An Absorbing Random Walk Classifier for Graphs

Παπαμιχαήλ Άγγελος Α.Μ 353

1 Φεβρουαρίου 2017

Περίληψη

Στην παρούσα εξαμηνιαία εργασία για το μάθημα Κοινωνικά Δίκτυα και Μέσα Ενημέρωσης δημιουργήθηκε ένας νέος τρόπος ομαδοποίησης ενός κοινωνικού δικτύου που βρίσκεται σε μορφή γράφου. Ο αλγόριθμος δανείζεται χαρακτηριστικά απο τον $K\ Means$ ενώ χρησιμοποιεί τη μετρική *Absorbing Random Walks* ως μέθοδο ομαδοποίησης. Η υλοποιήση εξετάστηκε ως προς τον χρόνο εκτέλεσής της και την αξιοπιστία των αποτελεσμάτων της μέσω της σύγκρισης με δύο γνωστές μεθόδους ομαδοποίησης τον $K\ Means$ και το *Spectral Clustering*, πάνω σε γνωστούς γράφους. Τέλος δημιουργήθηκε μια παράλληλη υλοποίησή του βασισμένη στη βιβλιοθήκη της *google* για γράφους, *pregel* 10 .

1 Εισαγωγή

Ένα απο τα πιό επίκαιρα προβλήματα στο χώρο των μέσων δικτύωσης είναι εύρεση κοινοτήτων, γνωστό και ως *community detection problem*. Το πρόβλημα αυτό χωρίζεται σε δύο κύρια σκέλη, την αξιοπιστία των αλγόριθμων που χρησιμοποιούνται καθώς και την χρονική και χωρική τους πολυπλοκότητα. Στην παρούσα εργασία υλοποιήθηκε ένας αλγόριθμος με πολυπλοκότητα χρόνου $\mathrm{O}(n\log(n))$ και πολυπλοκότητα χώρου $\mathrm{O}(n)$, όπου $n = |V| * |E|$ για γράφο $G = (V, E)$.

Η αναφορά για το *project* χωρίζεται σε πέντε κεφάλαια, αρχικά θα αναλύσουμε τον αλγόριθμο που προτείνουμε, τον *Propagation Absorbing Random Walks*, στο εξής $PARW$, στο κεφάλαιο 2. Στο κεφάλαιο 3 θα παρουσιάσουμε την παράλληλη υλοποίηση του αλγόριθμου, *Parallel Propagation Absorbing Random Walks*, στο εξής *Parallel* $PARW$, ενώ στο κεφάλαιο 4 παρατίθονται αποτελέσματα του ελέγχου απόδοσης των $PARW$ και *Parallel* $PARW$ σε σχέση με άλλους αλγόριθμους σε τρείς διαφορετικούς γνωστούς γράφους το $zachary's\ karate\ club$3, το γράφο ποδοσφαίρου της σαιζόν του φθινοπόρου του 2000 όπως έχει δοθεί απο τους M. Girvan και M. Newman4 και το *dblp collaboration* 5. Τέλος στο κεφάλαιο 5 αναφερόμαστε στα συμπεράσματα που εξάγαμε απο τη δουλειά μας και προτείνουμε κατευθύνσεις για μελλοντικές εργασίες.

Να σημειωθεί οτι στο συνολό του, ο κώδικας που υλοποιήθηκε είναι γραμμένος σε *python* έκδοσης 3.5.

2 Ο αλγόριθμος *Propagation Absorbing Random Walks*

Η κύρια ιδέα πίσω απο τον αλγόριθμο $PARW$ είναι η εξής. Ορίζουμε τον αριθμό των κοινοτήτων που ψάχνουμε, έστω K και τυχαία ορίζονται K αρχικά κέντρα. Έστω ένας κόμβος v ο οποίος έχει ώς γείτονες τους v_1, v_1 ... v_n η πιθανότητα του να απορροφηθεί απο το k_i κέντρο είναι το αθροισμα των πιθανοτήτων των γειτόνων του να απορροφηθούν επί 1 διά το *degree* του ίδιου. Αρχικοποιούμε όλους τους κόμβους με τη τιμή 0 πλήν των κόμβων κέντρων τα οποία έχουν την τιμή 1, σε κάθε επανάληψη οι τιμές διαδίδονται στο γράφο, όταν φτάσουμε στο σημείο όπου μια μεγάλη πλειοψηφία κόμβων έχει πάρει τιμές τότε γίνεται η ανάθεση νέων κέντρων.

Τα νέα κέντρα επιλέγονται με βάση μία απο τις παρακάτω $fitness$ συναρτήσεις, έστω για κάποιο κέντρο $k \in K$:

$$1.f(\mathbf{k}) = \max_{\forall v \in V} (ap[k][v] * K + \sum_{\forall j \in K-k} \frac{ap[k][v]}{ap[j][v]})$$

$$2.f(\mathbf{k}) = \max_{\forall v \in V} \sum_{\forall j \in K-k} \frac{ap[k][v]}{ap[j][v]})$$

Όπου $ap[i][v]$, *absorbing propability*, είναι η πιθανότητα ο v κόμβος να απορροφηθεί απο το i κέντρο. Η ιδέα πίσω απο τη συνάρτηση αυτή είναι ανάγκη ένας κόμβος που ανήκει σε μια ομάδα να έχει καλή πιθανότητα να απορροφηθεί απο το κέντρο της ομάδας του και μικρή πιθανότητα να απορροφηθεί απο κέντρα άλλων ομάδων. Το άθροισμα μέσα στη συνάρτηση μεγιστοποιείται για κόμβους που έχουν μικρή πιθανότητα να απορροφηθούν απο κέντρα πέραν της ομάδας όπου ανήκουν οι ίδιοι. Ένας κόμβος μπορεί να γίνει κέντρο μόνο στην ομάδα που ήδη

συμμετέχει. Και οι δύο συναρτήσεις έδωσαν καλά αποτελέσματα, όμοια μεταξύ τους.

Algorithm 1 Propagation Absorbing Random Walks

(Preparation Phase)
Compute K random nodes to use as centers
Compute degree of nodes
Create a hashmap with neighbors of each node
Initialize labels
(End of Preparation Phase)
while Two consecutive are not close enough **do**
    Initialize absorbing probabilities, centers have 1, others 0
    Create Random set of nodes, based on V
    while random set of nodes not empty **do**
        v = pop first of the set
        for $k \in K$ **do**
            Set v's probability of being absorbed by k to 0
            for $neighbors \in v$ **do**
                if neighbor not a center or neighbor is a center and the neighbor is *the* k center **then**
                    Update probability of being absorbed based on (neighbor's * 1/degree(v))
                end if
            end for
        end for
        for $v \in V$ **do**
            Compute label of v based on the maximum probability of being absorbed
            Add maximum probability to a sum for convergence purposes
        end for
        Compute new Centers based on function 1 or 2 from above
    end while
end while
Finish

3 Παραλληλοποίηση

Ιδιαίτερο ενδιαφέρον παρουσιάζει η παραλλήλοποίηση αλγορίθμων στο χώρο του *community detection* αλλά και γενικότερα στο χώρο της επεξεργασίας γράφων, κυρίως εξαιτίας του μεγέθους των δεδομένων που καθιστά μη αξιοποιήσιμους ακόμα και αλγόριθμους της τάξης $O(n^2)$. Η παρούσα εργασία κατάφερει να φέρει εις πέρας την παραλλήλη υλοποίηση του αλγόριθμου που παρουσίασε αξιοποιώντας την state of art βιβλιοθήκη pregel, αναπτυγμένη απο τη google. Επειδή η βασική υλοποίηση του pregel αφορά τη cpp και εμείς αναπτύξαμε την εργασία μας σε python δουλέψαμε με μια "toy" έκδοση, η οποία όμως έχει τις βασικές λειτουργίες του pregel. Η βασική διαφορά είναι οτι αυτή η έκδοση μπορεί να λειτουργήσει σε ένα μόνο μηχάνημα και όχι σε cluster . Ακολουθεί μια σύντομη περιγραφή του τί ακριβώς είναι το pregel .

