ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολη Εφαρμοσμενων Μαθηματικων & Φυσικων Επιστημων
Τομεας Μαθηματικων

Ανίχνευση Κοινοτήτων σε Γράφους με Χαρακτηριστικά στους Κόμβους

Βασιλεία Κουνή

Επιβλέπων:
Παγουρτζής Αριστείδης
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων:
Πέτρος Ποτίκας
ΕΔΙΠ

Αθήνα

23 Σεπτεμβρίου 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολη Εφαρμοσμενων Μαθηματικων & Φυσικων Επιστημων

Τομεας Μαθηματικων

Διπλωματική Εργασία

Ανίχνευση Κοινοτήτων σε Γράφους με Χαρακτηριστικά στους Κόμβους

Βασιλεία Κουνή

Επιβλέπων:
Παγουρτζής Αριστείδης, Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων:
Πέτρος Ποτίκας, ΕΔΙΠ

Εγκρίθηκε από την εξεταστική επιτροπή την 19η Σεπτεμβρίου 2022

Παγουρτζής Αριστείδης, Καθηγητής Ε.Μ.Π.

Στεφανέας Πέτρος, Αναπληρωτής Καθηγητής Ε.Μ.Π.

Ψαρράκος Παναγιώτης, Καθηγητής Ε.Μ.Π.

Αθήνα, 23 Σεπτεμβρίου 2022

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω προσωπικά τον επιβλέποντα καθηγητή, κο Παγουρτζή Αριστείδη, τον συνεπιβλέποντα, κο Ποτίκα Πέτρο και την συνεργάτιδα τους κα Σούλιου Θεοδώρα, οι οποίοι με την καθοδήγησή και τη εποπτεία τους αποτέλεσαν καταλυτικό παράγοντα στη διεκπεραίωση της έρευνας, που πραγματεύομαι.

Ιδιαίτερες ευχαριστίες θα ήθελα να αποδώσω, επίσης, στο Εθνικό Μετσόβιο Πολυτεχνείο και τους ανθρώπους του, καθηγητές και συμφοιτητές, για την υποστήριξη και τις γνώσεις, που αποκόμισα από αυτούς, καθ' όλη τη διάρκεια των σπουδών μου.

Περίληψη

Στην εργασία αυτή, αναλύθηκε το πρόβλημα ανίχνευσης κοινοτήτων σε γράφους με χαρακτηριστικά κόμβων και ο τρόπος με τον οποίο μπορεί η μέθοδος New Attributed Graph Clustering (NAGC), να δώσει λύση σε αυτό. Πραγματοποιήθηκαν και αξιολογήθηκαν με γνωστές μετρικές, πειράματα σχετικά με την προετοιμασία και την κατάλληλη μετατροπή των δεδομένων και αναλύθηκε το μαθηματικό υπόβαθρο των μεθόδων βελτιστοποίησης που χρησιμοποιήθηκαν. Στόχο της εργασίας αποτελεί, η βελτιστοποίηση της απόδοσης της μεθόδου NAGC, σχετικά με την ακρίβεια των κοινοτήτων που ανιχνεύει, η διερεύνηση εναλλακτικών τρόπων διαχείρισης δεδομένων και ο τρόπος με τον οποίο μπορεί αυτό, να επηρεάσει το τελικό αποτέλεσμα. Στην συνέχεια της εργασίας, παρουσιάζονται αναλυτικά οι μέθοδοι αρχικοποίησης και κανονικοποίησης των δεδομένων, που εισάγονται με την μορφή πινάκων, οι τεχνικές παραγοντοποίησης και συμπλήρωσης των πινάκων αυτών, η διαδικασία βελτιστοποίησης, που χρησιμοποιήθηκε, το είδος και η δομή των δειγμάτων δεδομένων, καθώς και τα αποτελέσματα του τελικού αλγορίθμου, τα οποία εξετάζονται από διαφορετικές σκοπιές. Συμπληρωματικά, στο τέλος του εγγράφου, υπάρχει ο κώδικας, που υλοποιεί την μέθοδο αυτή, αφότου δέχθηκε επεξεργασία, ώστε να εξυπηρετήσει τους σκοπούς της παρούσας εργασίας.

Τα δείγματα δεδομένων και ο αρχικός κώδικας αντλήθηκαν από την διεύθυνση https://github.com/seijimaekawa/NAGC/. Ο επεξεργασμένος κώδικας, τα δείγματα δεδομένων και τα αποτελέσματα των πειραμάτων μπορούν να εντοπιστούν στην διεύθυνση https://github.com/vkouni/NAGC-project. Η πρωτότυπη έρευνα ανήκει στους Seiji Maekawa, Koh Takeuchi και Makoto Onizuka και έχει δημοσιευθεί στην διεύθυνση https://www.jstage.jst.go.jp/article/ipsjjip/28/0/28_427/_pdf.

Abstract

In this thesis, the topics that were analyzed were the problem of community detection in graphs with node attributes and the way the New Attributed Graph Clustering (NAGC) method can provide a solution to this problem. Experiments on preparation and appropriate transformation of data were performed and evaluated with known metrics and the mathematical background of the optimization methods used was analyzed, as well. The aim of the assignment is to optimize the performance of the NAGC method, regarding the accuracy of the communities it detects and also, to investigate alternative ways of data management and how this can affect the final result. Further on this thesis, an analysis is conducted, regarding the initialization and normalization methods performed on the data, which are introduced in the form of matrices, the factorization and completion techniques of these matrices, the optimization process used, the type and structure of the data samples, as well as the results of the final algorithm, which are examined from different points of view. Additionally, the code that implements this method can be found at the end of the document. The code has been edited, in order to serve the purposes of this work.

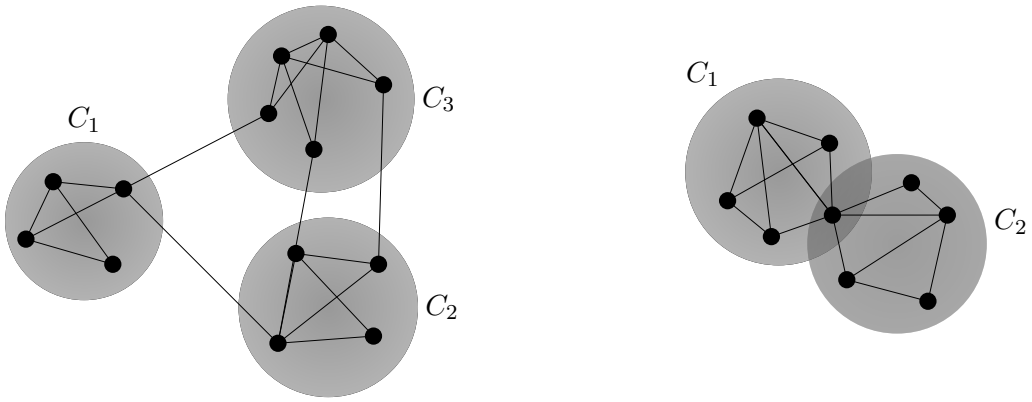
The sample data and code, were derived from the address https://github.com/seijimaekawa/NAGC/. The edited code for the scopes of this thesis, the data sets and the experiment results can be found at https://github.com/vkouni/NAGC-project. The original research has been done by Seiji Maekawa, Koh Takeuchi and Makoto Onizuka and has been published at https://www.jstage.jst.go.jp/article/ipsjjip/28/0/28_427/_pdf.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η πληθώρα δεδομένων στις σύγχρονες μελέτες, έχει οδηγήσει στην χρήση δικτύων, για την απεικόνιση τους, με στόχο την βέλτιστη διαχείριση τους. Η επιλογή αυτή, σε πολλές περιπτώσεις, βασίζεται στις ποικίλες και χρήσιμες ιδιότητες τους, οι οποίες παρέχουν πληροφορίες για τα δεδομένα, σε μορφή, που δεν απαιτεί σημαντικό υπολογιστικό κόστος. Μια τέτοια ιδιότητα, είναι και η ύπαρξη κοινοτήτων κόμβων σε ένα δίκτυο, δηλαδή η σύσταση, εκ κατασκευής, κόμβων σε ομάδες. Θεωρείται πως μια ομάδα, αποτελεί κοινότητα, όταν οι ακμές εντος της ομάδας, είναι πυκνότερες από τις ακμές που συνδέουν τους κόμβους της, με κόμβους εκτός της ομάδας [1]. Οι κοινότητες εμφανίζουν συγκεκριμένα χαρακτηριστικά και έχει αναλυθεί εκτενώς η δομή και ο τρόπος διάκρισης τους ακόμα και σε σύνθετα δίκτυα. Η διαδικασία, αυτή, της διάκρισης, συνήθως, εφαρμόζεται σε μοντελοποιήσεις με μορφή γράφων και ονομάζεται ανίχνευση κοινοτήτων (community detection). Είναι πολλοί οι αλγόριθμοι, που έχουν προταθεί προς αυτή την κατεύθυνση και κατηγοριοποιούνται ανάλογα με την μοροφολογία των γράφων και των δομικών τους στοιχείων (κόμβων, ακμών), που λαμβάνουν ως είσοδο, των πληροφοριών, που αξιοποιούν για την εξαγωγή αποτελεσμάτων, των συμπερασμάτων, που αποσκοπούν να εξάγουν και τελικά τις μεθόδους ανίχνευσης και βελτιστοποίησης, που χρησιμοποιούν, όπως για παράδειγμα αλγορίθμους μηχανικής μάθησης κ.α. Η αξιολόγηση των αποτελεσμάτων πραγματοποιείται μέσω ειδικών μετρικών, οι οποίες όμως, επίσης, διαφοροποιούνται ανάλογα με την περίπτωση. Μια μεγάλη κατηγορία αλγορίθμων αποτελούν και οι αλγόριθμοι, που ανιχνεύουν επικαλυπτόμενες κοινότητες (overlapping communities), δηλαδή ομάδες κόμβων, που είναι μεν διακεκριμένες κοινότητες, αλλά μοιράζονται κόμβους μεταξύ τους, καθώς συσχετίζονται με με τον ίδιο ή παρόμοιο τρόπο.



(α′) Non-overlapping communities

(β′) Overlapping communities

Σχήμα 1.1: Είδη κοινοτήτων

Η ανίχνευση κοινοτήτων στην ξένη βιβλιογραφία, συναντάται συχνά και με την έννοια clustering. Παρότι, πράγματι, στην περίπτωση της διάταξης των δεδομένων σε δομή γράφου, οι δύο έννοιες είναι ταυτόσημες, η τεχνική clustering αποτελεί μια κατηγορία ομαδοποίησης δεδομένων, της μηχανικής μάθησης και επεκτείνεται και σε άλλες συσχετίσεις μεταξύ στοιχείων, πέρα από τις ακμές. Θεωρείται, πως ανήκει στην κατηγορία unsupervised μεθόδων, δηλαδή μεθόδων στις οποίες δεν υπάρχει εξ αρχής καμία γνώση, που να συμβάλλει στην πρόβλεψη του τελικού αποτελέσματος [2]. Σαφώς, ανάλογα με την μορφή των διαθέσιμων δεδομένων, πολλές φορές, αυτή η διαδικασία μπορεί να μετατραπεί σε semi-supervised, ακόμα και supervised learning, ώστε να υπάρχει εξ αρχής μερική ή και πλήρης γνώση για τη τελική πρόβλεψη της συμμετοχής κόμβων σε κοινότητες. Η ανίχνευση κοινοτήτων βρίσκει εφαρμογές στην βιολογία, την ιατρική, την επιστήμη των υπολογιστών, τα κοινωνικά δίκτυα κι άλλους σημαντικούς επιστημονικούς τομείς.

Ένα σημαντικό είδος γράφων είναι και οι γράφοι με χαρακτηριστικά (attributed graphs). Με τον όρο αυτόν, περιγράφονται, συνήθως, γράφοι στους οποίους οι κόμβοι διαθέτουν συγκεκριμένα χαρακτηριστικά, αλλά είναι πολλές οι περιπτώσεις, που τα χαρακτηριστικά αυτά, αφορούν τις ακμές ή και τους κόμβους και τις ακμές, ταυτόχρονα. Οι γράφοι αυτοί είναι μεγάλης σημασίας, καθώς απεικονίζουν με μεγαλύτερη ακρίβεια πραγματικά δίκτυα και παρέχουν περισσότερες πληροφορίες. Για παράδειγμα, στα κοινωνικά δίκτυα οι χρήστες (κόμβοι) διαθέτουν χαρακτηριστικά, τα οποία διαδραματίζουν σημαντικό ρόλο για την διασύνδεση τους (ακμές) με άλλους χρήστες.

Εστιάζοντας στις περιπτώσεις γράφων, όπου οι κόμβοι διαθέτουν συγκεκριμένα χαρακτηριστικά (attributed graphs), φαίνεται, πως η ανίχνευση γίνεται μια αρκετά σύνθετη διαδικασία. Αποτελούν μια ιδιαίτερη κατηγορία ομαδοποίησης και έχουν αναπτυχθεί για αυτήν ειδικά μεγέθη αξιολόγησης, ώστε να διασφαλιστεί η εγκυρότητα τους. Σε πολλές γνωστές μεθόδους, θεωρείται αυθαίρετα, πως ο αριθμός των κοινοτήτων, όπως προκύπτουν από την πυκνότητα των ακμών και την τοπολογική διάταξη των κόμβων στο γράφημα, είναι ίδιος με τον αριθμό των κοινοτήτων, όπως προκύπτουν με βάση την ομοιότητα των χαρακτηριστικών τους. Αυτή η υπόθεση, αποδεικνύεται, σε πολλές περιπτώσεις, πως δεν παράγει ακριβή αποτελέσματα. Ουσιαστικά, η παρουσία των χαρακτηριστικών δεν αξιοποιείται στον μέγιστο βαθμό, αγνοώντας έτσι ένα σημαντικό μέρος της πληροφορίας.

