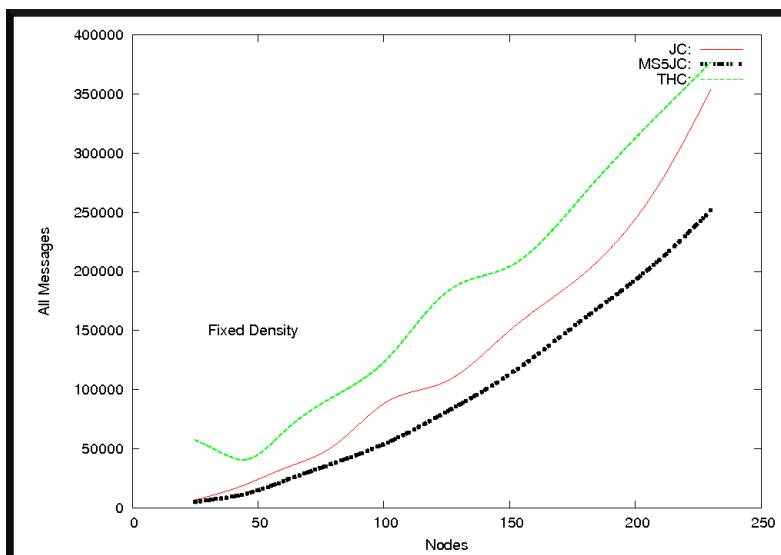


Σταθερή Πυκνότητα-Χρώματα

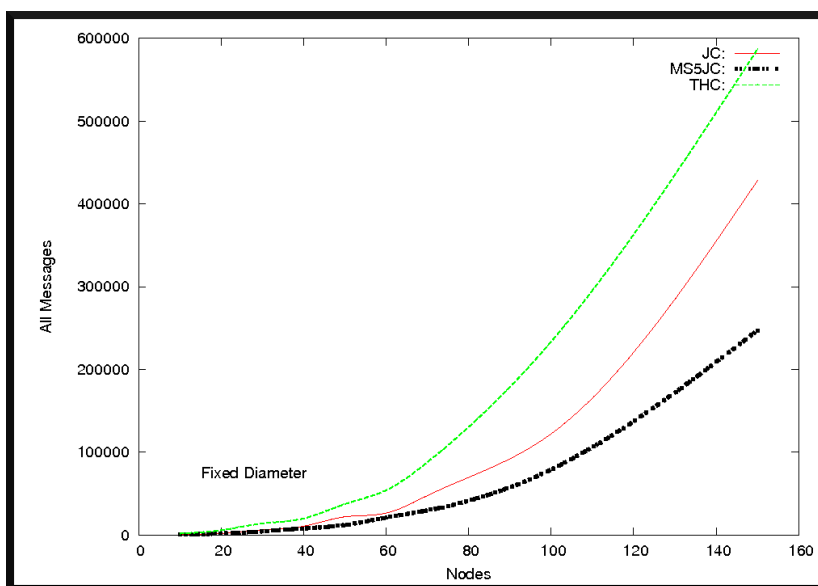
Εδώ τα χρώματα χειροτερεύουν αισθητά. Σε αντίθεση με τους προηγούμενους αλγόριθμους όπου ακόμα και στον παράλληλο με πολλαπλούς δικαστές είχαμε σταθερή απόδοση σε χρώματα, εδώ έχουμε συνεχή αύξηση σε σχέση με την είσοδο.

Το πρόβλημα της μερικής γνώσης γίνεται ιδιαίτερα αισθητό. Η αύξηση της διαμέτρου δεν επιτρέπει την εύκολη διάδοση χρωμάτων σε απομακρυσμένες γειτονιές.