Η βιβλιοθήκη pregel είναι μια βιβλιοθήκη για κατανεμημένους υπολογισμούς. Αναπτύχθηκε απο τη google και δημοσιεύθηκε το 2010 10. Απο τότε χρησιμοποιείται σε πληθώρα εφαρμογών της ίδιας της εταιρείας. Η βιβλιοθήκη στοχεύει περισσότερο σε αλγόριθμους που λειτουργούν με γράφους, που σημαίνει οτι αλγόριθμοι που χρησιμοποιούν το networkx 8 ή το snap9 μπορούν να ευνοηθούν απο τη χρήση της. Παραδείγματα προβλημάτων που μπορούν να ωφεληθούν απο τη χρήση της pregel είναι η απόφαση εάν δύο γράφοι συνδέονται, εύρεση στενά συνδεδεμένων κοινοτήτων σε γράφο και πολλά άλλα.

Εισέρχοντας τώρα στον ίδιο τον κώδικά της, βρίσκουμε τρείς βασικές κλάσεις, τις Vertex, Pregel, Worker. Η σχέση τους φένεται στο σχήμα 1. Η κλάση Vertexπεριέχει ένα *id*, μία τιμή που μπορεί να ειναι ακεραιος, διάνυσμα ή χαρακτήρας, τους γείτονες του κόμβου, εισερχόμενα και εξερχόμενα μηνύματα, την πληροφορία άν είναι ενεργός και σε ποιό *superstep* βρισκόμαστε. Ύστερα απο συγκεκριμένο αριθμό απο *superstep* ο κόμβος απενεργοποιείται και δε δέχεται άλλα μηνύματα, όταν όλοι απενεργοποιηθούν ο αλγόριθμος τελειώνει.

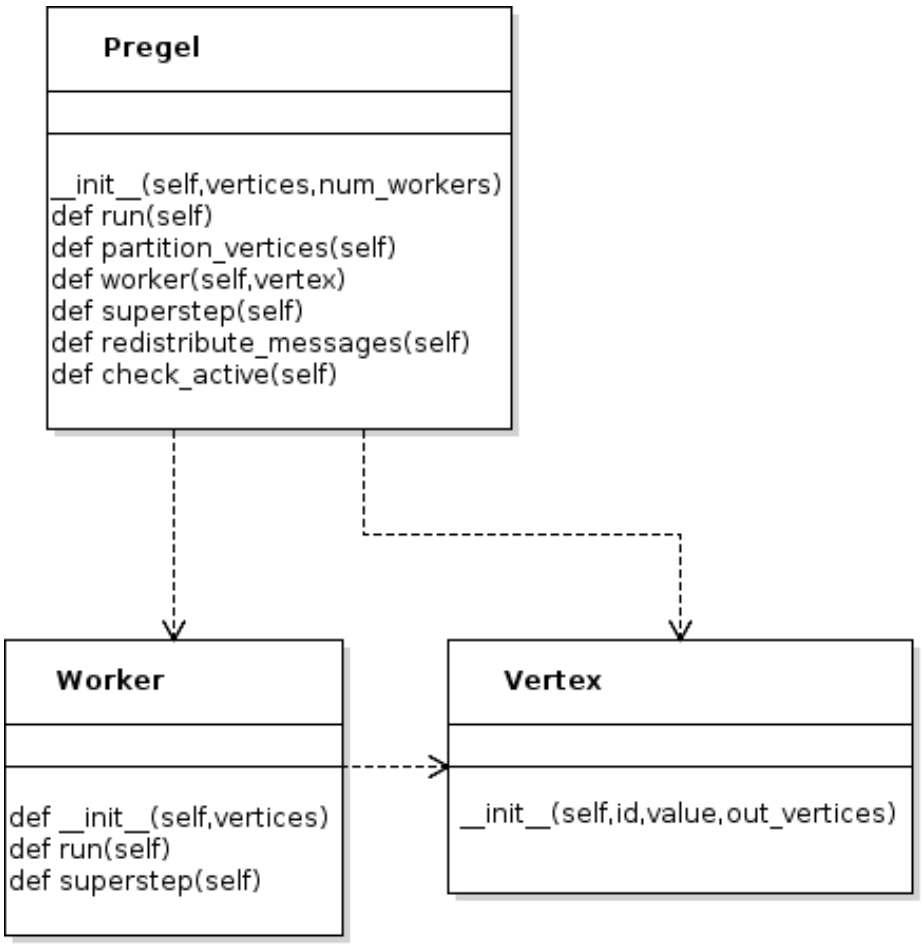
Η κλάση Worker αρχικοιποιεί τα νήματα που θα τρέχουν παράλληλα. Επίσης υλοποιεί τη *superstep* η οποία για κάθε κόμβο στο συνολό μας κάνει *update* τις τιμές. Η *update* είναι μια συνάρτηση που μπορούμε να κάνουμε *override* όπως και κάναμε στην υλοποίησή μας.

Τέλος η κλάση Pregel αρχικοποιείται με ορίσματα το σύνολο απο Vertex(es), χωρίζει το σύνολο αυτό σε υποσύνολα τα οποία ανακατανέμονται ισάριθμα στα νήματα και για όσο υπάρχουν *active* κόμβοι τρέχει τη *superstep*. Η *superstep* της κλάσης $Pregel$ καλεί την κλάση $Worker$ για κάθε νημα/υποσύνολο όπου όπως εξήγαμε παραπάνω καλείται η *superstep* της $Worker$.

Η υλοποίησή μας βασίζεται στις παραπάνω λειτουργίες. Συγκεκριμένα, αρχικά φτιάχνει το σύνολο απο τους κόμβους μαζί με τους γειτονές τους και θέτει ως τιμή για τον καθένα ένα διάνυσμα μεγέθους ίδιου με τα κέντρα που ορίζουμε. Αρχικά το διάνυσμα αυτό έχει μηδενικές τιμές εκτός και αν αναφερόμαστε σε κέντρα. Τα κέντρα απενεργοποιούνται. Εφόσον έγινε η αρχικοποίηση καλούμε το Pregel που όμως για να κάνει ενημέρωση τιμών θα χρησιμοποιεί μια δικιά μας μέθοδο. Η μέθοδος αυτή είναι η *update*, της κλάσης $Parw$. Η σχηματική αναπαράσταση, σε UML 11, βρίσκεται στο σχήμα 2.

Η ιδιαιτερότητα στη διαχείριση των νημάτων εκ μέρους της *python* μας ανάγκασε να προσθέσουμε μια επιπλέον κλάση που καλεί την κύρια όπως φένεται στο σχήμα 2. Η κλάση $ParallelParwService$ καλεί την $ParallelAbsorbingRandomWalks$ που επιστρέφει το μέσο $clustering\ coefficient$ της διαμέρισης, τα επιλεχθέντα κέντρα, το κέντρο όπου ανήκει ο κάθε κόμβος.