Σχήμα 1.2: Γράφημα με χαρακτηριστικά κόμβων στην μορφή χρωμάτων

Με γνώμονα αυτού του είδους τα προβλήματα, αναπτύχθηκε η μέθοδος New Attributed Graph Clustering (NAGC) από τους Makoto Onizuka, Seiji Maekawa και Koh Takeuch σε πρώτη δημοσίευση το 2018 [3]. Πάνω σε αυτή την έρευνα, βασίζεται και η παρούσα διπλωματική εργασία και παράλληλα γίνεται προσπάθεια βελτίωσης της. Η NAGC χρησιμοποιεί δομές πινάκων για την αποθήκευση και την επεξεργασία των δεδομένων, που δέχεται και παράγει. Λαμβάνει ως είσοδο ένα σύνολο κόμβων, καθώς και τις ακμές που τους συνδέουν. Παράλληλα, ως είσοδο δέχεται και τα χαρακτηριστικά των κόμβων αυτών. Στην συνέχεια, πραγματοποιώντας την κατάλληλη προετοιμασία στα δεδομένα, εφαρμόζονται ειδικές παραγοντοποιήσεις πινάκων και τελικά με την απαιτούμενη επαναληπτική διαδικασία βελτιστοποιήσης καταλήγει στην τελική ανάθεση των κόμβων σε ομάδες. Είναι σημαντικό να αναφερθεί, πως κατά την εκτέλεση του αλγορίθμου, πραγματοποιείται διαχωρισμός μεταξύ του αριθμού των τοπολογικών κοινοτήτων και των κοινοτήτων χαρακτηριστικών. Η μέθοδος αυτή, ανιχνεύει μόνο μη επικαλυπτόμενες κοινότητες.

Για την υλοποίηση της NAGC, χρησιμοποιούνται μερικές διαδεδομένες μέθοδοι της σύγχρονης Ανάλυσης Δεδομένων, χάρις τις οποίες έχουν αναπτυχθεί πολλοί αλγόριθμοι ανίχνευσης κοινοτήτων, όπως οι Nonnegative Matrix Factorization και PU Learning. Στα επόμενα κεφάλαια, γίνεται αναλυτική ανασκόπηση των μεθόδων της NAGC και ο τρόπος που εφαρμόζονται στα προβλήματα, τα οποία πραγματεύεται.

Οι τρόποι με τους οποίους επιδιώκεται η βελτιστοποιήση της μεθόδου NAGC αφορούν δύο βασικά σημεία επεξεργασίας των δεδομένων. Αρχικά, μετατρέπεται η διαδικασία αρχικοποιήσης των δεδομένων και πραγματοποιούνται δοκιμές με γνωστούς και αποδοτικούς αλγορίθμους. Η διαδικασία της αρχικοποίησης είναι ιδιαίτερης σημασίας, αφού μια καλή δομή στα δεδομένα, που θα λάβει ο αλγόριθμος ως είσοδο, θα οδηγήσει τελικά σε πιο άμεσα και πιο ακριβή αποτελέσματα. Συμπληρωματικά, διαφοροποιείται η συνάρτηση, που χρησιμοποιείται για την ομαλοποίηση των στοιχείων ενός βασικού πίνακα συσχέτισης των κοινοτήτων με κριτήριο τα χαρακτηριστικά και αυτών με βάση την τοπολογική διάταξη, στοχεύοντας στην διατήρηση σημαντικών πληροφοριών. Πράγματι, φαίνεται, πως οι μετατροπές αυτές, αποφέρουν και ποιοτική βελτιστοποίηση των αποτελεσμάτων, δηλαδή ακριβέστερη ανάθεση κόμβων σε κοινότητες, αλλά και μείωση του χρονικού κόστους υπολογισμών. Σε επόμενα κεφάλαια, παρουσιάζονται τα αποτελέσματα των συγκρίσεων και αξιολογούνται με κατάλληλα επιλεγμένες μετρικές για το είδος των δεδομένων υπό μελέτη.

Άλλες δημοσιεύσεις, που πραγματεύονται το πρόβλημα της ανίχνευσης κοινοτήτων σε γράφους, στις οποίες έχει παρατηρηθεί καλή απόδοση, βάσει αξιόπιστων μετρικών είναι οι : Pagourtzis κ.ά. [4], Kulkarni κ.ά. [5], Souliou κ.ά. [6], Potikas κ.ά. [7]

Κάνοντας μια προεπισκόπηση της παρούσας εργασίας, σε πρώτη φάση εισάγονται κάποιες βασικές μαθηματικές έννοιες, που αφορούν την NAGC. Στη συνέχεια, παρουσιάζονται οι δύο βασικές μέθοδοι Nonnegative Matrix Factorization και PU Learning και δομείται η τελική συνάρτηση κόστους. Σε αυτή, εφαρμόζονται ειδικοί κανόνες βελτιστοποίησης και αναλύεται το μαθηματικό τους υπόβαθρο. Καταληκτικά, περιγράφεται η πειραματική διαδικασία και ό,τι περιλαμβάνει, καθώς και τα συμπεράσματα, που προκύπτουν από αυτή. Τέλος, παρατίθεται η σχετική βιβλιογραφία και το παράρτημα με τον κώδικα σε Python, καθώς και ένα μέρος των αρχείων των αποτελεσμάτων.

Κεφάλαιο 2

Εισαγωγικές Έννοιες

Στη μέθοδο NAGC, όπως αναφέρθηκε, τα δεδομένα κατανέμονται κατάλληλα σε πίνακες, ώστε να πραγματοποιηθεί η επεξεργασία τους από τις μεθόδους, όπως θα παρουσιαστούν στα επόμενα κεφάλαια. Στο παρόν κεφάλαιο, θα εξηγηθούν οι ιδιότητες των πινάκων αυτών, βασικές μαθηματικές πρακτικές και έννοιες που χρησιμοποιούνται στη διαχείρισή τους.

Λαμβάνεται ως είσοδος ένα γράφημα $G = (V, E)$, όπου $V = \{1, 2, ..., n\}$ το σύνολο των κόμβων και $E = \{(i, j)\} \subseteq [n] \times [n]$ το σύνολο των ακμών.

Τα δεδομένα

Από το γράφημα, που λαμβάνεται ως είσοδος, κατασκευάζονται οι πίνακες : S γειτνίασης και X χαρακτηριστικών. Για τη διαχείριση των δεδομένων και την εξαγωγή αποτελεσμάτων, χρησιμοποιούνται, επίσης, οι πίνακες: U ανάθεσης τοπολογικών ομάδων (topology cluster assignment matrix), V ανάθεσης ομάδων χαρακτηριστικών (attribute factor matrix), H μεταφοράς (cluster assignment transfer matrix), οι οποίοι δεν προκύπτουν από τα δεδομένα του γραφήματος, αλλά αρχικοποιούνται κατάλληλα.

Ο σκοπός

Η βασική επιδίωξη της NAGC είναι η ομαδοποίηση των δεδομένων σε κατάλληλες κοινότητες ή ομάδες, όπως θα αναφέρονται συχνά στην παρούσα εργασία, όπου η ομοιότητα των κόμβων εντός των ομάδων, θα είναι μεγαλύτερη από εκείνη, μεταξύ κόμβων διαφορετικών ομάδων. Ουσιαστικά, επιθυμούμε να καταλήξουμε στην καλύτερη δυνατή εκδοχή του πίνακα ανάθεσης τοπολογικών ομάδων U, ο οποίος δέχεται ταυτόχρονα επιρροή και από την τοπολογική διάταξη του γραφήματος, αλλά και από τα χαρακτηριστικά του εκάστοτε κόμβου.

Η μέθοδος

Για την δόμηση της τελικής συνάρτησης προς βελτιστοποίηση χρησιμοποιούνται, εντός άλλων: η μέθοδος Nonnegative Matrix Factorization και η μέθοδος PU Learning. Για την βελτιστοποίηση, γίνεται χρήση των συνθηκών Karush-Kahn-Tucker (KKT conditions) και πειραματικές μέθοδοι για την εύρεση του καλύτερου συνδυασμού παραμέτρων. Τελικά, πραγματοποιούνται συγκρίσεις και αξιολόγηση των αποτελεσμάτων με κατάληλες μετρικές.

2.1 Βασικές Μαθηματικές Έννοιες

Παρακάτω παρουσιάζονται κάποια σημαντικά μαθηματικά εργαλεία, που χρησιμοποιούνται για την εκτέλεση της μεθόδου.

2.1.1 Στοιχεία Πινάκων

- **Τετραγωνικός Πίνακας (Square Matrix) :** Ένας πίνακας με ίσο αριθμό γραμμών και στηλών ονομάζεται τετραγωνικός πίνακας. [8]
- **Ανάστροφος Πίνακας (Transpose Matrix) :** Ανάστροφος ενός πίνακα A λέγεται ο πίνακας, που έχει γραμμές τις στήλες του A και στήλες τις γραμμές του A. Θα συμβολίζεται ως A^T. [8]
- **Συμμετρικός Πίνακας (Symmetric Matrix) :** Ένας τετραγωνικός πίνακας, που είναι ίσος με τον ανάστροφο του, ονομάζεται συμμετρικός. Θα συμβολίζεται ως A^S. [8]
- **Συζυγής Πίνακας (Conjugate Matrix) :** Συζυγής ενός πίνακα A λέγεται ο πίνακας, που έχει ως στοιχεία του τα μιγαδικά συζυγή των στοιχείων του A. Θα συμβολίζεται ως A^*. Αν τα στοιχεία του A είναι μόνο πραγματικά, τότε $A = A^*$. [9]
- **Αναστροφοσυζυγής Πίνακας (Conjugate Transpose Matrix) :** Αναστροφοσυζυγής συζυγής ενός πίνακα A λέγεται ο ανάστροφος του συζυγούς του A (ή ο συζυγής του ανάστροφου του A). Θα συμβολίζεται ως A^H. Αν τα στοιχεία του A είναι μόνο πραγματικά, τότε $A^H = A^T$. [8]
- **Διαγώνιος Πίνακας (Diagonal Matrix) :** Ένας πίνακας του οποίου όλα τα στοιχεία είναι μηδενικά, εκτός από αυτά της κύριας διαγωνίου του, ονομάζεται διαγώνιος πίνακας. [8]
- **Ερμιτιανός Πίνακας (Hermitian Matrix) :** Ένας τετραγωνικός πίνακας A, που ισούται με τον αναστροφοσυζυγή του, λέγεται ερμιτιανός. Θα συμβολίζεται ως A^+ και ισχύει $A^+ = A^*$. [8]
- **Εσσιανός Πίνακας (Hessian Matrix) :** Είναι ο πίνακας δεύτερων παραγώγων μιας συνάρτησης. Για διπλά παραγωγίσιμη συνάρτηση πολλών μεταβλητών, έστω f, ο εσσιανός πίνακας ορίζεται ως :

$$\nabla^2 f(x) = G(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_N} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & & \\ \vdots & & \ddots & \\ \dfrac{\partial^2 f}{\partial x_N \partial x_1} & & & \dfrac{\partial^2 f}{\partial x_N} \end{bmatrix}$$

  Τα στοιχεία του είναι $G_{i,j} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$. [10]
- **Θετικά Ημιορισμένος Πίνακας (Positive Semidefinite Matrix) :** Ένας ερμιτιανός (συμμετρικός στην περίπτωση πραγματικών πινάκων) πίνακας A, είναι θετικά ορισμένος, εάν όλες οι ιδιοτιμές του είναι μη αρνητικές. [10]

- **Ιδιάζουσες τιμές (Singular Values) :** Οι τιμές αυτές προκύπτουν μέσω της μεθόδου παραγοντοποίησης πινάκων SVD (Singular Value Decomposition) $A = U\Sigma V^*$ στην διαγώνιο του πίνακα Σ. Ισχύει πως οι μη μηδενικές ιδιάζουσες τιμές του B, είναι οι τετραγωνικές ρίζες των μη μηδενικών ιδιοτιμών του A^*A ή του AA^*, δηλαδή $\sigma(A) = \sqrt{A^*A}$ ή $\sigma(A) = \sqrt{AA^*}$. [10]

- **Ίχνος πίνακα (Trace) :** Ίχνος πίνακα (trace) ονομάζεται το άθροισμα των διαγωνίων στοιχείων ενός τετραγωνικού πίνακα και αποτελεί αναλλοίωτη του.

$$Tr(A) = \sum_i a_{ii}$$

Ιδιότητες Ίχνους

* $Tr(ABC) = Tr(CAB) = Tr(BCA)$
* $Tr(A) = Tr(A^T)$
* $Tr(cA) = cTr(A)$
* $Tr(A + B) = Tr(A) + Tr(B)$

Ιδιότητες Παραγώγισης Ίχνους

* $\nabla_X Tr(AX) = A^T$
* $\nabla_X Tr(X^T A) = A$
* $\nabla_X Tr(X^T AX) = (A + A^T)X$
* $\nabla_X Tr(XAX^T) = X(A^T + A)$
* $\nabla_X Tr(X^T XX^T X) = 4XX^T X$

[10]

- **Νόρμες Πινάκων**

  Έστω B πίνακας διάστασης $n \times m$ (μπορεί $n = m$) :

  – **Schatten 1-νόρμα :** Συναντάται συχνά στην βιβλιογραφία ως trace norm ή nuclear norm. Συμβολίζεται $||B||_*$. Αποτελεί νόρμα πινάκων και προκύπτει από τον τύπο :

$$||B||_* = \sum_i^m \sigma_i(B)$$

  , όπου με $\sigma(B)$ συμβολίζονται οι ιδιάζουσες τιμές του πίνακα B. [11]

  – **Schatten 2-νόρμα :** Συναντάται συχνά στην βιβλιογραφία ως Frobenius νόρμα. Συμβολίζεται $||B||_F$. Αποτελεί νόρμα πινάκων/διανυσμάτων και προκύπτει από τον τύπο :

$$||B||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|b_{ij}|^2} \implies ||B||_F = \sqrt{Tr(BB^H)} \quad [11]$$

2.1.2 Κυρτότητα

Έστω συνάρτηση f, συνεχής στο διάστημα Δ και διπλά παραγωγίσιμη στο εσωτερικό του. Πραγματοποιώντας κανείς μελέτη στην συνάρτηση f, μπορεί εύκολα να αποφανθεί για την μονοτονία της, την κυρτότητα της, καθώς και για τα ακρότατα και τα σημεία καμπής της, αν υπάρχουν. Με τον όρο κυρτότητα (convexity), όπως συναντάται στην βιβλιογραφία, αναφερόμαστε σε κυρτές (convex) και κοίλες (concave) συναρτήσεις. Οποιαδήποτε περίπτωση συνάρτησης δεν μπορεί να χαρακτηριστεί με κάποια από τις προαναφερθείσες έννοιες, ορίζεται γενικά ως μη κυρτή (non convex). Στα προβλήματα βελτιστοποίησης, η έννοια της κυρτότητας αποτελεί καταλυτικό παράγοντα, αφού καθορίζει τον τρόπο διαχείρισης και επίλυσης του προβλήματος συνολικά. Πιο συγκεκριμένα, μελετώντας μια κυρτή συνάρτηση, αναμένουμε να καταλήξουμε σε ένα ολικό ακρότατο το οποίο θα είναι μοναδικό, ή στην απόδειξη πως δεν είναι εφικτός ο προσδιορισμός της λύσης. Στην αντίθετη περίπτωση, μελετώντας μια μη κυρτή συνάρτηση, αναμένουμε να συναντήσουμε πολλαπλά τοπικά ακρότατα, ποικιλομορφία στο είδος της κυρτότητας και δυσκολία στην απόδειξη ύπαρξής ή όχι λύσης. Στις σύγχρονες μελέτες, οι περισσότερες συναρτήσεις, που χρησιμοποιούνται, ανήκουν στην κατηγορία των μη κυρτών, εξαιτίας της πολυπλοκότητας τους, η οποία δεν περιορίζει την μοντελοποίηση σύνθετων προβλημάτων. [12]

Παρακάτω παρουσιάζονται κάποιοι βασικοί ορισμοί [13] για την περίπτωση συναρτήσεων πολλών μεταβλητών, που αποτελούν το βασικό εργαλείο στη σύγχρονη ανάλυση δεδομένων.