Αριθμός Μηνυμάτων

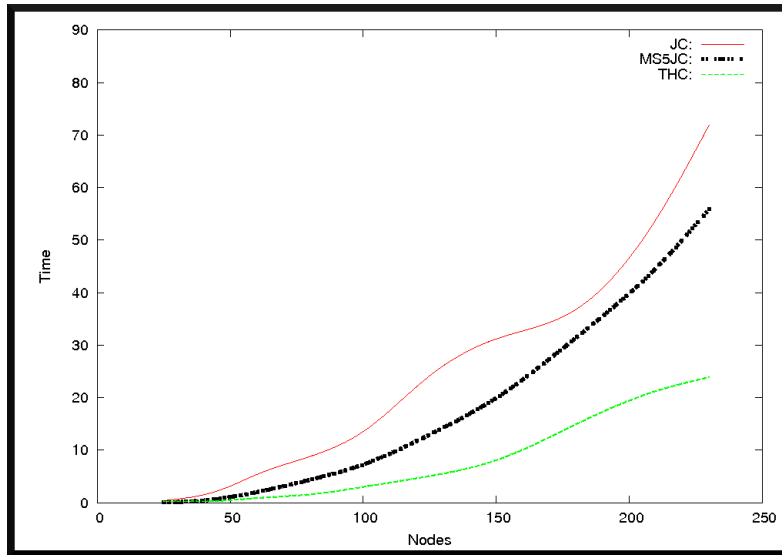


Σταθερή Πυκνότητα-Αριθμός Συνολικών Μηνυμάτων



Σταθερή Πυκνότητα-Αριθμός Μηνυμάτων Αλγορίθμου

Τα μηνύματα όπως και όταν κρατήσαμε σταθερή την διάμετρο είναι αισθητά περισσότερα, αν και όχι στον ίδιο βαθμό.

Χρόνος

Σταθερή Πυκνότητα-Χρόνος

Τα κέρδη στον χρόνο αν και πολύ μεγάλα, δεν είναι στην ίδια κλίμακα με τα προηγούμενα.

6.4 Συμπεράσματα

Παραπάνω είδαμε τα αποτελέσματα των αλγορίθμων που υλοποιήθηκαν. Ο τελευταίος αλγόριθμος με την τυχαία επιλογή $\log(n)$ κόμβων σε κάθε γύρω δεν αξιολογήθηκε γιατί τα αποτελέσματά του δεν έδιναν καλούς χρωματισμούς, ο αλγόριθμος έπρεπε να εκτελείται για πολύ μεγάλο χρόνο και πολλές φορές δεν τελείωνε ποτέ.

Για τους υπόλοιπους τέσσερις προκύπτουν τα παρακάτω συμπεράσματα:

- Για δίκτυα στα οποία μας ενδιαφέρει αποκλειστικά και μόνο ο χρωματικός αριθμός η επίλυση με τον ένα δικαστή είναι η βέλτιστη που μπορούμε να χρησιμοποιήσουμε.
- Ως γενική λύση για τα περισσότερα δίκτυα, η καλύτερη είναι ο σειριακός αλγόριθμος πολλαπλών δικαστών. Ο συγκεκριμένος δίνει καλό αποτέλεσμα ως προς τα χρώματα, τα λιγότερα μηνύματα - άρα στο σύνολο είναι ο καλύτερος ενεργειακά - και πολύ καλούς χρόνους.
- Αν ο χρόνος είναι το ζητούμενο αλλά με κάποια έμφαση στα χρώματα, ο αλγόριθμος με τους παράλληλους δικαστές είναι η ιδανική επιλογή μας. Τα μηνύματα είναι καλά σε αριθμό ενώ τα χρώματα προσεγγίζουν με ικανοποιητικό τρόπο αυτά που δίνουν οι δυο προηγούμενοι.
- Τέλος αν ο χρόνος είναι το μόνο που μας απασχολεί, ο ιδανικότερος είναι ο αλγόριθμος των δυο βημάτων. Ακόμα και αν τα μηνύματα είναι πολλά σε σχέση με τα αντίστοιχα των υπολοίπων, ο αλγόριθμος αυτός είναι ο πιο δίκαιος ενεργειακά γιατί απαιτεί το ίδιο από όλους τους κόμβους του.
- Οι επιλύσεις με δικαστές, όσο καλά και να είναι τα αποτελέσματά τους σε χρωματικό αριθμό σε σχέση με την λύση των δυο βημάτων έχουν και άλλα μειονεκτήματα απέναντί του. Ο two-hops δεν απαιτεί καμία ομαδοποίηση και κανέναν κόμβο ιδιαίτερα πιο ισχυρό από τους άλλους.