Σημαντική σημείωση είναι το γεγονός ότι το συνολικό πλήθος των φορών που ενας κόμβος κάνει $update$ είναι 10 για κάθε επανάληψη, ενώ στον $Parw$ που συνήθως συγκλίνει στο τρίτο βήμα οι συνολικές επαναλήψεις για ένα κόμβο είναι 15. Επίσης έγιναν αλλαγές και στο κώδικα της βιβλιοθήκης ώστε να ανταποκρίνεται στις ανάγκες του $PARW$. Συγκεκριμένα αλλάξαμε το είδος τιμών για έναν κόμβο απο ακέραιο σε διάνυσμα όπως αναφέρθηκε παραπάνω και εισάγαμε μια μεταβλητή K για να αλλάζουμε το πλήθος των κέντρων.



Σχήμα 1: UML διάγραμμα σχέσεων κλάσεων στη βιβλιοθήκη $pregel$

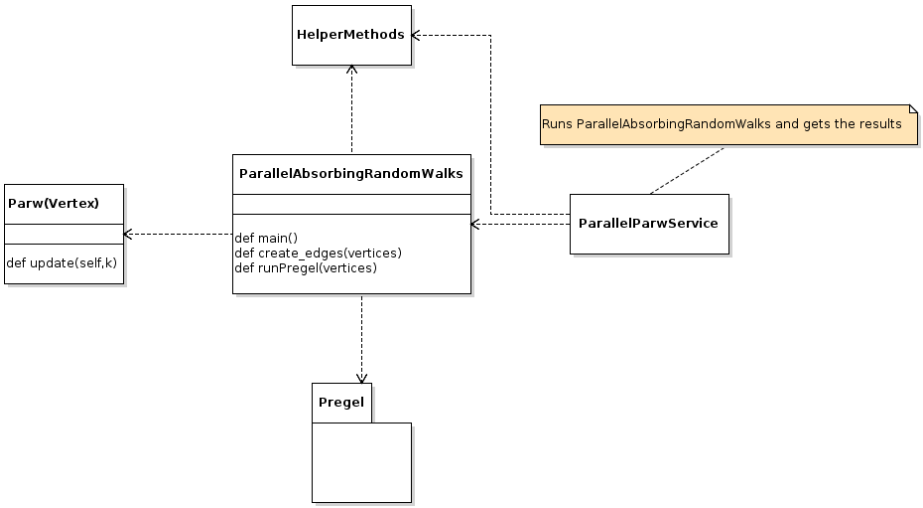


Figure 2: UML διάγραμμα σχέσεων κλάσεων στην υλοποίηση της παράλληλης έκδοσης του *Propagation Absorbing Random Walks*

4 Αξιολόγηση

Για την σύγκριση της υλοποίησής μας θα χρησιμοποιήσουμε την βιβλιοθήκη $scikit - learn$[6]. Η βιβλιοθήκη αυτή είναι ένα απο τα καλυτερα εργάλεια για *python* στο χώρο του *machine learning*. Για τα πειράματά μας θέτουμε ώς ένα ελάχιστο τη χρήση δύο πολύ διαφορετικών αλγόριθμων, τον $Kmeans$ και τον *spectral clustering*.

Το είδος των πειραμάτων χωρίζεται σε δύο σκέλη, την ποιότητα των αποτελεσμάτων και την ταχύτητα διεκπαιρέωσης της διαδικασίας *clustering*. Όσο αφορά την ποιότητα χρησιμοποιήσαμε την μετρική $clustering\ coefficient$. Η μετρική αυτή μας δείχνει το μέγεθος στο οποίο μια κοινότητα είναι πυκνά συνδεδεμένη. Συγκεκριμένα, η τιμή για κάθε κόμβο u ορίζεται ως το σύνολο των τριγωνων στο οποίο ανήκει επί 2, διαιρώντας με το $degree$ επί το $degree - 1$ όπως φένεται στην εξίσωση 4. Το άθροισμα αυτών των τιμών δια το σύνολο των κόμβων της εκάστοτε κοινότητας μας δίνει το $clustering\ coefficient$.

$$eq3 : c_u = \frac{2T(u)}{deg(u)(deg(u)-1)},$$

$$eq4 : C = \frac{1}{n}\sum_{v \in G} c_v,$$

Η μέτρηση του $coefficient$ γίνεται μέσω του χωρισμού του αρχικού γράφου σε *induced* υπογράφους μέσω μιας διαδικασίας όπου θέτουμε ως βάρη στις ακμές τα *labels* των κόμβων. Οι γράφοι που χρησιμοποιήθηκαν για τον ποιοτικό έλεγχο είναι ο zachary's karate club και ο American football games μεταξύ των κολλεγίων τη σαιζόν φθινοπώρου του 2000.
Η διαφορά στη χρονική πολυπλοκότητα μεταξύ των αλγορίθμων μετρήθηκε με τη χρήση του γράφου *dblp* απο τον οποίο φορτώσαμε περιορισμένο πλήθος γραμμών καθώς οι πίνακες που χρησιμοποιεί ο $Kmeans$ και ο *spectral* εμφάνιζαν σφάλμα

ώς προς το μέγεθος της μνήμης.

Ο γράφος με τη σχολή καράτε είναι ένας απο τους κλασσικούς στην βιβλιογραφία για προβλήματα *clustering* και περιέχει δύο κοινότητες. Τέλος ο γράφος με τις ομάδες ποδοσφαίρου περιέχει έντεκα κοινότητες.

4.1 Ποιότητα Αποτελεσμάτων

Αρχικά θα παραθέσουμε το πίνακα4.1 με το *clustering coefficient* του γράφου *karate club* όπου η κάθε γραμμή αφορά ένα αλγόριθμο και η στήλη την τιμή για κάθε ομάδα. Ακολουθεί ο αντίστοιχος για 3 κέντρα. Επίσης διδονται γραφικές αναπαραστάσειςς των αποτελεσμάτων των αλγόριθμων για *community detection* στο γράφο *karate club*.

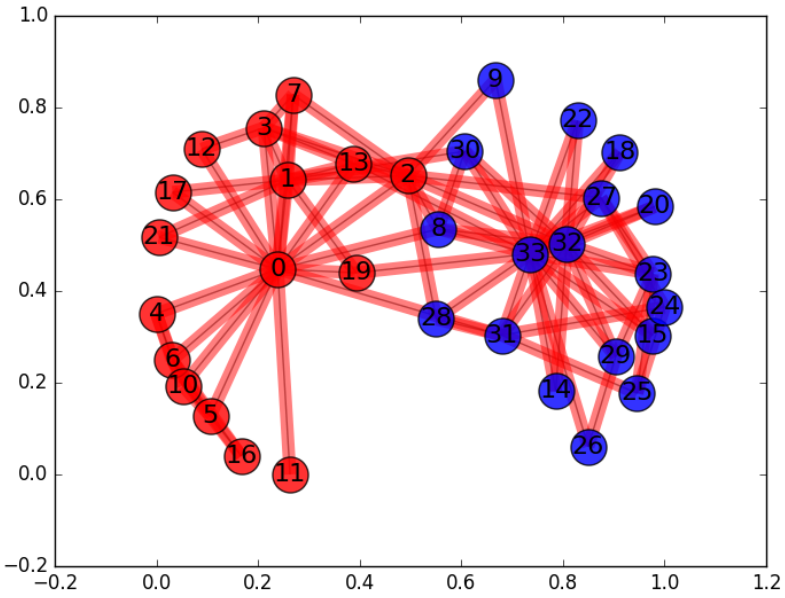
Για το γράφο με τις ομάδες ποδοσφαίρου παρατίθεται το *clustering coefficient* για τη κάθε κοινότητα στον πίνακα παρακάτω. Σε αυτό το σημείο να σημειωθεί οτι οι αναγραφόμενες λύσεις για κάθε αλγόριθμο προέκυψαν ύστερα απο αρκετές επαναλήψεις, όχι περισσότερες απο δέκα.