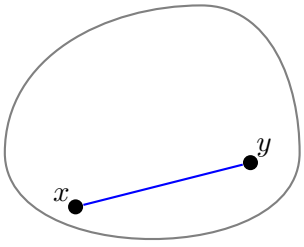
- **Κυρτό σύνολο :** Ένα σύνολο S n-διανυσμάτων, ονομάζεται κυρτό αν

$$(1-\lambda)x + \lambda y \in S$$

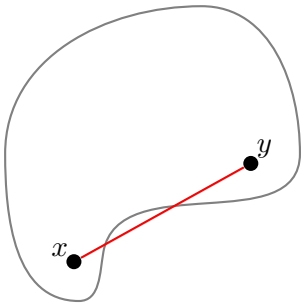
, για κάθε $x \in S$, $y \in S$ και $\lambda \in [0,1]$.

  - Για $n = 1$, ο ορισμός συμπίπτει με τον ορισμό του διαστήματος.
  - Για $n = 2$, κάθε δύο σημεία του συνόλου θα πρέπει να μπορούν να ενωθούν από μία ευθεία γραμμή, η οποία θα πρέπει να περιέχεται πλήρως στο σύνολο.

Στο Σχήμα 2.1α′ παρουσιάζεται ένα παράδειγμα κυρτού συνόλου. Στο Σχήμα 2.1β′ παρουσιάζεται ένα παράδειγμα μη κυρτού συνόλου.



(α′) Κυρτό σύνολο



(β′) Μη κυρτό σύνολο

Σχήμα 2.1: Κυρτό — Μη κυρτό σύνολο

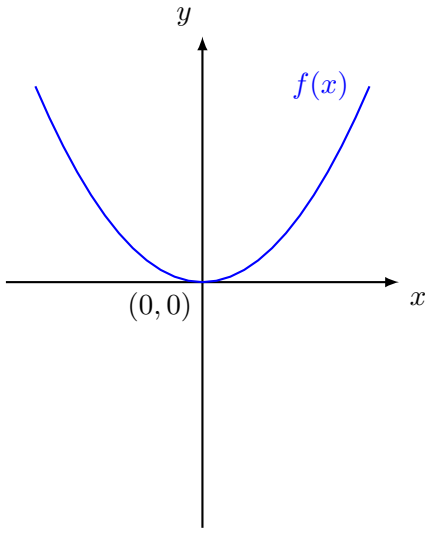
- **Κυρτή Συνάρτηση :** Έστω f συνάρτηση πολλών μεταβλητών ορισμένη σε ένα σύνολο S. Τότε η f είναι :
  - **κυρτή**, αν για κάθε $x \in S$ και $y \in S$ και για κάθε $\lambda \in (0,1)$ ισχύει

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

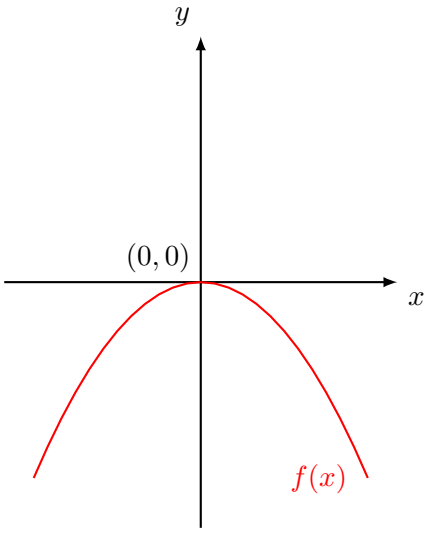
    Στο Σχήμα 2.2α΄ παρουσιάζεται ένα παράδειγμα κυρτής συνάρτησης.
  - **κοίλη**, αν για κάθε $x \in S$ και $y \in S$ και για κάθε $\lambda \in (0,1)$ ισχύει

$$f((1-\lambda)x + \lambda y) \geq (1-\lambda)f(x) + \lambda f(y)$$

    Στο Σχήμα 2.2β΄ παρουσιάζεται ένα παράδειγμα κοίλης συνάρτησης.



(α΄) Κυρτή συνάρτηση



(β΄) Κοίλη συνάρτηση

Σχήμα 2.2: Κυρτή — Κοίλη συνάρτηση

- **Κυρτό πρόβλημα βελτιστοποίησης :** Ως κυρτό πρόβλημα βελτιστοποίησης ορίζεται κάθε πρόβλημα, στο οποίο όλοι οι περιορισμοί και η βασική σχέση προς βελτιστοποιήση είναι κυρτές συναρτήσεις. [14]
- **Μη κυρτό πρόβλημα βελτιστοποίησης :** Ως μη κυρτό πρόβλημα βελτιστοποίησης ορίζεται κάθε πρόβλημα, στο οποίο τουλάχιστον ένας από περιορισμούς ή η βασική σχέση προς βελτιστοποίηση δεν είναι κυρτές συναρτήσεις. [14]

Βασικό ερώτημα αποτελεί, ο τρόπος με τον οποίο μπορεί κανείς, να καταλήξει σε συμπεράσματα σχετικά με την κυρτότητα μιας συνάρτησης. Παρακάτω παρουσιάζονται κάποιες βασικές αποδεδειγμένες προτάσεις προς αυτή την κατεύθυνση [15].

- **Γραφική απεικόνιση :** Μια συνάρτηση f πολλών μεταβλητών ορισμένη σε ένα κυρτό σύνολο S είναι :
  - **κυρτή**, αν και μόνον αν το σύνολο επί και άνω της γραφικής της παράστασης $\{(x, y) : x \in S \text{ και } y \geq f(x)\}$ είναι κυρτό.
  - **κοίλη**, αν και μόνον αν το σύνολο επί και υπό της γραφικής της παράστασης $\{(x, y) : x \in S \text{ και } y \leq f(x)\}$ είναι κυρτό.

- **Ανισότητα Jensen :** Μια συνάρτηση f πολλών μεταβλητών, ορισμένη σε ένα κυρτό σύνολο S είναι :
  - **κυρτή**, αν και μόνον αν για κάθε $n \geq 2$

    $$f(\lambda_1 x_1 + \cdots + \lambda_n x_n) \geq \lambda_1 f(x_1) + \cdots + \lambda_n f(x_n)$$

    ,για κάθε $x_1 \in S, \ldots, x_n \in S$ και για κάθε $\lambda_1 \geq 0, \ldots, \lambda_n \geq 0$ με $\sum_{i=1}^{n} \lambda_i = 1$.
  - **κοίλη**, αν και μόνον αν για κάθε $n \leq 2$

    $$f(\lambda_1 x_1 + \cdots + \lambda_n x_n) \geq \lambda_1 f(x_1) + \cdots + \lambda_n f(x_n)$$

    ,για κάθε $x_1 \in S, \ldots, x_n \in S$ και για κάθε $\lambda_1 \geq 0, \ldots, \lambda_n \geq 0$ με $\sum_{i=1}^{n} \lambda_i = 1$.

- **Κριτήριο Πρώτης Παραγώγου :** Μια παραγωγίσιμη συνάρτηση f πολλών μεταβλητών ορισμένη, σε ένα κυρτό σύνολο S είναι :
  - **κυρτή**, αν και μόνον αν

    $$f(x) - f(y) \geq \sum_{i=1}^{n} f_i'(y)\dot{(}x_i - y_i)$$

    , για κάθε $x \in S$ και $y \in S$.
  - **κοίλη**, αν και μόνον αν

    $$f(x) - f(y) \leq \sum_{i=1}^{n} f_i'(y)\dot{(}x_i - y_i)$$

    , για κάθε $x \in S$ και $y \in S$.

- **Κριτήριο Δεύτερης Παραγώγου - Θεώρημα Εσσιανής :** Μια διπλά παραγωγίσιμη συνάρτηση f πολλών μεταβλητών, ορισμένη σε ένα κυρτό σύνολο S είναι :
  - **κυρτή**, αν και μόνον αν ο εσσιανός πίνακας της συνάρτησης είναι θετικά ημιορισμένος.
  - **κοίλη**, αν και μόνον αν ο εσσιανός πίνακας της συνάρτησης είναι θετικά ημιορισμένος.

2.1.3 Μη γραμμικές συναρτήσεις

Ως μη γραμμική χαρακτηρίζεται οποιαδήποτε συνάρτηση, η οποία κατά την απεικόνιση της γραφικά, δεν έχει την μορφή ευθείας γραμμής στο πεδίο ορισμού της. Οι συναρτήσεις αυτές, χρησιμοποιούνται συχνά στις σύγχρονες εφαρμογές, καθώς μπορούν να περιγράψουν τα μη τεχνητά προβλήματα, που διέπουν την φύση. Παράλληλα, στον τομέα της Πληροφορικής και συγκεκριμένα στην ανάλυση των δεδομένων και στην μηχανική μάθηση οι μη γραμμικές συναρτήσεις διαδραματίζουν σημαίνοντα ρόλο. Για παράδειγμα, στην δημιουργία ενός νευρωνικού δικτύου επιλέγονται κάποιες μη γραμμικές συναρτήσεις, οι οποίες ονομάζονται συναρτήσεις ενεργοποίησης (activation functions) και καθορίζουν τον τρόπο εκπαίδευσης του μοντέλου και το τελικό αποτέλεσμα της πρόβλεψης τους [16]. Επίσης, σε πολλές περιπτώσεις, όπως και στην περίπτωση της NAGC, οι συναρτήσεις αυτές χρησιμοποιούνται για την αναδιαμόρφωση των μεγεθών, ώστε να διαχειριζόμαστε συγκρίσιμα στοιχεία (θα συμβολίζεται με f). Για τους σκοπούς της παρούσας μελέτης, θα αναλυθούν δύο γνωστές περιπτώσεις συναρτήσεων : η σιγμοειδής συνάρτηση [17] (sigmoid function) και η συνάρτηση ReLU (Rectified Linear Activation function) [16].

- **Σιγμοειδής Συνάρτηση (Sigmoid function) :** Έχει πάρει το όνομα της από την μορφολογία της, καθώς όταν απεικονίζεται γραφικά, έχει την μορφή του αγγλικού "S". Συχνά, ως σιγμοειδή συνάρτηση αναφερόμαστε στον τύπο :

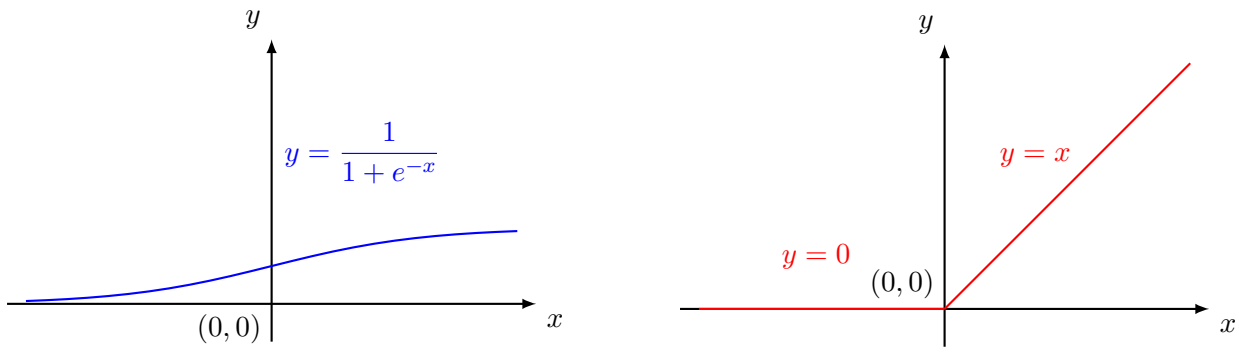
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

  Στο Σχήμα 2.3α′ παρουσιάζεται ένα μέρος της σιγμοειδούς συνάρτησης.

- **Συνάρτηση Διορθωμένης Γραμμικής Μονάδας (Rectified Linear Activation function - ReLU) :** Είναι μια μερικώς ή τμηματικά γραμμική συνάρτηση και κατηγοριοποιείται στις μη γραμμικές συναρτήσεις. Λαμβάνει ως είσοδο έναν πραγματικό αριθμό και επιστρέφει την τιμή του, μόνο αν είναι μεγαλύτερος του μηδενός, διαφορετικά επιστρέφει την τιμή μηδέν. Περιγράφεται από τον παρακάτω τύπο :

$$f(x) = \max\{0, x\} \tag{2.2}$$

  Στο Σχήμα 2.3β′ παρουσιάζεται ένα μέρος της συνάρτησης ReLU.