Το γενικό συμπέρασμα είναι ότι κανένας από τους αλγορίθμους που παρουσιάστηκαν δεν υπερσχύει σε όλους τους τομείς. Στα δίκτυα αισθητήρων είναι πολλές φορές διαφορετικές οι απαιτήσεις, και αυτό από μόνο του καθιστά χρήσιμους όλους τους παραπάνω αλγορίθμους. Το ζήτημα της επιλογής είναι καθαρά σχεδιαστικό και έχει να κάνει με τα πλεονεκτήματα του καθενός σε συνδυασμό με το τι ζητάμε, για να καταλήξουμε στο ποιον θα επιλέξουμε στην πράξη.

6.5 Μελλοντική Εργασία και Στόχοι

Η εργασία μας πάνω στα ασύρματα δίκτυα αισθητήρων και στους αλγορίθμους χρωματισμού, μας έδωσε αρκετά και χρήσιμα στοιχεία ως προς την πολυπλοκότητα και την απόδοση προσεγγιστικών λύσεων για το πρόβλημα αυτό. Πραγματοποιήθηκε εκτενής μελέτη πάνω σε δύο ειδών σύνολα τοπολογιών και είδαμε τις συμπεριφορές των αλγορίθμων μας πάνω σε αυτά. Παρ' όλα αυτά υπάρχει πολύ μεγάλο βάθος στην έρευνα που μπορεί να γίνει πάνω σε αυτό το θέμα για να καλυφθεί από μια τέτοια εργασία. Ο κυριότερος μας στόχος για το μέλλον είναι να διερευνήσουμε το ζήτημα αυτό εκτενέστερα. Αυτό σκοπεύουμε να το πραγματοποιήσουμε με διάφορους τρόπους.

1. Ο πρώτος μας στόχος είναι να εκτελέσουμε τους αλγορίθμους σε πραγματικούς αισθητήρες και να δούμε την συμπεριφορά σε πραγματικό περιβάλλον. Αυτό θα μας δώσει μια καλή αντίληψη πραγματικών χρόνων εκτέλεσης και μια πιο σφαιρική άποψη για τα συγκριτικά πλεονεκτήματα των αλγορίθμων.
2. Επιπλέον είναι σημαντικό να δούμε την συμπεριφορά και τις απαιτήσεις που έχουν σε περιπτώσεις με σφάλματα. Η ανάκαμψη από τα σφάλματα σε δίκτυα αισθητήρων είναι από τα πιο σημαντικά κομμάτια της έρευνας που πραγματοποιείται, και είναι απαραίτητη η μελέτη των αλγορίθμων για περιβάλλοντα όπου χάνονται μηνύματα.
3. Ακόμα πρέπει να γίνει επέκταση του κώδικα ώστε να μπορεί να ανακάμπτει από μεταβολές στο ίδιο το δίκτυο. Στην πράξη πολλές είναι οι φορές που οι κόμβοι χάνονται (είτε τελειώνει η ενέργειά τους ή καταστρέφονται). Σε αυτές τις περιπτώσεις ο χρωματισμός πρέπει να αναθεωρείται και να παραμένει πάντα καλός.
4. Χρήσιμες δοκιμές είναι και άλλοι αλγόριθμοι δρομολόγησης. Ο αλγόριθμος TORA που χρησιμοποιήσαμε θεωρείται καλή επιλογή για δοκιμές, είναι όμως πολύ πιθανόν να υπάρχουν άλλοι που μας δίνουν ακόμα καλύτερα αποτελέσματα.
5. Καινούργιοι αλγόριθμοι πλήρως κατανεμημένοι ευελπιστούμε να υλοποιηθούν στο εγγύς μέλλον, ενώ σημαντικός στόχος μας είναι να δοκιμάσουμε τον χρωματισμό και σε άλλα είδη δικτύων, πέραν αυτών με την σταθερή διάμετρο και την σταθερή πυκνότητα.
6. Τέλος στόχος μας είναι να δοκιμάσουμε τους αλγορίθμους μας χρησιμοποιώντας ομαδοποίηση που να συμβαίνει κατά την διάρκεια της εκτέλεσης, με στόχο να μην απαιτείται ανθρώπινη παρέμβαση. Οι αλγόριθμοι που παρουσιάστηκαν λειτουργούν αρκετά καλά ακόμα και αν η επιλογή τους γίνει τυχαία, όμως θεωρούμε ότι χρειάζεται έστω μια υποτυπώδης διαδικασία επιλογής τους.