Οι πίνακες αποδεικνύουν ότι ο $PARW$ αποδίδει τουλάχιστον το ίδιο καλά με τους $Kmeans$ και τον *spectral*. Ενώ όπως θα δούμε στη συνέχεια είναι τουλάχιστον ταχύτερος απο τον *spectral*, αρκετά πιο αργός απο τον $Kmeans$, αλλά με μικρότερη χωρική πολυπλόκοτητα.

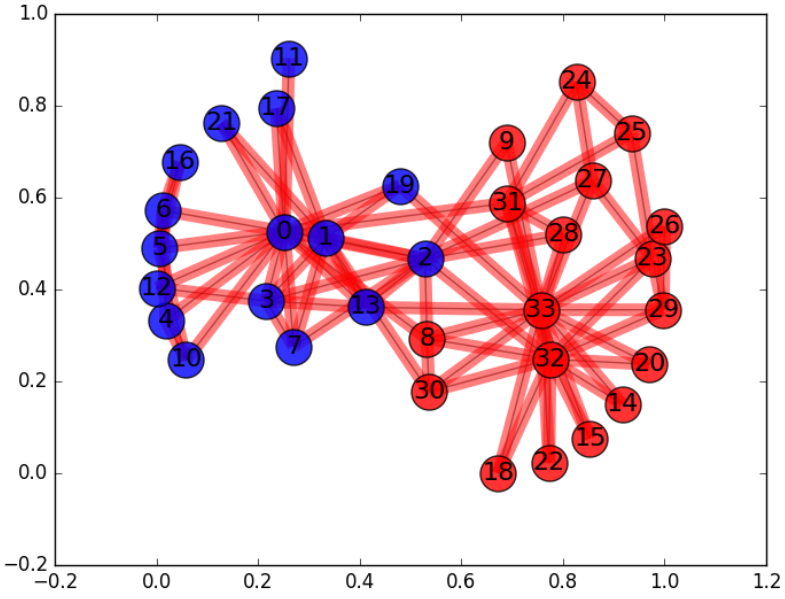
$$\begin{pmatrix} PARW & 0.719 & 0.423 \\ Kmeans & 0.719 & 0.423 \\ Spectral & 0.719 & 0.423 \\ Parralel\ PARW & 0.719 & 0.423 \end{pmatrix}$$

$$\begin{pmatrix} PARW & 0.5 & 0.287 & 0.103 \\ Kmeans & 0.0595 & 0.538 & 0.719 \\ Spectral & 0.598 & 0.419 & 0.059 \\ Parralel\ PARW & 0.0 & 0.499 & 0.595 \end{pmatrix}$$

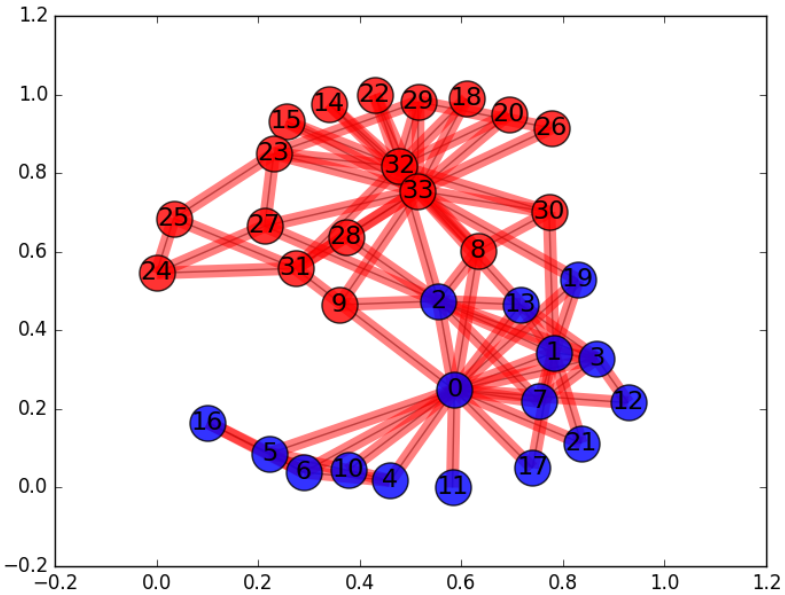
$$\begin{pmatrix} PARW & 0.317 & 0.284 & 0.302 & 0.24 & 0.318 & 0.305 & 0.216 & 0.28 & 0.293 & 0.394 & 0.393 \\ Kmeans & 0.263 & 0.347 & 0.221 & 0.291 & 0.465 & 0.282 & 0.22 & 0.166 & 0.244 & 0.214 & 0.388 \\ Spectral & 0.178 & 0.221 & 0.293 & 0.395 & 0.234 & 0.244 & 0.291 & 0.22 & 0.347 & 0.416 & 0.388 \\ Parralel\ PARW & 0.175 & 0.445 & 0.052 & 0.361 & 0.348 & 0.305 & 0.32 & 0.376 & 0.22 & 0.202 & 0.302 \end{pmatrix}$$



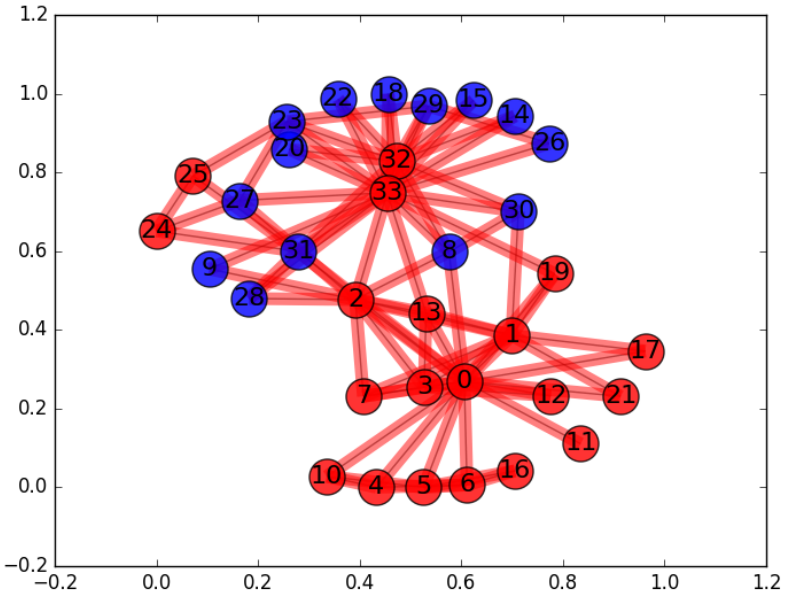
Σχήμα 3: Zachary's Karate club, PARW, K=2