(α′) Σιγμοειδής συνάρτηση

(β′) Συνάρτηση ReLu

Σχήμα 2.3: Σιγμοειδής συνάρτηση — Συνάρτηση ReLu

2.2 Πίνακες

Οι πίνακες παρουσιάζονται αναλυτικά παρακάτω :

Ο πίνακας γειτνίασης S

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix}$$

- Γραμμές : Κόμβοι
- Στήλες : Κόμβοι
- $\in \mathbb{R}_{+}^{n \times n}$

Ένας τετραγωνικός πίνακας του οποίου τα στοιχεία υποδεικνύουν, εάν ζεύγη κόμβων είναι γειτονικά ή όχι στο δεδομένο γράφημα. Εάν είναι, εισάγεται το 1 (ή το βάρος της ακμής σε σταθμισμένα γραφήματα) στο σχετικό κελί και 0 σε κάθε άλλη περίπτωση. Είναι, συνήθως, διαγώνιος (αν δεν υπάρχουν self loops) και σε περίπτωση μη κατευθυνόμενων γραφημάτων, αυτός ο πίνακας είναι, επίσης, συμμετρικός. Από τον πίνακα S προκύπτει ο πίνακας κάλυψης W, για τον οποίο ισχύει $w_{ij} = 0$, αν $s_{ij} = 0$ και $w_{ij} = 1$, αν $s_{ij} \neq 0$.

Ο πίνακας χαρακτηριστικών X

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

- Γραμμές : Κόμβοι
- Στήλες : Χαρακτηριστικά
- $\in \mathbb{R}_{+}^{n \times m}$

Ένας πίνακας χωρίς συγκεκριμένη μορφολογία στα δεδομένα του. Τα στοιχεία του αντιπροσωπεύουν τη σχέση μεταξύ των κόμβων του γραφήματος και των χαρακτηριστικών που μπορεί να έχουν. Εάν ένας κόμβος αντιπροσωπεύεται από ένα χαρακτηριστικό, εισάγεται 1 (ή το αντίστοιχο βάρος) στο σχετικό κελί ή 0 σε κάθε άλλη περίπτωση. Ένας κόμβος μπορεί να έχει περισσότερα από ένα χαρακτηριστικά.

Ο πίνακας ανάθεσης τοπολογικών ομάδων U

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1k_1} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nk_1} \end{bmatrix}$$

- Γραμμές: Κόμβοι
- Στήλες: Τοπολογικές κοινότητες
- $\in \mathbb{R}_+^{n \times k_1}$

Ένας πίνακας χωρίς συγκεκριμένη μορφολογία στα δεδομένα του. Τα στοιχεία του αντιπροσωπεύουν την τάση ενός κόμβου να ανήκει σε μία ή περισσότερες κοινότητες, όπως αυτές προκύπτουν με βάση την τοπολογία του γραφήματος.

Ο πίνακας ανάθεσης ομάδων χαρακτηριστικών V

$$V = \begin{bmatrix} v_{11} & \cdots & v_{1k_2} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mk_2} \end{bmatrix}$$

- Γραμμές: Χαρακτηριστικά
- Στήλες: Κοινότητες με βάση τα χαρακτηριστικά
- $\in \mathbb{R}_+^{m \times k_2}$

Ένας πίνακας χωρίς συγκεκριμένη μορφολογία στα δεδομένα του. Τα στοιχεία του αντιπροσωπεύουν την τάση ενός χαρακτηριστικού να ανήκει σε μία ή περισσότερες κοινότητες, όπως αυτές προκύπτουν με βάση τα χαρακτηριστικά των κόμβων.

Ο πίνακας μεταφοράς H

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1k_2} \\ \vdots & \ddots & \vdots \\ h_{k_1 1} & \cdots & h_{k_1 k_2} \end{bmatrix}$$

- Γραμμές: Τοπολογικές κοινότητες
- Στήλες: Κοινότητες με βάση τα χαρακτηριστικά
- $\in \mathbb{R}_+^{k_1 \times n}$

Ένας πίνακας χωρίς συγκεκριμένη μορφολογία στα δεδομένα του. Τα στοιχεία του αντιπροσωπεύουν τη συσχέτιση μεταξύ των κοινοτήτων, που προκύπτουν με βάση την τοπολογία του γραφήματος και εκείνων, που προκύπτουν με βάση τα χαρακτηριστικά των κόμβων.

Κεφάλαιο 3

Η προετοιμασία των δεδομένων

Με την πάροδο των ετών τα δεδομένα προς διαχείριση τείνουν να αυξάνονται σε μέγεθος και σε πολυπλοκότητα και οι σύγχρονες μέθοδοι ανάλυσης καλούνται, συνεχώς, να προσαρμόζονται στις νέες αυτές απαιτήσεις.

Για να επιτύχουμε τα καλύτερα δυνατά αποτελέσματα, καθώς και μικρούς χρόνους εκτέλεσης στην ανάλυση των δεδομένων, είναι απαραίτητο να φροντίζουμε πως τα δεδομένα, που πρόκειται να δώσουμε ως είσοδο σε έναν αλγόριθμο, είναι όσο το δυνατόν περισσότερο βελτιστοποιημένα. Κατά την συλλογή των δεδομένων, είναι αναμενόμενο, πως η μορφή τους δεν θα είναι πάντα κατάλληλη για την άμεση αξιοποίηση τους, με αποτέλεσμα να πρέπει να υποβληθούν σε κάποια προεργασία. Προς αυτή την κατεύθυνση έχουν αναπτυχθεί διάφορες μέθοδοι προετοιμασίας, κατάλληλες για κάθε διαφορετική περίπτωση δεδομένων. Σε σύγχρονες μεθόδους, όπως η NAGC, συχνά, παρατηρούμε, εντός άλλων μεθόδων προετοιμασίας των δεδομένων και την κανονικοποίηση (normalization), που αφορά την μορφοποίηση της κλίμακας των τιμών των δεδομένων σε ένα συγκεκριμένο εύρος. Έτσι, όλα τα στοιχεία, που απαρτίζουν το εκάστοτε δείγμα δεδομένων βρίσκονται σε κοινή κλίμακα, είναι συγκρίσιμα και επηρεάζουν με όμοιο τρόπο το τελικό αποτέλεσμα. Παράλληλα, με τον τρόπο αυτό, οι αποστάσεις μεταξύ των στοιχείων παραμένουν αναλλοίωτες, κάτι το οποίο καθίσταται αναγκαίο, όταν ο αλγόριθμος ανάλυσης βασίζεται σε κάποια τέτοια μετρική [18].

Σαφώς, αυτού του είδους η προεργασία πάνω στα δεδομένα δεν είναι πάντα απαραίτητη (πχ. όταν τα δεδομένα έχουν ήδη μια κοινή κλίμακα ή ακολουθούν μια γνωστή κατανομή). Η κανονικοποίηση μπορεί να πραγματοποιηθεί με την χρήση γνωστών μεθόδων, όπως η Min-Max Scalar, ή μέσω προσαρμοσμένων κάθε φορά συναρτήσεων στα εκάστοτε δεδομένα. [19]

Στα πλαίσια της προετοιμασίας των δεδομένων, πολλές φορές, καλούμαστε, επίσης, να αρχικοποιήσουμε παραμέτρους, καθώς δεν είναι εφικτό να γίνουν δεκτές από τους αλγορίθμους ανάλυσης χωρίς κάποια αρχική τιμή. Σε αυτές τις περιπτώσεις, επιλέγεται συχνά να δίνονται τυχαίες τιμές οι οποίες, όμως, τελικά διαδραματίζουν πολύ σημαντικό ρόλο στην εξαγωγή αποτελεσμάτων, αλλά και στον χρόνο εκτέλεσης των υπολογισμών.

Στην NAGC χρειάζεται να εκτελεστεί αρχικοποίηση στο περιεχόμενο των πινάκων U,V και H. Όπως αναφέρθηκε, η NAGC αναπαράγει μια μη κυρτή συνάρτηση και τα προβλήματα βελτιστοποίησης αυτού του είδους, δεν μπορούν να προσεγγίσουν στην επίλυση τους ολικά ακρότατα, αλλά πολλά διαφορετικά τοπικά. Καταλαβαίνουμε, λοιπόν, πως η αρχικοποίηση των δεδομένων, που θα δεχτεί ο αλγόριθμος, συμβάλλει σημαντικά στην σύγκλιση και στο χρονικό κόστος. Στην αρχική δημοσίευση γίνεται αρχικοποίηση των πινάκων U και V με την μέθοδο k-means (k-means++), η οποία εφαρμόζεται στον πίνακα X. Με την ίδια λογική, προστέθηκαν επιπλέον οι αλγόριθμοι : Agglomerative Clustering με τρία διαφορετικά linkages, Spectral Clustering και DBSCAN, ώστε να εξεταστεί η συμπεριφορά και τα αποτελέσματα (ποιοτικά και χρονικά) του αλγορίθμου σε διαφορετικές αρχικοποιήσεις.

Τέλος, η χρήση ομαδοποίησης στα δεδομένα πριν την εφαρμογή του βασικού αλγορίθμου, εντάσσεται στα πλαίσια βελτιστοποίησης των δεδομένων, καθώς με την κατηγοριοποίηση που πραγματοποιείται, πολλά σημεία, που αποτελούν "θόρυβο" στα δεδομένα και συνεπώς δεν συγκαταλέγονται σε κάποια κοινότητα, τελικά εξαιρούνται από τα δεδομένα υπό μελέτη. Στην περίπτωση της NAGC, που αποτελεί μέθοδο ομαδοποίησης, αυτού του είδους η αρχικοποίηση προσφέρει πολλαπλά οφέλη στην απόδοση της.

3.1 k-means clustering

Η μέθοδος k-means αποτελεί μια από τις πιο γνωστές μεθόδους ανίχνευσης κοινοτήτων. Σύμφωνα με την πηγή [20], σαν σκοπό έχει τη δημιουργία k διαφορετικών ομάδων, οι οποίες θα περιέχουν περίπου ισάριθμα σημεία του δείγματος. Κάθε ομάδα εκπροσωπείται από ένα σημείο, το οποίο ονομάζεται κεντροειδές (centroid). Εξηγώντας με περισσότερη λεπτομέρεια, ένα κεντροειδές σημείο είναι η μέση τιμή όλων των σημείων, τα οποία ανήκουν στην εκάστοτε ομάδα, που αυτό αντιπροσωπεύει και έτσι με την προσθήκη νέων σημείων, η τιμή του ανανεώνεται βάσει της παρακάτω σχέσης :

$$C_i = \frac{1}{||S_i||} \sum_{x_j \in S_i} x_j \tag{3.1}$$

, όπου

- C_i: το i-οστό κεντροειδές
- S_i: όλα τα σημεία που ανήκουν στην ομάδα με κεντροειδές το C_i
- x_j: το j-οστό σημείο της ομάδας με κεντροειδές το C_i
- $||S_i||$: ο αριθμός των σημείων στην ομάδα με κεντροειδές το C_i

Η βασική ιδέα της k-means είναι να εντοπίσει k κεντροειδή σημεία και κάθε σημείο του δείγματος να ανήκει σε κάποιο από τα k σύνολα, που αυτά αντιπροσωπεύουν. Κριτήριο αποτελεί η Ευκλείδεια μετρική, καθώς για κάθε σημείο πραγματοποιείται η μέτρηση της Ευκλείδειας απόστασης από κάθε ένα από τα κεντροειδή και το σημείο μεταφέρεται στην αντίστοιχη ομάδα, από της οποίας το κεντροειδές είχε την μικρότερη απόσταση. Συνεπώς, η συνάρτηση κόστους είναι η :

$$C_1, C_2, ..., C_k = \text{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - Ci||^2 \tag{3.2}$$

Το πρόβλημα αυτό θεωρείται NP-hard και για την επίλυσή του χρησιμοποιείται ο αλγόριθμος του Lloyd [21]. Ο αλγόριθμος αυτός, ουσιαστικά, αποτελεί μια προσεγγιστική επαναληπτική διαδικασία με τα εξής βήματα :

1. Επιλέγεται ένας αριθμός k - ο αριθμός των ομάδων.
2. Επιλέγονται τυχαία k κεντροειδή από τα σημεία του δείγματος.
3. Για κάθε σημείο του δείγματος, υπολογίζεται η ευκλείδεια απόσταση μεταξύ του σημείου και όλων των κεντροειδών. Το σημείο ανήκει στην ομάδα με το κοντινότερο κεντροειδές.

4. Η τιμή του κεντροειδούς ανανεώνεται με την νέα μέση τιμή.

5. Τα βήματα 3 και 4 επαναλαμβάνονται μέχρι να επιτευχθεί κάποιο κριτήριο τερματισμού ή σύγκλιση.

Η επίτευξη της σύγκλισης, δηλαδή, είτε τα κεντροειδή να μην ανανεώνονται σε νέες τιμές πλέον, είτε τα σημεία, που ανήκουν σε μία ομάδα να παραμένουν ίδια, είναι μια αρκετά χρονοβόρα διαδικασία και συνήθως αποφεύγεται. Πιο συχνά, συναντάμε κριτήρια τερματισμού, όπως προκαθορισμένο αριθμό επαναλήψεων ή προκαθορισμένη τιμή απόστασης των σημείων από το κεντροειδές τους. Τα κριτήρια τερματισμού θα πρέπει να επιλέγονται προσεκτικά, καθώς μπορεί να οδηγήσουν σε κακής ποιότητας αποτελέσματα.

3.1.1 k-means++ clustering

Ανάμεσα στα πολλά πλεονεκτήματα της μεθόδου k-means συναντάμε και κάποια μειονεκτήματα, όπως το ότι τα αρχικά κεντροειδή επιλέγονται τυχαία και φαίνεται, πως καθορίζουν σε μεγάλο βαθμό το τελικό αποτέλεσμα. Την λύση σε αυτό το πρόβλημα δίνει η μέθοδος k-means++. Η k-means++ είναι μια μέθοδος αρχικοποίησης της k-means, όπου τα κεντροειδή δεν επιλέγονται με τυχαίο τρόπο, αλλά με την εξής διαδικασία :

1. Επιλέγεται ένα σημείο, έστω c_1 ως το πρώτο κεντροειδές από το δείγμα με τυχαίο τρόπο.