Βιβλιογραφία

- [1] Gonthier, Georges (2008), "*Formal Proof–The Four-Color Theorem*", Notices of the American Mathematical Society 55
- [2] Appel, Kenneth; Haken, Wolfgang (1989), "*Every Planar Map is Four-Colorable*", Providence, RI: American Mathematical Society, ISBN 0821851039
- [3] Richard M. Karp (1972). "*Reducibility Among Combinatorial Problems*". in R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. pp. 85-103.
- [4] Stephen Cook (1971). "*The Complexity of Theorem Proving Procedures*". Proceedings of the third annual ACM symposium on Theory of computing. pp. 151-158.
- [5] Skiena, S. "*Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*". Reading, MA: Addison-Wesley, 1990.
- [6] Holyer - Siam J. Comput. "*The NP-completeness of edge-colouring*", 1981.
- [7] Marx, Daniel (2004), "*Graph colouring problems and their applications in scheduling*", Periodica Polytechnica, Electrical Engineering, 48, pp. 11-16
- [8] Gregory J. Chaitin, Mark A. Auslander, Ashok K. Chandra, John Cocke, Martin E. Hopkins, and Peter W. Markstein. "*Register allocation via coloring.*" Computer Languages, 6:47-57, 1981.
- [9] Hideo Ogawa. "*Labeled point pattern matching by delaunay triangulation and maximal cliques. Pattern Recognition*", 19(1):35-40, 1986.
- [10] Alan Cobham (1965). "*The intrinsic computational difficulty of functions*". Proc. Logic, Methodology, and Philosophy of Science II. North Holland.
- [11] Stephen Cook. "*The Complexity of Theorem Proving Procedures*", 1971

- [12] Vijay V. Vajirani. "*Approximation Algorithms*"
- [13] Panagiota N. Panagopoulou, Paul G. Spirakis. "*A Game Theoretic Approach for Efficient Graph Coloring*".
- [14] Fabian Kuhn, Roger Wattenhofer. "*On the Complexity of Distributed Graph Coloring*", 2006.
- [15] M. Luby. "*A simple parallel algorithm for the maximal independent set problem*". Proc. of ACM Symposium on Theory of Computing (1985), 1–10.
- [16] Christian Scheideler, Kishore Kothapalli, Christian Schindelhauer, Melih Onus. "*Distributed Coloring in $O(\sqrt{\log n})$ Bit Rounds*", 2006.
- [17] Ojvind Johansson. "*Simple Distributed $\Delta + 1$ -Coloring of Graphs*", 2005.
- [18] Dijkstra, E.W. "*A note on two problems in connexion with graphs*". Numerische Mathematik 1: 269–271, (1959)
- [19] Richard Bellman. "*On a Routing Problem, in Quarterly of Applied Mathematics*", 16(1), pp.87-90, 1958.
- [20] Ossama Younis, Sonia Fahmy. "*Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach*", IEEE Transactions on Mobile Computing, 2004.
- [21] Eshref Januzaj, Hans-Peter Kriegel, Martin Pfeifle. "*Towards Effective and Efficient Distributed Clustering*". Workshop on Clustering Large Data Sets (ICDM2003), Melbourne, FL, 2003.
- [22] Stefano Basagni. "*Distributed Clustering for ad-hoc networks*", ispan, 1999.
- [23] Hillol Kargupta, Weiyun Huang, Krishnamoorthy Sivakumar, Erik Johnson. "*Distributed Clustering Using Collective Principal Component Analysis*", Knowledge and Information Systems 3:422-448, 2001 .