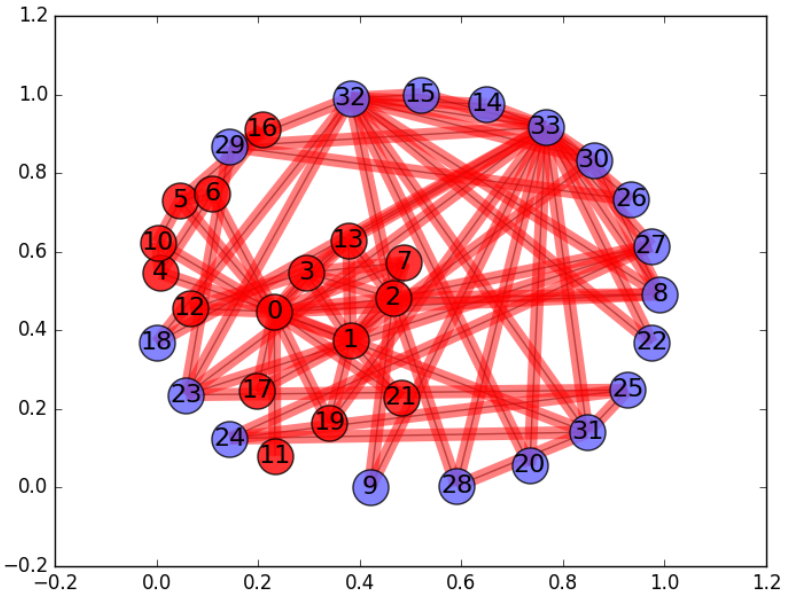
Σχήμα 4: Zachary's Karate club, Kmeans, K=2



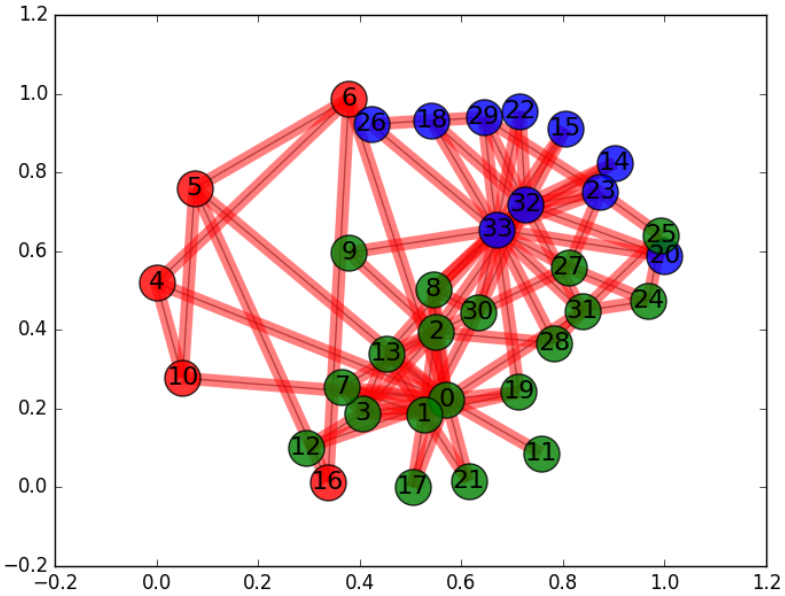
Σχήμα 5: Zachary's Karate club, Spectral, gamma =0.1, K=2



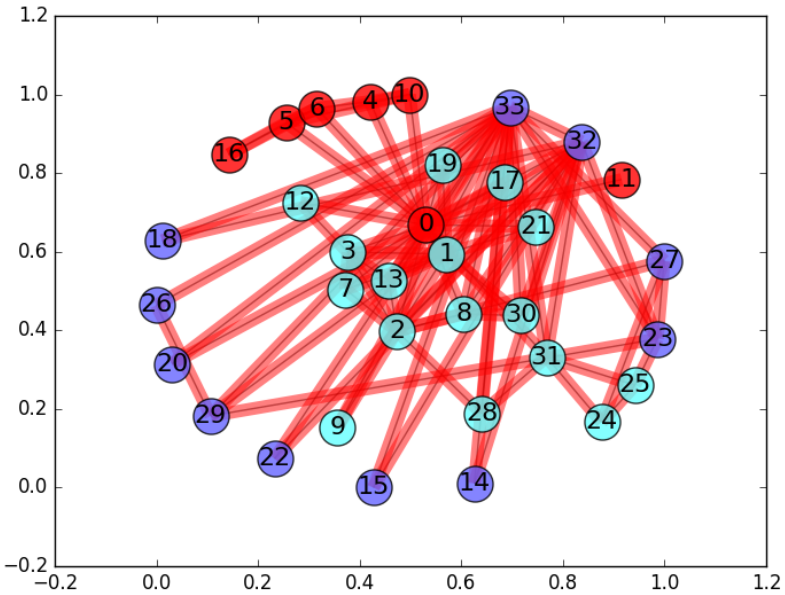
Σχήμα 6: Zachary's Karate club, Spectral, gamma =0.3, K=2



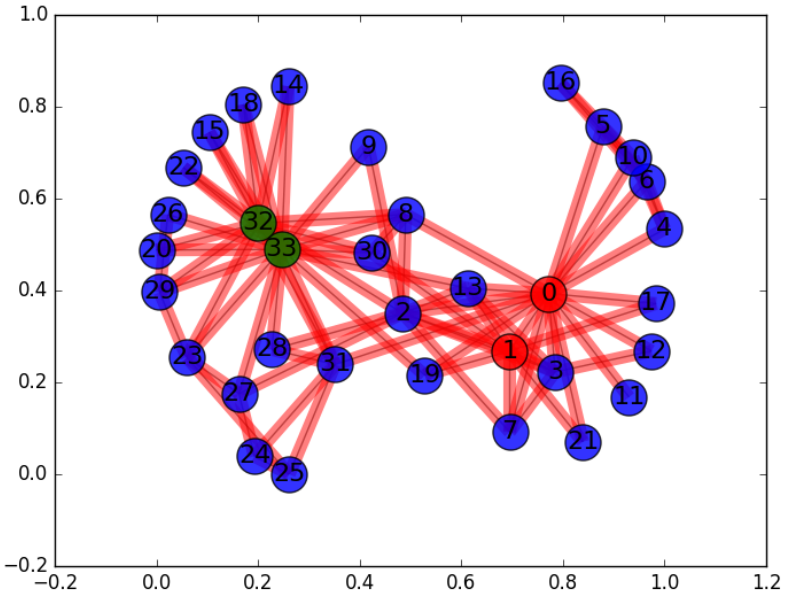
Σχήμα 7: Zachary's Karate club, ParralelPARW, K=2