2. Συγκρίνεται η απόσταση όλων των σημείων του δείγματος με το σημείο c_1 και επιλέγεται το σημείο με την μεγαλύτερη απόσταση, έστω x_1.

3. Το σημείο x_1 γίνεται το επόμενο κεντροειδές.

4. Η διαδικασία επαναλαμβάνεται, έως ότου εντοπιστούν τα k κεντροειδή.

5. Εφαρμόζονται τα βήματα $3-5$ της k-means.

Εξαιτίας του πρώτου βήματος, όπου επιλέγεται τυχαία το πρώτο κεντροειδές, η μέθοδος δεν είναι ντετρμινιστική και για τα ίδια δεδομένα ενδέχεται, κάθε φορά να δίνει διαφορετικά αποτελέσματα. [22]

3.2 Agglomerative Clustering

Η κλάση αλγορίθμων Agglomerative ανήκει στην κατηγορία της ιεραρχικής ομαδοποίησης (hierarchical clustering). Πιο συγκεκριμένα, οι αλγόριθμοι αυτοί, ακολουθούν την λογική "bottom-up", αφού ξεκινούν από μεγάλο αριθμό ομάδων καταλήγοντας σε μία και θεωρείται πως βασίζονται σε αποστάσεις, ως μέτρο ομοιότητας (distanse-based algorithm). Γενικά, ένας agglomerative (συσσωρευτικός) αλγόριθμος αποτελεί μια επαναληπτική διαδικασία και παρουσιάζει πολλά πλεονεκτήματα σε σχέση με άλλες μεθόδους ομαδοποίησης, όπως για παράδειγμα το ότι δεν είναι απαραίτητο να δίνεται ως είσοδος ο αριθμός των ομάδων ή να πραγματοποιείται κάποια τυχαία αρχικοποίηση στα δεδομένα (δηλαδή, κάθε φορά, που καλείται αυτή η μέθοδος για τα ίδια δεδομένα, πρόκειται να καταλήξει στα ίδια αποτελέσματα). Στα μειονεκτήματα της θα πρέπει να αναφερθεί, πως έχει παρατηρηθεί χαμηλή απόδοση σε μεγάλου μεγέθους δείγματα.
Παρακάτω παρουσιάζονται τα βήματα ενός τέτοιου είδους αλγορίθμου :

1. Για κάθε σημείο του δείγματος ορίζεται μια ομάδα.
2. Κατασκευάζεται ένας πίνακας εγγύτητας (proximity matrix), διάστασης $n \times n$, όπου καταγράφονται οι αποστάσεις όλων των σημείων/ομάδων μεταξύ τους. Η μετρική απόστασης μπορεί να επιλεχθεί σύμφωνα με τις ανάγκες του δείγματος.
3. Επιλέγονται κάθε φορά οι μικρότερες τιμές/αποστάσεις του πίνακα εγγύτητας και τα εκάστοτε σημεία συμπτύσσονται σε μία κοινή ομάδα. Στη συνέχεια, ο πίνακας εγγύτητας ανανεώνεται με κατάλληλο τρόπο με χρήση του εκάστοτε linkage, όπως θα εξηγηθεί παρακάτω, βάσει των νέων αλλαγών.
4. Το βήμα 3 επαναλαμβάνεται μέχρι την δημιουργία μιας ενιαίας ομάδας.

Όπως αναφέρθηκε, η μέθοδος αυτή βασίζεται στις αποστάσεις των ομάδων μεταξύ τους. Όταν βρισκόμαστε στο αρχικό στάδιο, οι αποστάσεις μεταξύ των ομάδων μπορούν να υπολογιστούν με γνωστές μετρικές συναρτήσεις. Όταν, όμως, έχουν προστεθεί στις ομάδες επιπλέον σημεία, δεν μπορεί να εφαρμοστεί η ίδια τεχνική για τον υπολογισμό της απόστασης. Για τον λόγο αυτό, χρησιμοποιούνται κάποιες συναρτήσεις που ονομάζονται linkages. Παρακάτω παρουσιάζονται οι βασικές μορφές των linkages, που χρησιμοποιήθηκαν και στην αρχικοποίηση των δεδομένων της NAGC.

- **Complete linkage** : Επιστρέφει την τιμή της απόστασης μεταξύ των πιο μακρινών σημείων δύο ομάδων.
- **Average linkage** : Επιστρέφει την μέση τιμή των αποστάσεων μεταξύ όλων των σημείων δύο ομάδων.
- **Single linkage** : Επιστρέφει την τιμή της απόστασης μεταξύ των πιο κοντινών σημείων δύο ομάδων.

Εξαιτίας της ιεραρχικής δομής που κατασκευάζεται από αυτού του είδους τους αλγορίθμους, μπορεί να γίνει χρήση δενδρογράμματος για την απεικόνιση των αποτελεσμάτων ανά βήμα. Αν επιλέξουμε, να μην δώσουμε σαν είσοδο στον αλγόριθμο τον αριθμό των ομάδων, θα πρέπει να δώσουμε ένα όριο (threshold) το οποίο, αν εφαρμοστεί στο δενδρόγραμμα, είναι εφικτό να προβλέψει τελικά τον αριθμό των ομάδων. [23]

3.3 Spectral Clustering

Η κλάση αλγορίθμων Spectral Clustering αποτελεί βασική τενική ομαδοποίησης και οι ρίζες της εδρεύουν στην θεωρία των γραφημάτων. Η φιλοσοφία του Spectral Clustering αφορά, την διαμόρφωση ομάδων κόμβων με βάση τις ακμές που τους συνδέουν. Παρά την άμεση συσχέτιση με δεδομένα σε μορφή γράφων, είναι εξίσου λειτουργική και σε δεδομένα που δομούνται με άλλες μορφές [24]. Ένας Spectral αλγόριθμος αξιοποιεί το σύνολο των ιδιοτιμών (spectrum) ειδικών πινάκων, οι οποίοι προκύπτουν από τον γράφο ή από το αντίστοιχο σύνολο δεδομένων υπό μελέτη, με σκοπό την ταξινόμησή των κόμβων στις κατάλληλες ομάδες. Αξίζει να σημειωθεί, πως έχει παρατηρηθεί αρκετά γρήγορη απόδοση σε αραιά (sparse) και οποιουδήποτε σχήματος/δομής δεδομένα. Βέβαια, δεν μπορεί να παραληφθεί, πως ο υπολογισμός των ιδιοτιμών και ιδιοδιανυσμάτων είναι μια χρονοβόρα διαδικασία για μεγάλα δείγματα δεδομένων και επίσης, πως ο αριθμός των ομάδων θα πρέπει να προκαθορίζεται κάθε φορά. [25] Όπως αναφέρθηκε, χρησιμοποιούνται κάποιοι πίνακες για την εξαγωγή των αποτελεσμάτων. Παρακάτω παρουσιάζονται οι σχετικοί πίνακες :
Έστω γράφημα $G = (V, E)$ ένας απλός, μη κατευθυνόμενος γράφος και $V = \{v_1, v_2, \ldots, v_n\}$ το σύνολο των κόμβων.

Πίνακας Γειτνίασης (Adjacency Matrix)
Είναι ο $|V| \times |V|$ τετραγωνικός πίνακας, που ορίζεται ως :

$$A_{ij} = \begin{cases} 1, \text{ αν υπάρχει ακμή μεταξύ } v_i \text{ και } v_j \\ 0, \text{ σε κάθε άλλη περίπτωση} \end{cases}$$

Σε απλά γραφήματα τα στοιχεία της διαγωνίου του πίνακα αυτού είναι μηδενικά. [26]

Πίνακας Βαθμών (Degree Matrix)
Είναι ο $|V| \times |V|$ τετραγωνικός, διαγώνιος πίνακας, που ορίζεται ως :

$$D_{ij} = \begin{cases} deg(v_i), \text{ αν } i = j \\ 0, \text{ σε κάθε άλλη περίπτωση} \end{cases}$$

, όπου $deg(v_i)$ ο αριθμός των ακμών που καταλήγουν στον κόμβο v_i. [26]

Πίνακας Ομοιότητας (Affinity/Similarity Matrix)
Είναι ένας $|V| \times |V|$ τετραγωνικός πίνακας, έστω A' και τα στοιχεία του εκφράζουν την ομοιότητα μεταξύ των κόμβων. Τα μέτρα ομοιότητας, που μπορούν να χρησιμοποιηθούν είναι ποικίλα (π.χ απόσταση, ομοιότητά συνημιτόνου (cosine similarity) κ.α.) και η επιλογή του κατάλληλου εξαρτάται από την μορφή των δεδομένων. Ισχύει, πως για πλήρως όμοιους κόμβους αναμένουμε την τιμή 1 και για ανόμοιους την τιμή 0. Οι τιμές αυτές, διαδραματίζουν τον ρόλο βάρους στις ακμές και συχνά ερμηνεύονται ως πιθανότητες συσχέτισης μεταξύ κόμβων. Την θέση του μπορεί να πάρει ο πίνακας Γειτνίασης ($A = A'$). Σε απλά γραφήματα, τα στοιχεία

της διαγωνίου του πίνακα αυτού είναι μηδενικά. Πολλές φορές, για τυχαίες κατανομές δεδομένων κρίνεται απαραίτητη η δημιουργία γράφων ομοιότητας με γνωστές μεθόδους όπως k-nearest neihbor graphs κ.α. και μεταγενέστερα μέσα από αυτούς τους γράφους μπορεί να προκύψει και ο πίνακας ομοιότητας. [27]

Λαπλασιανός Πίνακας Γράφου (Graph Laplacian Matrix)
Είναι ένας $|V| \times |V|$ συμμετρικός, θετικά ημιορισμένος πίνακας, έστω L και ορίζεται από την διαφορά μεταξύ του πίνακα Βαθμών και του πίνακα Ομοιότητας :

$$L = D - A' \tag{3.3}$$

Στην διαγώνιο του συναντάμε τους βαθμούς του εκάστοτε κόμβου. Στην βιβλιογραφία παρουσιάζονται συχνά παραλλαγές του Λαπλασιανού πίνακα (Normalized Laplacian, Generalized Laplacian, Relaxed Laplacian κ.α.), που ουσιαστικά διαφέρουν στην κανονικοποίηση των δεδομένων και την διαχείριση του πίνακα A' και επιλέγονται με γνώμονα την μορφή των δεδομένων. [26]

- **Spectral Gap :** Η μικρότερη μη μηδενική ιδιοτιμή του πίνακα L. Δίνει πληροφορία για την πυκνότητα του γράφου.
- **Fiedler Value :** Η δεύτερη ιδιοτιμή του πίνακα L. Το αντίστοιχο ιδιοδιάνυσμα ονομάζεται ιδιοδιάνυσμα Fiedler. Η ιδιοτιμή Fiedler προσεγγίζει την ελάχιστη τομή του γράφου (minimum graph cut), που χρειάζεται για να χωριστεί σε δύο μέρη. Κάθε στοιχείο του ιδιοδιανύσματος Fiedler μας δίνει πληροφορίες για το μέρος του γράφου (μεταξύ των δύο), στο οποίο βρίσκεται ο εκάστοτε κόμβος.

Κανονικοποιημένος Λαπλασιανός Πίνακας Γράφου (Normalized Graph Laplacian Matrix)
Είναι ένας πίνακας, ο οποίος ορίζεται [28] ως :

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{3.4}$$

$$L_{rw} = D^{-1} L \tag{3.5}$$

Ο πρώτος ονομάζεται L_{sym}, καθώς είναι συμμετρικός και ο δεύτερος L_{rw}, καθώς μπορεί να παρομοιαστεί με έναν τυχαίο περίπατο (random walk). Οι δύο αυτοί πίνακες έχουν πολλές κοινές ιδιότητες και είναι ισχυρά συνδεδεμένοι. Ο κανονικοποιημένος Λαπλασιανός πίνακας προτιμάται συχνά σε εφαρμογές, καθώς διατηρεί πολλές κοινές ιδιότητες με τον Λαπλασιανό πίνακα και παράλληλα δεν επηρεάζεται από τους βρόχους (self loops), που μπορεί να υπάρχουν στους γράφους.

Για τον πίνακα L_{sym} [29] :

$$D^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{\sqrt{\deg(v_1)}} & 0 & \dots & 0 \\ 0 & \frac{1}{\sqrt{\deg(v_2)}} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sqrt{\deg(v_n)}} \end{bmatrix} \quad D^{\frac{1}{2}} = \begin{bmatrix} \sqrt{\deg(v_1)} & 0 & \dots & 0 \\ 0 & \sqrt{\deg(v_2)} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\deg(v_n)} \end{bmatrix}$$

Τα στοιχεία του πίνακα L_{sym} προσδιορίζονται με τον εξής τρόπο :

$$L_{ij}^{sym} = \begin{cases} 1 & , \text{ αν } i = j \text{ και } \deg(v_i) \neq 0 \\ \frac{-1}{\sqrt{\deg(v_i)\deg(v_j)}} & , \text{ αν } i \neq j \text{ και } v_i \text{ συνδέεται με } v_j \\ 0 & , \text{σε κάθε άλλη περίπτωση} \end{cases}$$

Για τον πίνακα L_{rw} :

$$D^{-1} = \begin{bmatrix} \frac{1}{\deg(v_1)} & 0 & \dots & 0 \\ 0 & \frac{1}{\deg(v_2)} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\deg(v_n)} \end{bmatrix}$$

Τα στοιχεία του πίνακα L_{rw} προσδιορίζονται με τον εξής τρόπο :

$$L_{ij}^{rw} = \begin{cases} 1 & , \text{ αν } i = j \text{ και } \deg(v_i) \neq 0 \\ \frac{-1}{\deg(v_i)} & , \text{ αν } i \neq j \text{ και } v_i \text{ συνδέεται με } v_j \\ 0 & , \text{σε κάθε άλλη περίπτωση} \end{cases}$$

Έστω γράφημα G, μη κατευθυνόμενο με μη αρνητικά βάρη. Η πολλαπλότητα k της ιδιοτιμής 0 των Λαπλασιανών πινάκων(L, L_{sym} και L_{rw}) υποδεικνύει τον αριθμό των συνεκτικών συνιστωσών του γράφου, έστω $A_1, A_2, \dots A_k$ [26].

Έχουν προταθεί πολλές παραλλαγές αλγορίθμων ανάλογα και με τις διαφορετικές παραλλαγές του Λαπλασιανού Πίνακα. Για την αρχικοποίηση της μεθόδου NAGC χρησιμοποιήθηκε ο Κανονικοποιημένος Λαπλασιανός Πίνακας. Παρακάτω παρουσιάζονται οι δύο βασικοί αλγόριθμοι, που χρησιμοποιούν αυτού του είδους πίνακα. Δίνεται σαν είσοδος ο αριθμός k των ομάδων.

Normalized spectral clustering - Shi and Malik (2000)

1. Κατασκευάζεται ο πίνακας Ομοιότητας A' ή τίθεται $A' = A$.
2. Υπολογίζεται ο πίνακας Βαθμών D.
3. Υπολογίζεται ο Λαπλασιανός πίνακας L (μη κανονικοποιημένος).
4. Υπολογίζονται τα πρώτα k ιδιοδιανύσματα $(u_1, u_2, \ldots, u_k)$ του γενικευμένου προβλήματος ιδιοτιμών $Lu = \lambda Du$.
5. Έστω πίνακας $U \in \mathbb{R}^{n\times k}$ με στήλες τα διανύσματα $u_1, u_2, \ldots, u_k$.
6. Για $i = 1, \ldots, n$, έστω $y_i \in \mathbb{R}^k$ να είναι το διάνυσμα που αντιστοιχεί στην i-οστή γραμμή του πίνακα U.
7. Εφαρμόζεται η μέθοδος k-means για τα σημεία $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ και παράγονται τελικά οι ομάδες $C_1, \ldots, C_k$.
8. Καταλήγουμε στις ομάδες $A_1, \ldots, A_k$ με $A_i = \{j | y_j \in C_i\}$.

Τα ιδιοδιανύσματα που προκύπτουν από την επίλυση του γενικευμένου προβλήματος ιδιοτιμών είναι επίσης και ιδιοδιανύσματα του πίνακα L_{rw}. Συνεπώς, ο αλγόριθμος των Shi and Malik χρησιμοποιεί ιδιοδιανύσματα κανονικοποιημένου Λαπλασιανού πίνακα και για αυτόν τον λόγο παρουσιάζεται ως Normalized spectral clustering τεχνική.

$$L_{rw}u = \lambda u \overset{L_{rw}=D^{-1}L}{\Longrightarrow} D^{-1}Lu = \lambda u \implies DD^{-1}Lu = \lambda Du \implies Lu = \lambda Du$$

Συγκεκριμένα, ο πίνακας L_{rw}, συχνά προτιμάται αντί των L_{sym} και L στις περιπτώσεις που οι βαθμοί των ακμών του γράφου υπό μελέτη είναι κατανεμημένοι σε ένα μεγάλο εύρος. Στην περίπτωση που το μεγαλύτερο μέρος των κόμβων έχει σχετικά κοινούς βαθμούς, όλοι οι Λαπλασιανοί πίνακες φαίνεται να είναι παρόμοιοι και οι μεταβολές στα αποτελέσματα από την επιλογή κάποιου εκ των άλλων πινάκων είναι μικρή. Επίσης, για μεγάλα δείγματα δεδομένων φαίνεται να λειτουργεί πιο αποδοτικά, αφού δεν χρειάζεται να πραγματοποιηθεί καμία μετατροπή υπάρχοντος πίνακα. Τέλος, τα ιδιοδιανύσματα που παράγει, είναι άμεσα διανύσματα ένδειξης κοινότητας. (cluster indicator vectors). [28]

Normalized spectral clustering - Ng, Jordan, and Weiss (2002)

1. Κατασκευάζεται ο πίνακας Ομοιότητας A' ή τίθεται $A' = A$.
2. Υπολογίζεται ο πίνακας Βαθμών D.
3. Υπολογίζεται ο Κανονικοποιημένος Λαπλασιανός πίνακας L_{sym}.
4. Υπολογίζονται τα πρώτα k ιδιοδιανύσματα $(u_1, u_2, \ldots, u_k)$ του L_{sym}.
5. Έστω πίνακας $U \in \mathbb{R}^{n\times k}$ με στήλες τα διανύσματα $u_1, u_2, \ldots, u_k$.

6. Έστω πίνακας $T \in \mathbb{R}^{n \times k}$, που ορίζεται μέσω του U, κανονικοποιώντας τις γραμμές (νόρμα 1) με τον εξής τρόπο :

$$t_{ij} = \frac{u_{ij}}{(\sum_k u_{ik}^2)^{1/2}} \tag{3.6}$$

7. Για $i = 1, \ldots, n$, έστω $y_i \in \mathbb{R}^k$ να είναι το διάνυσμα, που αντιστοιχεί στην i-οστή γραμμή του πίνακα T.

8. Εφαρμόζεται η μέθοδος k-means για τα σημεία $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ και παράγονται τελικά οι ομάδες $C_1, \ldots, C_k$.

9. Εξάγονται οι ομάδες $A_1, \ldots, A_k$ με $A_i = \{j | y_j \in C_i\}$. [28]

3.4 DBSCAN

Η μέθοδος ομαδοποίησης Density-Based Spatial Clustering of Applications with Noise (DBSCAN) ανήκει στην κατηγορία των αλγορίθμων ομαδοποίησης, που βασίζονται στην πυκνότητα των κοινοτήτων κόμβων (desnsity-based) ενός δείγματος. Αυτό το είδος αλγορίθμων έχει αποδειχθεί, πως λειτουργεί αποδοτικά ακόμη και σε περιπτώσεις δειγμάτων με τυχαία κατανομή παρατηρήσεων ή με μεγάλο πληθυσμό παρατηρήσεων σε μεγάλη απόσταση από την πλειοψηφία, οι οποίες προκαλούν σφάλματα στα τελικά αποτελέσματα και αποτελούν θόρυβο. Σύμφωνα με την μέθοδο DBSCAN, ένας κόμβος ανήκει σε μία ομάδα, αν η απόσταση του από έναν μεγάλο αριθμό κόμβων της ομάδας είναι μικρή. Η απόσταση αυτή, όπως και σε πολλές άλλες μεθόδους, συνήθως προσδιορίζεται με την ευκλείδεια μετρική. Στην DBSCAN κρίνεται απαραίτητο εξ αρχής, να προσδιοριστούν ως παράμετροι, αρχικά η μέγιστη απόσταση στην οποία δύο κόμβοι θεωρούνται γειτονικοί (eps) και τον ελάχιστο αριθμό κόμβων, που μπορούν να συντελέσουν μια κοινότητα (minPts). Με βάση αυτές τις παραμέτρους, οι κόμβοι ενός δείγματος διαφοροποιούνται σε τρεις κατηγορίες :

- **Core points :** Οι κόμβοι που περιβάλλονται από τουλάχιστον minPts κόμβους (συμπεριλαμβανομένου του εαυτού τους) σε ακτίνα ίση με eps.
- **Border points :** Οι κόμβοι που είναι γειτονικοί με κάποιο core point, αλλά ο αριθμός των κόμβων που τους περιβάλλουν είναι λιγότεροι από minPts (συμπεριλαμβανομένου του εαυτού τους).
- **Outliers :** Οι κόμβοι που δεν είναι core points και δεν είναι γειτονικοί με κανένα core point.

Καταληκτικά, ο αλγόριθμος λειτουργεί με τον εξής τρόπο :

1. Ορίζονται οι παράμετροι (eps,minPts).
2. Επιλέγεται τυχαία ένα σημείο και ελέγχεται η γειτονία του με βάση το προκαθορισμένο μέγεθος eps. Εφόσον, το σημείο περιβάλλεται από τουλάχιστον minPts κόμβους (συμπεριλαμβανομένου του εαυτού τους) σε ακτίνα ίση με eps, θεωρείται core point. Με αυτόν τον τρόπο, ξεκινάει η δημιουργία της πρώτης ομάδας με στοιχεία την πρώτη γειτονιά που εντοπίστηκε. Αν τα στοιχεία αυτά της γειτονιάς αποτελούν, επίσης, core points, συμπεριλαμβάνεται στην υπάρχουσα ομάδα και η δική τους γειτονιά κ.ο.κ. Στην περίπτωση που το πρώτο σημείο, που επιλέχθηκε τυχαία δεν αποτελεί core point, αφήνεται ως θόρυβος και επιλέγεται κάποιο άλλο, ώσπου να βρεθεί το πρώτο core point.
3. Στην συνέχεια, επιλέγεται τυχαία επόμενος κόμβος ανάμεσα σε αυτούς, που δεν έχουν επιλεγεί (συμπεριλαμβανομένων αυτών, που έχουν θεωρηθεί ως θόρυβος) και εφαρμόζεται το Βήμα 2.
4. Η διαδικασία ολοκληρώνεται, όταν έχουν επιλεγεί όλοι οι κόμβοι. [30]

Κεφάλαιο 4

Nonnegative Matrix Factorization

Η μέθοδος Nonnegative Matrix Factorization (NMF) αποτελεί μια ευρέως γνωστή μέθοδο παραγοντοποίησης και μείωσης διαστάσεων πινάκων (Linear Dimentionality Reduction – LDR). Εφαρμόζεται υπό την προϋπόθεση, πως τα στοιχεία των πινάκων, που δέχεται ως είσοδο, είναι μη αρνητικά και οι παράγοντες πίνακες της έχουν, επίσης, αυτή την ιδιότητα. Αποτελεί σημαντικό εργαλείο στην ανάλυση πολυδιάστατων δεδομένων, εξαιτίας αυτής της μη αρνητικότητας των δεδομένων και, συνεπώς, εύκολης απεικόνισης και ερμηνείας των αποτελεσμάτων της και ανά τα χρόνια έχει συντελέσει σε πολλές και σημαντικές επιστημονικές προσεγγίσεις καίριων ζητημάτων.

Μία από τις πρώτες μορφές της μεθόδου διαμορφώθηκε στις αρχές της δεκαετίας του 1990, με την έννοια της παραγοντοποίησης θετικών πινάκων (Positive Matrix Factorization - PMF), παρότι μελέτες έχουν πραγματοποιηθεί από το 1970. Εξελίχθηκε σε μέθοδο για μη αρνητικούς πίνακες από τους Paatero και Tapper το 1994 και μεταγενέστερα καθιερώθηκε ανάμεσα σε άλλες σημαντικές μεθόδους μέσω της έρευνας των Lee και Seung το 2000 [31], στην οποία πραγματοποιήθηκε ανάλυση των ιδιοτήτων της και προτάθηκαν κάποιοι αλγόριθμοι παραγοντοποίησης, που εφαρμόζονται μέχρι και σήμερα. Η χρησιμότητα της NMF σε διάφορους κλάδους των μαθηματικών και της επιστήμης των υπολογιστών (αστρονομία, ομαδοποίηση εγγράφων/εικόνων, επεξεργασία ακουστικών σημάτων, συστήματα συστάσεων κ.α.), έχει οδηγήσει σε πολλές και διαφορετικές παραλλαγές της, όπως οι SNMF, NMTF κ.α., οι οποίες θα αναλυθούν στη συνέχεια.

Οι λύσεις, που παράγει, δεν είναι μοναδικές και αντιμετωπίζεται ως ένα μη κυρτό πρόβλημα, γι αυτό και γίνεται χρήση κανόνων ανανέωσης σε κάθε βήμα, με σκοπό την επίτευξη της σύγκλισης στο καλύτερο δυνατό αποτέλεσμα. Η μέθοδος ολοκληρώνεται με το πέρας κάποιου προκαθορισμένου αριθμού επαναλήψεων και ο χρόνος, που χρειάζεται, εξαρτάται σε μεγάλο βαθμό από την μορφή των δεδομένων, που δέχεται κάθε φορά ως είσοδο. Συνεπώς, τις περισσότερες φορές τα δεδομένα εισόδου προετοιμάζονται κατάλληλα πριν από την επεξεργασία. Σαν πρόβλημα χαρακτηρίζεται πως ανήκει στην κλάση NP-hard και μπορεί να επιλυθεί μόνο προσεγγιστικά, με χρήση ευριστικών (heuristic) αλγορίθμων [32]. Η NMF σαν μέθοδος είναι αρκετά ευαίσθητη στην αρχικοποίηση των δεδομένων, που δέχεται ως είσοδο.