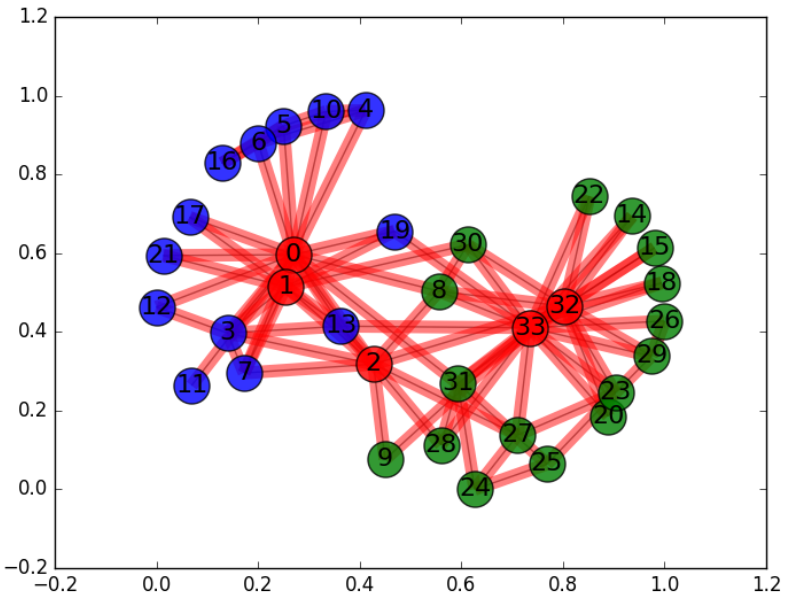
Σχήμα 8: Zachary's Karate club, PARW, K=3



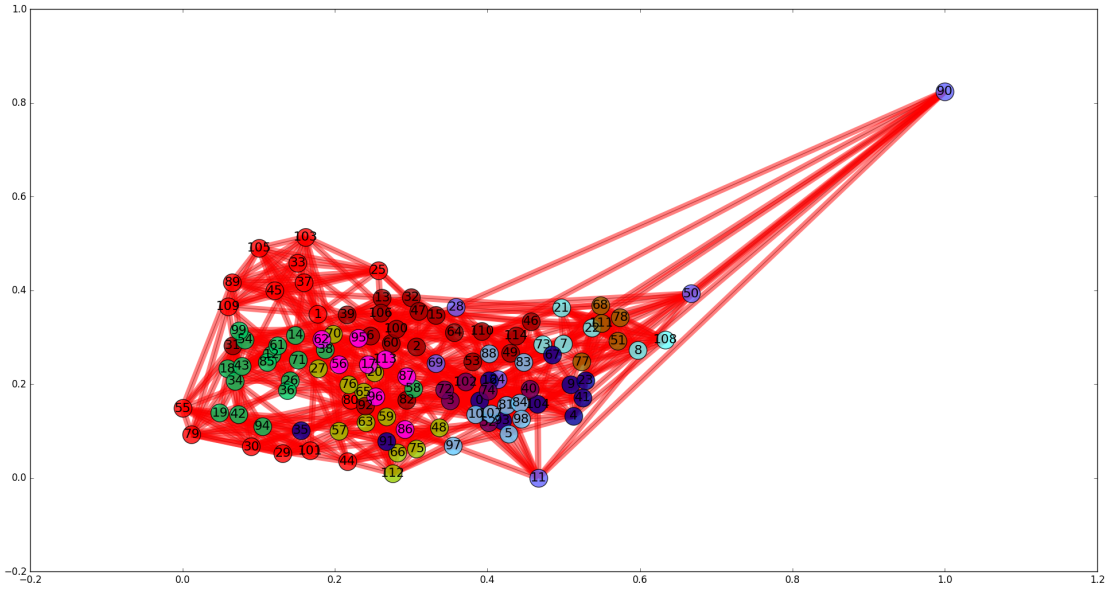
Σχήμα 9: Zachary's Karate club, ParralelPARW, K=3



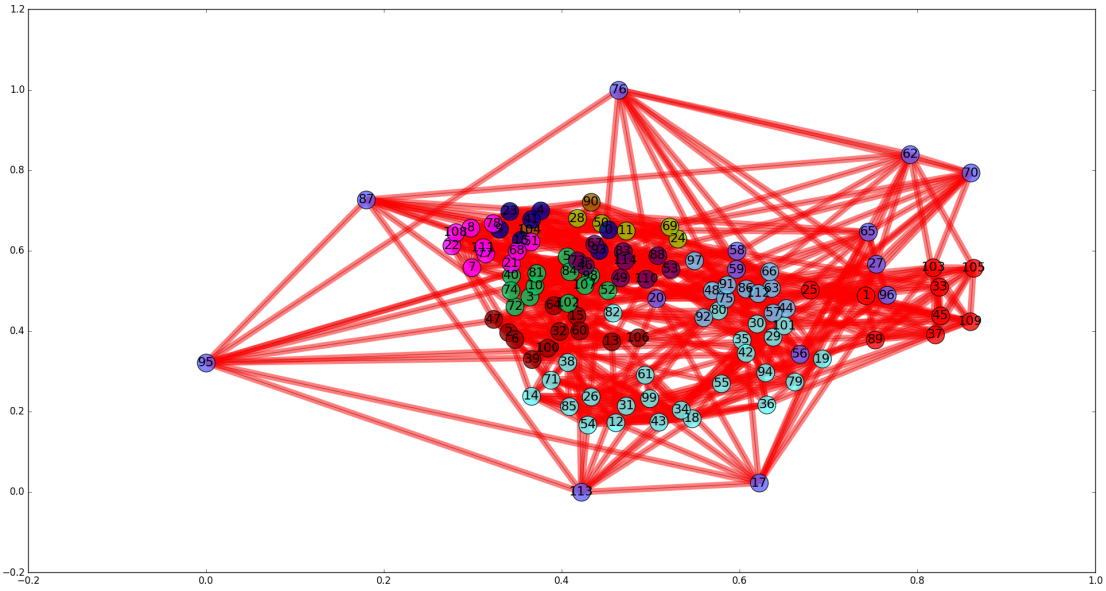
Σχήμα 10: Zachary's Karate club, Kmeans, K=3



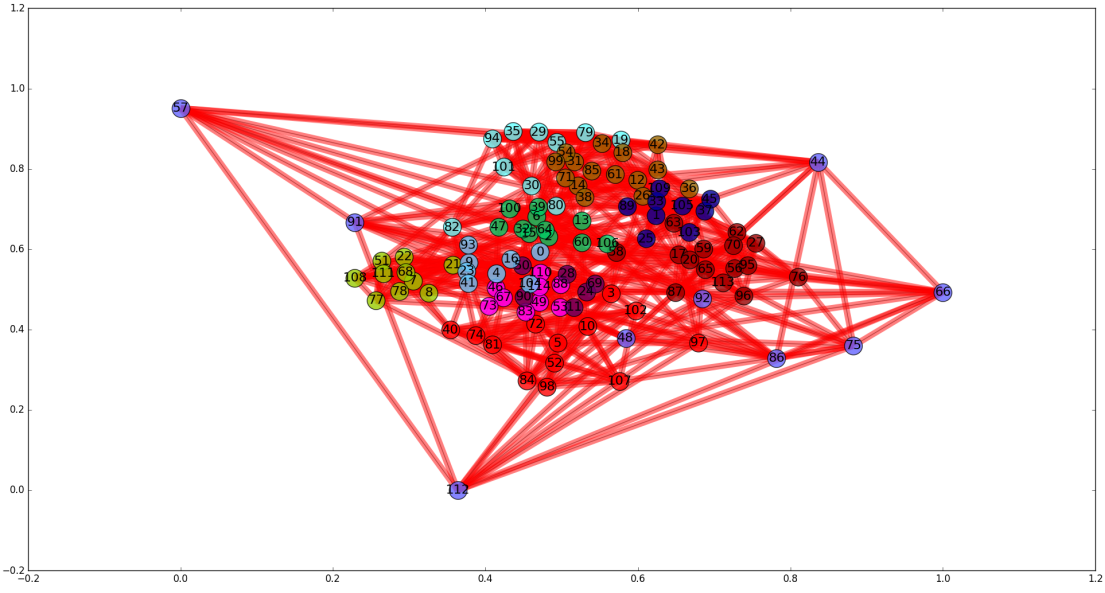
Σχήμα 11: Zachary's Karate club, Spectral, gamma =0.1, K=3



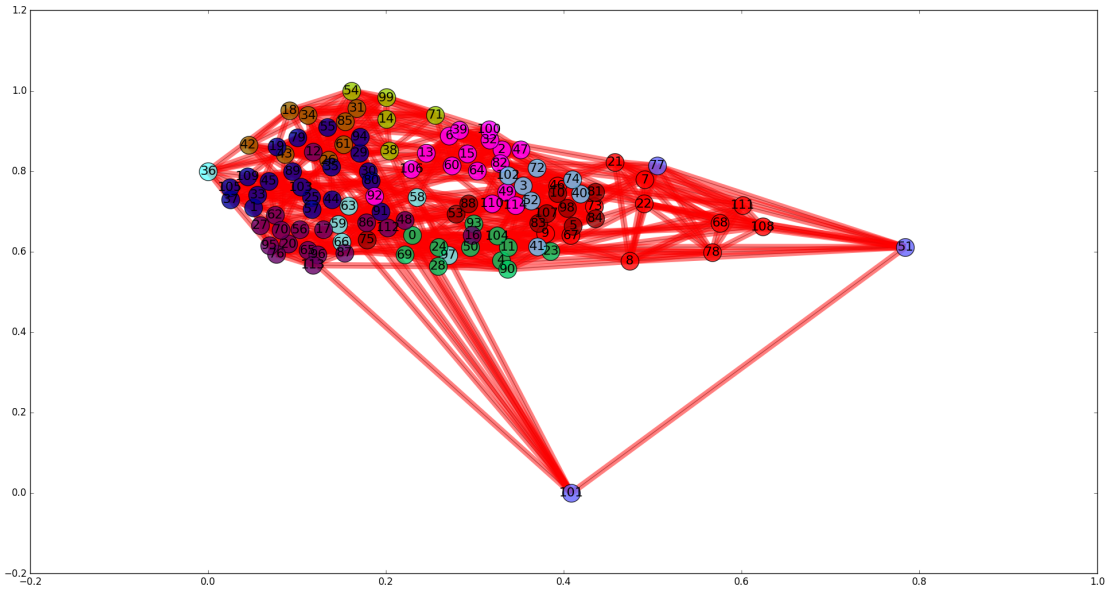
Σχήμα 12: American football graph, K=11, PARW



Σχήμα 13: American football graph, K=11, Kmeans



Σχήμα 14: American football graph, K=11, Spectral, gamma=0.1



Σχήμα 15: American football graph, K=11, Parralel PARW

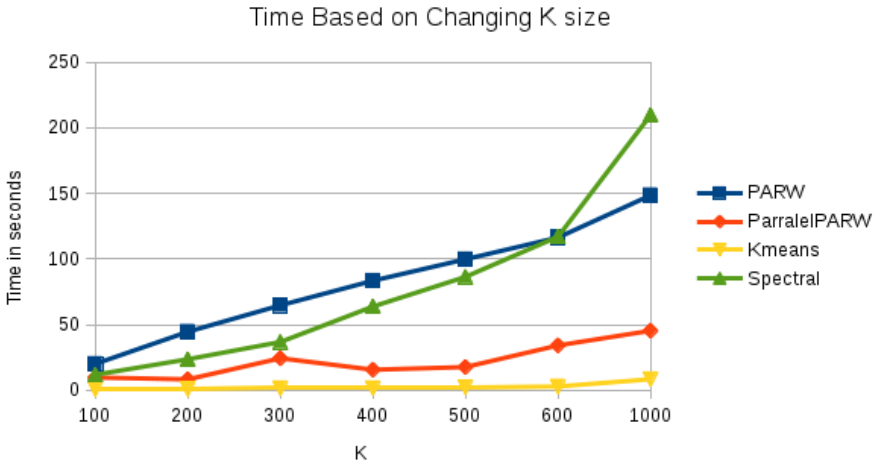
4.2 Χρονική και Χωρική Πολυπλοκότητα

Για να μετρήσουμε την χρονική απόδοση των αλγόριθμων φορτώσαμε ένα μέρος του γράφου *dblp*, τις πρώτες 5000 γραμμές και θεωρήσαμε αυθαίρετα την ύπαρξη 100, 200, 300, 400, 500, 600 και 1000 κέντρων 16 και στο γράφημα ;; για αριθμό κέντρων σταθερό, K=500, αλλάξαμε το πλήθος των γραμμών. Το μηχάνημα στο οποίο λάβαν μέρος τα πειράματα αποτελούνταν απο $8gb$ μνήμη και τον επεξεργαστή $i7-4660$. Υπενθυμίζεται οτι οι δομές που χρησιμοποιούνε οι δύο αλγόριθμοι πέραν του δικού μας, χρησιμοποιούν ποσά μνήμης πολλαπλάσια του $PARW$, τετραπλάσια για τον $Kmeans$, έως και για τριανταπλάσια για τον *spectral* όπως φένεται στο γράφημα 18. Κρατήσαμε το μέγεθος του γράφου σχετικά μικρό ώστε να μην υπάρχει μετάβαση απο τη μνήμη στο σκληρό δίσκο. Η μετάβαση είναι ιδιαίτερα χρονοβόρα και θα αλλοίωνε τα τελικά αποτελέσματα που αφορούν την περίπτωση όπου όλα τα δεδομένα βρίσκονται στη μνήμη, με αυτό το τρόπο μετράμε το χρόνο που απατείται απο τον επεξεργαστή για τους υπολογισμούς μας.

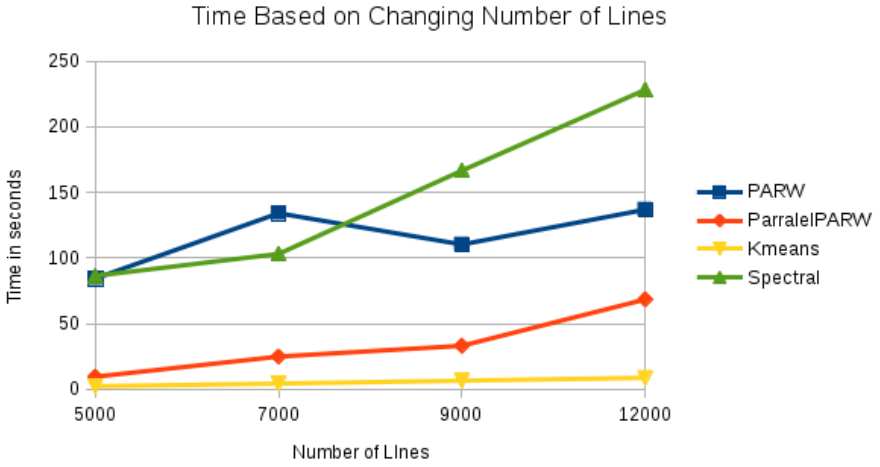
Μεγάλο ενδιαφέρον παρουσιάζει και η χρήση του επεξεργαστή κατα την εκτέλεση του κάθε αλγόριθμου. Για τον $PARW$ 13%, $Parralel\ PARW$ 13%, σπεστραλ [23-100]% στο μεγαλύτερο μέρος εκτέλεσής του ήταν στο 100% και για τον $Kmeans$ απο 13% έως 20%. Αυτό σημαίνει για τον $PARW$ ότι χρησιμοποιώντας το 1/5 σε επεξεργαστική ισχύ σε σχέση με τον *spectral* επιτυγχάνει καλήτερες επιδόσεις. Για τον $Parallel\ PARW$ σημαίνει οτι τα νήματα που χρησιμοποιεί το *Pregel den douleoun* όπως θα έπρεπε. Η παρατήρηση αυτή επιβεβαιώθηκε με τη χρήση πλήθους διαφορετικού αριθμού νημάτων, απο 1 έως 64,