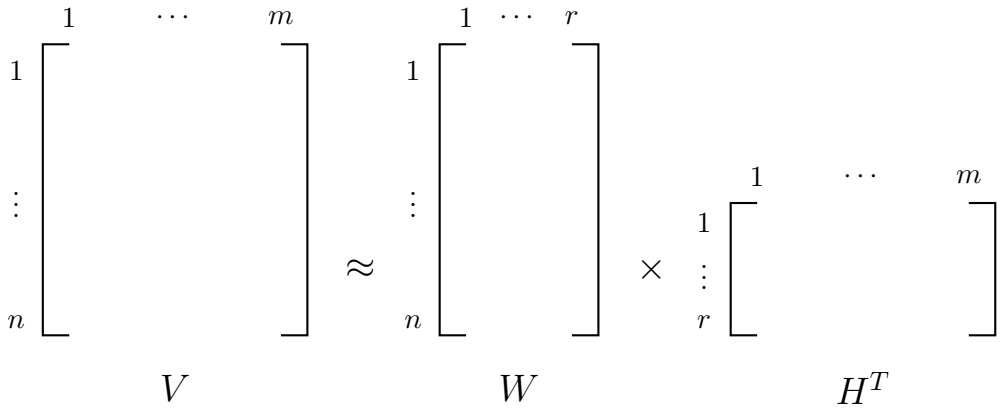
Στην NAGC, χρησιμοποιείται μια παραλλαγή για την ανάλυση του πίνακα X (πίνακας κοινοτήτων - χαρακτηριστικών) και του πίνακα S (πίνακας γειτνίασης), κατά μία έννοια σε απλούστερους παράγοντες πίνακες. Παράλληλα, η μέθοδος NMF πραγματοποιεί φυσικά μια κατηγοριοποίηση των δεδομένων εξαιτίας της δομής των παραγόμενων πινάκων της, όπως θα δούμε στην συνέχεια. Τέλος, η κλασική μορφή της NMF μπορεί να εφαρμοστεί αποδοτικά, μόνο σε γραμμικά δεδομένα και για αυτόν τον λόγο αναπτύχθηκαν τρόποι βελτίωσης της μεθόδου. [31]

4.1 NMF (Nonnegative Matrix Factorization)

Δεδομένου μη αρνητικού πίνακα V διάστασης $m \times n$, η μέθοδος NMF προσεγγίζει δύο μη αρνητικούς παράγοντες πίνακες, έστω W διάστασης $m \times r$ και H διάστασης $n \times r$, έτσι ώστε: $V \sim WH^T$. Η διάσταση r επιλέγεται, συνήθως, μικρότερη από το $\min\{n, m\}$, έτσι ώστε οι παραγόμενοι πίνακες να είναι αρκετά μικρότερης διάστασης από τον πίνακα της εισόδου. Με άλλα λόγια, κάθε διάνυσμα δεδομένων v, δηλαδή κάθε στήλη του πίνακα V, προσεγγίζεται από έναν γραμμικό συνδυασμό του πίνακα του W, σταθμισμένο με τα στοιχεία-στήλες του πίνακα H ($v_j = Wh_j$). Επομένως, στα πλαίσια και της μείωσης των διαστάσεων, ο W μπορεί να θεωρηθεί πως περιέχει μια βάση, που έχει προσαρμοστεί για τη γραμμική προσέγγιση των δεδομένων στο V. Αφού υπάρχουν σχετικά λίγα διανύσματα βάσης W, που χρησιμοποιούνται για να εκφράσουν πολλά διανύσματα δεδομένων (που κατασκευάζουν τον πλήρη χώρο), καλή προσέγγιση μπορεί να επιτευχθεί, μόνο εάν τα διανύσματα βάσης αυτά ανακαλύψουν κάποια λανθάνουσα δομή στα δεδομένα. Παρατηρούμε, πως υπάρχει εκ κατασκευής η ιδιότητα ομαδοποίησης των στηλών του πίνακα V. Σε μια αντιστοιχία με την μέθοδο k-means, θα μπορούσε κανείς να πει, πως ο πίνακας W παρουσιάζει την συμμετοχή στις ομάδες και ο πίνακας H τα αντίστοιχα κεντροειδή των ομάδων αυτών. Κάθε ομάδα μπορεί να αντιπροσωπευθεί από ένα μοναδικό διάνυσμα βάσης του W. Τέλος, ο πίνακας H^T μπορεί να θεωρηθεί μια αναπαράσταση του πίνακα V σε έναν χώρο χαμηλότερης διάστασης. Κατά την ανάθεση κόμβου σε ομάδες επιλέγεται η μεγαλύτερη τιμή της αντίστοιχης στήλης του πίνακα H^T και αντίστοιχα σε επικαλυπτόμενες κοινότητες, επιλέγονται οι τιμές, που είναι μεγαλύτερες (ή μικρότερες) από μια προκαθορισμένη τιμή (threshold).

Για να βρούμε μια κατά προσέγγιση παραγοντοποίηση $V \sim WH^T$, πρέπει πρώτα να ορίσουμε μια συνάρτηση κόστους, που ποσοτικοποιεί την ποιότητα της προσέγγισης. Μια τέτοια συνάρτηση κόστους μπορεί να κατασκευαστεί, χρησιμοποιώντας κάποιο μέτρο απόστασης μεταξύ των μη αρνητικών πινάκων V και WH^T. Ένα χρήσιμο μέτρο, που χρησιμοποιείται στην NAGC, είναι η νόρμα Frobenius. Η NMF εξαιτίας της χρήσης γραμμικού συνδυασμού για την προσέγγιση του αρχικού πίνακα, όπως αναφέρθηκε, δεν έχει καλά αποτελέσματα σε μη γραμμική δομή δεδομένων. [31]

$$\min_{V \geq 0, W \geq 0, H \geq 0} ||V - WH^T||_F^2 \tag{4.1}$$



Σχήμα 4.1: Η μέθοδος NMF

4.1.1 Κανόνες Ανανέωσης (Updating Rules)

Η συνάρτηση, που παράγεται από προβλήματα μη αρνητικής παραγοντοποίησης πινάκων (NMF) και των διαφόρων παραλλαγών της, επί τω πλείστων (πέρα από ελάχιστες περιπτώσεις ειδικών παραλλαγών) δεν είναι κυρτή ως προς τις μεταβλητές (παραγόμενους πίνακες), στην γενική περίπτωση : W και H, ταυτόχρονα, αλλά είναι κυρτή ως προς κάθε μία από αυτές ξεχωριστά. Ως αποτέλεσμα, δεν μπορούμε να αναμένουμε, να είναι εφικτή η εύρεση ενός ολικού ακρότατου σημείου κατά την βελτιστοποίηση της. Προς αυτή την κατεύθυνση έχουν προταθεί αποτελεσματικοί αλγόριθμοι αριθμητικής εύρεσης τοπικών ακρότατων (ή αρκετά καλών στάσιμων σημείων) σε λογικούς χρόνους εκτέλεσης. Μια από τις πιο διαδεδομένες μεθόδους, και αυτή που χρησιμοποιείται στην NAGC, η οποία έγινε γνωστή μέσω της έρευνας των Lee και Seung [31], είναι η μέθοδος των επαναληπτικών πολλαπλασιαστικών κανόνων ανανέωσης (multiplicative updating rules). Θεωρείται, πως αποτελεί ειδική περίπτωση της μεθόδου ANLS (Alternating Nonnegative Least Squares) [33] υπό την προϋπόθεση επιλογής της νόρμας Frobenius ως μέτρο απόστασης μεταξύ των πινάκων. Δεδομένου ενός προβλήματος της NMF, με την χρήση της ANLS δημιουργούνται δύο υποπροβλήματα, προερχόμενα από το αρχικό πρόβλημα. Κατά την επίλυση του πρώτου, θεωρούμε σταθερή την μία μεταβλητή/παράγοντα πίνακα και κατά την επίλυση του δεύτερου θεωρούμε σταθερή την δεύτερη μεταβλητή/παράγοντα πίνακα. Σημειώνεται, πως η σειρά επίλυσης των υποπροβλημάτων δεν έχει σημασία. Με τον τρόπο αυτό, μετατρέπουμε το μη κυρτό πρόβλημα σε δύο κυρτά και οι βέλτιστες λύσεις τους είναι εφικτό να βρεθούν. Με την ίδια φιλοσοφία, η μέθοδος των πολλαπλασιαστικών κανόνων ανανέωσης καλείται να επιλύσει τα δύο εξής υποπροβλήματα :

$$\min_{V\geq 0, W\geq 0} ||V - WH^T||_F^2 \tag{4.2}$$

, για H σταθερό.

$$\min_{V\geq 0, H\geq 0} ||V - WH^T||_F^2 \tag{4.3}$$

, για W σταθερό.

Προκύπτουν οι παρακάτω πολλαπλασιαστικοί κανόνες ανανέωσης :

$$W \to \frac{VH}{WH^TH} \tag{4.4}$$

, για H σταθερό.

$$H \to \frac{WV^T}{WW^TH} \tag{4.5}$$

, για W σταθερό.

Απόδειξη των σχέσεων (4.4) και (4.5)

Έστω $f(W,V) = ||V - WH^T||_F^2$. Από τις ιδιότητες του ίχνους προκύπτει :

$$\begin{aligned} f(W,V) = ||V - WH^T||_F^2 &= Tr[(V - WH^T)(V - WH^T)^T] = \\ &= Tr[(V - WH^T)(V^T - HW^T)] \\ &= Tr[VV^T - VHW^T - WH^TV^T + WH^THW^T] = \\ &= Tr(VV^T) - Tr(VHW^T) - Tr(WH^TV^T) + Tr(WH^THW^T) \end{aligned}$$

Εφαρμόζεται μερική παραγώγιση ως προς την μεταβλητή W :

$$\frac{\partial f(W,V)}{\partial W} = -VH - VH + W(H^TH + H^TH)$$

Αντίστοιχα, για την μεταβλητή H :

$$\frac{\partial f(W,V)}{\partial H} = -WV^T - WV^T + (WW^T + WW^T)H$$

Σύμφωνα με την δημοσίευση των Lee και Seung [31] με την χρήση του τύπου :

$$\theta \to \theta \frac{\nabla_\theta^- f(\theta)}{\nabla_\theta^+ f(\theta)} \tag{4.6}$$

Καταλήγουμε στις σχέσεις (4.4) και (4.5).

Η μέθοδος αυτή χρησιμοποιείται αρκετά, εξαιτίας του ότι είναι σχετικά απλή, δεν περιλαμβάνει εξωτερικές παραμέτρους, έχει χαμηλό κόστος σε κάθε επανάληψη και η σύγκλιση σε κάποιο ακρότατο σημείο είναι αποδεδειγμένη [31]. Παρ'όλα αυτά, πολλές φορές είναι αργή και σε κάποιες έρευνες έχει αμφισβητηθεί η αποτελεσματικότητα της. Το είδος του προκύπτοντος ακρότατου και κατά συνέπεια η βελιστότητα της λύσης, μπορεί να επιβεβαιωθεί και να εξασφαλιστεί με την χρήση επικουρικών μεθόδων της αριθμητικής ανάλυσης, όπως είναι η μέθοδος Lagrange και οι συνθήκες Karush Kahn Tucker, στην περίπτωση της NAGC, οι οποίες θα αναλυθούν σε μεγαλύτερο βάθος σε επόμενο κεφάλαιο. Το γενικότερο πλαίσιο της ANLS και κατά συνέπεια και η μέθοδος των πολλαπλασιαστικών κανόνων ανανέωσης, ακολουθούν την φιλοσοφία του Expectation-Maximization τύπου αλγορίθμου. Οι EM αλγόριθμοι, όπως λέγονται, χρησιμοποιούνται στον κλάδο της στατιστικής και ουσιαστικά είναι επαναληπτικές μέθοδοι για την εύρεση τοπικά μέγιστης πιθανοφάνειας ή μέγιστες aposteriori εκτιμήτριες παραμέτρων σε στατιστικά μοντέλα, όταν τα μοντέλα δέχονται επιρροή από λανθάνουσες παραμέτρους. [34]

4.1.2 Η NMF ως μη κυρτό πρόβλημα βελτιστοποίησης

Έστω $\min_{V,W,H\geq 0} ||V-WH||_F^2$ η συνάρτηση που καλούμαστε να βελτιστοποιήσουμε σε ένα γενικό πρόβλημα παραγοντοποίησης μη αρνητικού πίνακα, σε μη αρνητικούς πίνακες (NMF). Όπως αναφέρθηκε, η NMF θεωρείται, γενικά, πως αντιπροσωπεύει ένα μη κυρτό πρόβλημα βελτιστοποίησης. Παρακάτω παρουσιάζεται μια σύντομη απόδειξη προς επίρρωση αυτού του ισχυρισμού.

Απόδειξη:

$$\min_{V,W,H\geq 0} ||V-WH||_F^2$$

Οι διαστάσεις των πινάκων υπό μελέτη, ορίζονται ως 1 (βαθμωτή περίπτωση).

$$\min_{v,w,h\geq 0} (v-wh)^2 = \min_{v,w,h\geq 0} (v^2 - 2vwh + w^2h^2)$$

Για $f_v(w,h) = v^2 - 2vwh + w^2h^2$ υπολογίζουμε το διάνυσμα πρώτων παραγώγων της συνάρτησης (gradient) και τον Εσσιανό πίνακα.

$$\nabla f_v(w,h) = \begin{bmatrix} \dfrac{\partial f_v}{\partial w} \\ \\ \dfrac{\partial f_v}{\partial h} \end{bmatrix} = \begin{bmatrix} 2wh^2 - 2vh \\ 2w^2h - 2vw \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Ο Εσσιανός πίνακας :

$$\nabla(\nabla f_v(w,h)) = \begin{bmatrix} \dfrac{\partial f_1}{\partial w} & \dfrac{\partial f_2}{\partial w} \\ \\ \dfrac{\partial f_1}{\partial h} & \dfrac{\partial f_2}{\partial h} \end{bmatrix} = \begin{bmatrix} 2h^2 & 4wh - 2v \\ 4wh - 2v & 2w^2 \end{bmatrix}$$

Για να έχουμε μια κυρτή συνάρτηση, ο Εσσιανός πίνακας, που προκύπτει, θα πρέπει να είναι θετικά ημιορισμένος (συμμετρικός με θετικά ιδιοδιανύσματα), όπως αναφέρθηκε στο Κεφάλαιο 2. Ο πίνακας H δεν ικανοποιεί, όμως, αυτή τη συνθήκη, καθώς, αν θέσουμε διακεκριμένα παραδείγματα τιμών, μπορούμε να εντοπίσουμε ύπαρξη αρνητικών ιδιοδιανυσμάτων. Συνεπώς, γενικεύοντας, καταλήγουμε πως η NMF παράγει μη κυρτές συναρτήσεις προς βελτιστοποίηση.

Για την περίπτωση που ακολουθούμε την τακτική των σχέσεων (4.2), (4.3), δηλαδή όταν αντιμετωπίζουμε το πρόβλημα (4.1) ως δύο διαφορετικά προβλήματα, διατηρώντας κάθε φορά τον έναν από τους δύο παράγοντες πίνακες ως σταθερά, τότε, τα δύο αυτά υποπροβλήματα θεωρούνται κυρτά.

Απόδειξη:

$$\min_{V,W,H\geq 0} ||V - WH||_F^2$$

Οι διαστάσεις των πινάκων υπό μελέτη, ορίζονται ως 1 (βαθμωτή περίπτωση).

$$\min_{v,w,h\geq 0} (v - wh)^2 = \min_{v,w,h\geq 0} (v^2 - 2vwh + w^2h^2)$$

Για w σταθερό, $f_w(h) = v^2 - 2vwh + w^2h^2$ υπολογίζεται η πρώτη και η δεύτερη παράγωγος της συνάρτησης ως προς h.

$$\nabla f_w(h) = -2wv + 2w^2h = f_3$$

Η δεύτερη παράγωγος ως προς h :

$$\nabla(\nabla f_w(h)) = \frac{\partial f_3}{\partial h} = 2w^2$$

Συνεπώς, σύμφωνα με το Κριτήριο Δεύτερης Παραγώγου η συνάρτηση είναι κυρτή.