όλα έδωσαν παρόμοια αποτελέσματα. Επομένως η χρονική βελτίωση που φένεται στον *Parallel PARW* οφείλεται στο πώς αντιμετωπίζεται ο κάθε κόμβος απο τη βιβλιοθήκη *pregel*.

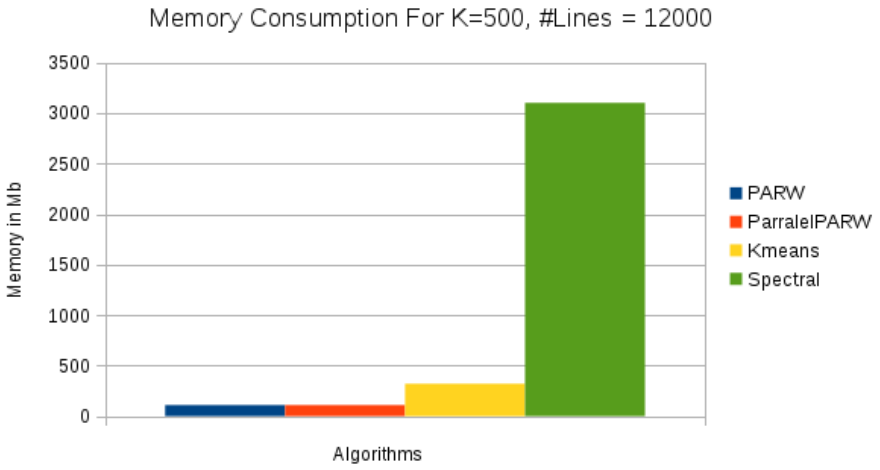
Στο σχήμα 16 παρατίθεται η γραφική αναπαράσταση του χρόνου εκτέλεσης για τον κάθε αλγόριθμο. Παρατηρούμε οτι ο *Kmeans* είναι σαφώς ο ταχύτερος ενώ ακολουθεί ο *Parralel PARW*. Ο *PARW* φένεται να βρίσκεται κοντά στον *spectral* αλλά σαφώς ταχύτερος.



Σχήμα 16: Χρόνος εκτέλεσης, αλλάζοντας το K κάθε φορά



Σχήμα 17: Χρόνος εκτέλεσης, αλλάζοντας το πλήθος των γραμμών κάθε φορά



Σχήμα 18: Κατανάλωση μνήμης για αριθμό κέντρων K=500, και αριθμό γραμμών 12000

5 Συμπεράσματα και προοπτικές

Υλοποιήσαμε έναν αλγόριθμο που απαιτεί τον ορισμό των κέντρων του και δημιουργεί συστάδες με βάση την πιθανότητα ενός κόμβου να απορροφηθεί απο κάποιο κέντρο. Ο αλγόριθμος είναι τουλάχιστον ποιο γρήγορος απο τον *spectral* με παρόμοιες δυνατότητες στο μέτρο που αυτές εξετάστηκαν. Χρησιμοποιώντας τις κατάλληλες δομές επιτυγχάνει ελάχιστη χρήση μνήμης, $O(n)$, εώς και 30 φορές λιγότερη απο το *spectral* στα πειράματα που διεξήχθησαν.

Επίσης δημιουργήσαμε την παράλληλη υλοποίησή του, που φένεται να είναι αρκετές φορές ταχύτερη απο την αρχική. Λόγο περιορισμών της *python* επι της ουσίας λειτούργησε για ένα νήμα και οι διαφορές στο χρόνο υπήρξαν χάρη στο τρόπο που αντιμετωπίζει η βιβλιοθήκη *pregel* τον κάθε κόμβο. Υποθέτουμε οτι η αδελφή της, γραμμένη σε *cpp* θα λειτουργεί με καλύτερο τρόπο. Μια σωστή χρήση των νημάτων εκ μέρους της *pregel*, σε εναν υπολογιστή σαν αυτόν που χρησιμοποιήθηκε για τα πειράματα θα μπορούσε να φέρει βελτίωση με μεγιστο όριο τις 0.8*8= 6.4 φορες, όπου 0.8 η θεωρητική ταχύτητα που δίνεται απο το κάθε νήμα και 8 ο μέγιστος αξιοποιήσιμος αριθμός νημάτων για έναν τετραπύρινο επεξεργαστή με τεχνολογία *hyperthreading*14. Σε μια τέτοια περίπτωση η απόδοση ως προς το χρόνο θα ήταν όμοια με του *Kmeans*.

Τέλος, ενδιαφέρον θα παρουσίαζε η χρήση διαφορετικών μεθόδων αξιολόγησης των πιθανών νέων κέντρων καθώς και αλλαγή στις συνθήκες σύγκλισης του αλγόριθμου.

Αναφορές

[1] *http : //perso.crans.org/aynaud/communities/api.html Βιβλιοθήκη της python για επεξεργασία γράφων σε συνδυασμό με το networkx*

[2] *https : //networkx.github.io*

[3] *W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33, 452-473 (1977).*

[4] *M. Girvan and M. E. J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002)*

[5] *J. Yang and J. Leskovec. Defining and Evaluating Network Communities based on Ground-truth. ICDM, 2012. Ολοκληρομένο εργαλείο επεξεργασίας γράφων*

[6] *http : //scikit − learn.org Σύνολο εργαλειών που αφορούν το machine learning βασισμένοσ στη python*

[7] *http : //www.michaelnielsen.org/ddi/pregel Toy βιβλιοθήκη αντίστοιχη της google για δυνατότητες κλιμακωσιμότητας μεσω πολλαπλών μηχανημάτων στα προβλήματα επεξεργασίας γράφων*

[8] *Βιβλιοθήκη για επεξεργασία γράφων https : //networkx.github.io/*

[9] *Βιβλιοθήκη για επεξεργασία γράφων https : //snap.stanford.edu/snap/*

[10] *http : //www.dcs.bbk.ac.uk/ dell/teaching/cc/paper/sigmod10/p135 − malewicz.pdf*

[11] *https : //en.wikipedia.org/wiki/Unified_Modeling_Language*

[12] *Improving Diversity in Ranking using Absorbing Random Walks https : //pdfs.semanticscholar.org/a45e/8c83e137b5b4490be0046784e9f74ce1216a.pdf*

[13] *A measure of betweenness centrality based on random walks http : //www.sciencedirect.com/science/article/pii/S0378873304000681*

[14] *https : //en.wikipedia.org/wiki/Hyper − threading*