Ομοίως, για h σταθερό, $f_h(w) = v^2 - 2vwh + w^2h^2$ υπολογίζεται η πρώτη και η δεύτερη παράγωγος της συνάρτησης ως προς w.

$$\nabla f_h(w) = -2h + 2wh^2 = f_4$$

Η δεύτερη παράγωγος ως προς h :

$$\nabla(\nabla f_h(w)) = \frac{\partial f_4}{\partial w} = 2h^2$$

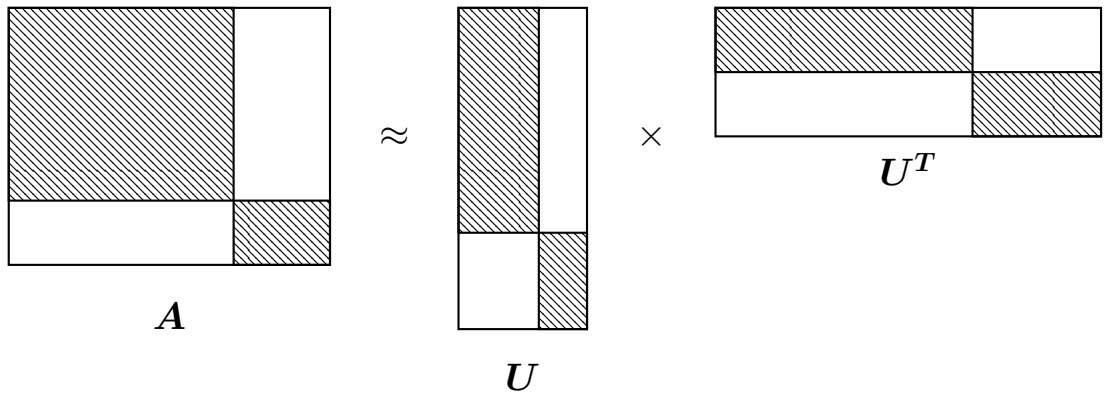
Συνεπώς, σύμφωνα με το Κριτήριο Δεύτερης Παραγώγου του Κεφαλαίου 2 η συνάρτηση είναι κυρτή.

4.2 SNMF (Symmetric Nonnegative Matrix Factorization)

Η βασική επιδίωξη της ομαδοποίησης σε έναν γράφο είναι η επίτευξη της μεγαλύτερης δυνατής ομοιότητας μεταξύ των κόμβων μιας ομάδας, σε σχέση με την ομοιότητα των κόμβων μεταξύ διαφορετικών ομάδων, η οποία θα πρέπει να είναι σημαντικά μικρότερη. Συνδυαστικά, η καλύτερη δυνατή λύση είναι η εύρεση ενός διανύσματος βάσης, κατάλληλου, για την απεικόνιση μιας ομάδας. Με την χρήση της NMF για αυτόν τον σκοπό, δεν είναι εφικτό να λαμβάνουμε πάντα καλά αποτελέσματα, αφού δεν παρέχει υψηλή ανταπόκριση σε όλες τις περιπτώσεις δεδομένων. Όπως αναφέραμε, η NMF δεν μπορεί να ανταποκριθεί στη μη γραμμική δομή των υποκείμενων κοινοτήτων ενός γράφου, εξαιτίας της χρήσης του γραμμικού συνδυασμού για την προσέγγιση το αρχικού πίνακα. Σύμφωνα με μελέτες, μια λύση στο παρόν πρόβλημα δίνει η μέθοδος SNMF, η οποία, όμως, εφαρμόζεται μόνο σε τετραγωνικούς και συμμετρικούς πίνακες. Οι πίνακες αυτοί, δεν χρειάζεται να είναι απαραίτητα μη αρνητικοί (πίνακες ομοιότητας/similarity matrices), γιατί φαίνεται, πως η παρουσία αρνητικών στοιχείων δεν έχει καμία επιρροή στο τελικό αποτέλεσμα. Η διατήρηση, βέβαια, της μη αρνητικότητας προσφέρει την αμεσότερη διάγνωση των αποτελεσμάτων και συνεπώς προτιμάται η μετατροπή όλων των τιμών των στοιχείων σε μη αρνητικές πριν την εκτέλεση της μεθόδου. Το μέτρο αυτό, της ομοιότητας (πίνακας ομοιότητας A), απεικονίζει την σχέση μεταξύ των κόμβων ανά ζεύγη, δηλαδή πόσο όμοιοι είναι οι κόμβοι μεταξύ τους σε τιμές μεταξύ 0 και 1 και θα πρέπει, κάθε φορά, να επιλέγεται ανάλογα με την αρχική δομή των δεδομένων. Σε μια συσχέτιση μεταξύ NMF και SNMF, ο πίνακας ομοιότητας, ο οποίος χρησιμοποιείται στην NMF είναι ο $A = XX^T$, που σημαίνει πως χρησιμοποιεί το εσωτερικό γινόμενο ως μέτρο ομοιότητας (συνημιτονική ομοιότητα), κάτι το οποίο δεν είναι κατάλληλο για όλες τις μορφές δεδομένων. [35]

Στην NAGC, η μέθοδος SNMF χρησιμοποιείται σε μία γενικότερη μορφή της, για την ανάλυση του πίνακα γειτνίασης S $(n \times n)$ στους πίνακες ανάθεσης τοπολογικών ομάδων U και U^T αντίστοιχα. Η μαθηματική μοντελοποίηση του προβλήματος αυτού με την χρήση της νόρμας Frobenius είναι η εξής :

$$\min_{A \geq 0, U \geq 0} ||A - UU^T||_F^2 \tag{4.7}$$



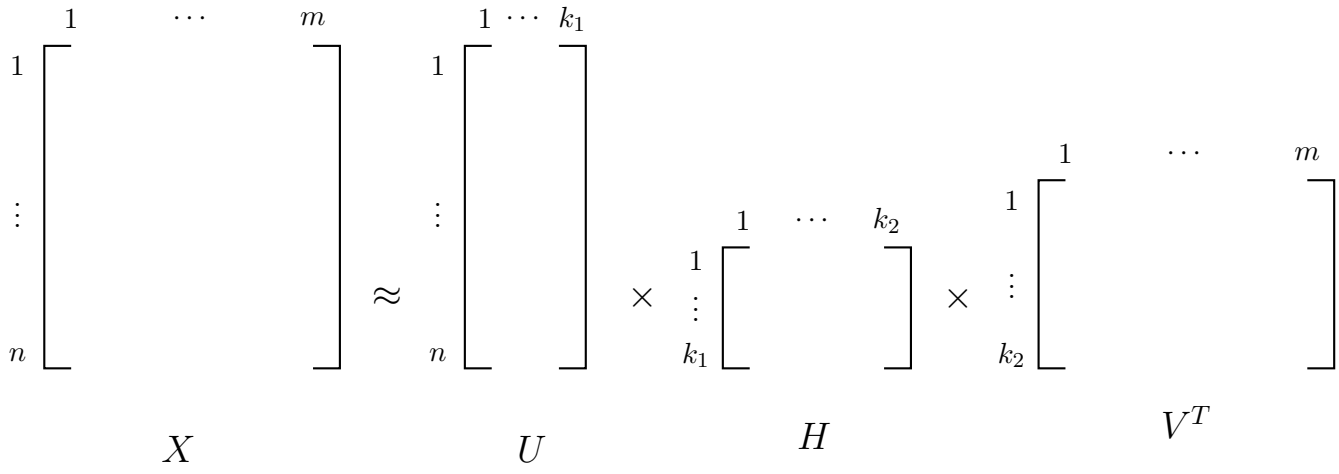
Σχήμα 4.2: Η μέθοδος SNMF.
Πηγή αρχικού σχήματος: Kuang, Ding και Park [35]

4.3 NMTF (Nonnegative Matrix Tri Factorization)

Ένα σύνηθες πρόβλημα, που συναντάται κατά την εφαρμογή της μεθόδου NMF, είναι η περίπτωση στην οποία χρειάζεται να γίνει παράλληλη ομαδοποίηση ως προς τις γραμμές, αλλά και τις στήλες ενός πίνακα. Κατά την ομαδοποίηση αυτή, δεν είναι πάντα απαραίτητο, πως ο αριθμός των ομάδων του αντικειμένου που εκφράζουν οι στήλες του πίνακα, θα είναι ο ίδιος με αυτόν του αντικειμένου, που εκφράζουν οι γραμμές του και το γεγονός αυτό, προκαλεί αλγεβρική ασυμφωνία στις διαστάσεις των πινάκων, με αποτέλεσμα να μην είναι εφικτή η παραγοντοποίηση του αρχικού πίνακα εισόδου. Την λύση στο πρόβλημα αυτό φαίνεται να δίνει η μέθοδος NMTF, κατά την οποία εισάγεται στον γνωστό αλγόριθμο της NMF έναν επιπλέον ενδιάμεσο πίνακα. Ο πίνακας αυτός, διαδραματίζει ρόλο εξισορρόπησης μεταξύ των διαστάσεων των δύο επιμέρους παραγόντων πινάκων, αλλά και της συνολικής διαφοράς κλίμακας, που ενδέχεται να υπάρχει μεταξύ του πίνακα εισόδου και των παραγόμενων. [36]

Στην NAGC, θεωρούμε πως ο αριθμός των ομάδων, όπως διαχωρίζονται με βάση την τοπολογία του γράφου, δεν ταυτίζεται με εκείνον, που έχει ως κριτήριο τα χαρακτηριστικά των κόμβων. Έτσι, η NMTF χρησιμοποιείται για την ανάλυση του πίνακα χαρακτηριστικών X $(n \times m)$ στους πίνακες ανάθεσης τοπολογικών ομάδων U, ανάθεσης ομάδων χαρακτηριστικών V και τέλος στον ενδιάμεσο πίνακα μεταφοράς H. Ο πίνακας H χρησιμοποιείται στην NAGC, ως πίνακας γεφύρωσης του χάσματος κλίμακας μεταξύ των υπόλοιπων πινάκων, αλλά και η φυσική του σημασία αφορά την συσχέτιση μεταξύ των κοινοτήτων, όπως προκύπτουν με γνώμονα την τοπολογική διάταξη (k_1) και των κοινοτήτων που προκύπτουν με βάση τα χαρακτηριστικά των κόμβων (k_2). Η μαθηματική μοντελοποίηση του προβλήματος αυτού, με την χρήση της νόρμας Frobenius είναι η εξής :

$$\min_{X \geq 0, U \geq 0, H \geq 0, V \geq 0} ||X - UHV^T||_F^2 \tag{4.8}$$



Σχήμα 4.3: Η μέθοδος NMTF

Κεφάλαιο 5

Biased Matrix Completion

Σε πολλές περιπτώσεις δεδομένων από στοιχεία του πραγματικού κόσμου, δεχόμαστε ως παραδοχή, πως δεν είναι εφικτό να διατηρήσουμε την απόλυτα ολοκληρωμένη μορφή τους, παρά μόνο μία προσέγγιση τους, σύμφωνα με την χρονική στιγμή στην οποία λήφθηκαν. Αυτή η παραδοχή οδήγησε στην μελέτη της βελτιστοποίησης, ουσιαστικά, της αρχικής αυτής πληροφορίας, με σκοπό την απόδοση μίας πιο αντικειμενικής δομής στα δεδομένα, η οποία θα ανταποκρίνεται σε μεγαλύτερο βαθμό στην πραγματικότητα. Στην περίπτωση που η πληροφορία παρουσιάζεται με την μορφή πίνακα, μπορούμε να αναφερθούμε σε ένα πρόβλημα ολοκλήρωσης πίνακα (matrix completion problem). Τέτοιου είδους προβλήματα παρουσιάζονται πολύ συχνά σε μελέτες, που πραγματεύονται, για παράδειγμα συστήματα συστάσεων (recommendation systems) ή κοινωνικά δίκτυα (social networks) και γενικά σε περιπτώσεις, όπου είναι είτε ανέφικτο, είτε πολύ επιζήμιο να χρησιμοποιηθεί η πλήρως ολοκληρωμένη πληροφορία και αποτελεί καίριο ζήτημα η επίλυση τους, καθώς η διάρθρωση ορθότερων δεδομένων οδηγεί σε πιο ακριβή αποτελέσματα.

Προς αυτή την κατεύθυνση εργάστηκαν οι Elkan και Noto [37] παρουσιάζοντας μια ειδική περίπτωση τέτοιου είδους βελτιστοποίησης. Πιο συγκεκριμένα, εισήγαγαν για πρώτη φορά την έννοια του Positive Unlabeled Learning για δυϊκούς ταξινομητές (binary classifiers), δηλαδή τον τρόπο με τον οποίο μπορεί με δεδομένα : ένα ημιτελές σύνολο θετικά ετικετοποιημένων και με την κοινώς χρησιμοποιούμενη, αγγλική ορολογία labeled-positive στοιχείων και ένα σύνολο μη ετικετοποιημένων (unlabeled) στοιχείων, που αποτελείται από positive και negative στοιχεία, ένας αλγόριθμος να εκπαιδευτεί και να αποφανθεί την τελική, πραγματική ετικετοποίηση όλων των unlabeled στοιχείων, με αυτό να συνεπάγεται την ολοκλήρωση του αρχικού πίνακα δεδομένων. Στην μελέτη αυτή, γίνεται δεκτό, πως η επιλογή του γνωστού υποσυνόλου θετικών ετικετοποιημένων στοιχείων πραγματοποιείται με τυχαίο τρόπο και πως η υπάρχουσα ετικετοποίηση θετικών στοιχείων δεν τίθεται υπό αμφισβήτηση. Στην έρευνα τους οι Elkan και Noto [37] προτάσσουν κάποιες τεχνικές για την ολοκλήρωση πινάκων, διαφοροποιώντας σε μη ντετερμινιστικό και σε ντετερμινιστικό πλαίσιο τους διάφορους σχετικούς αλγορίθμους που τις υλοποιούν. Σε κάθε μία περίπτωση, τα δεδομένα είναι κοινά, αλλά τα αποτελέσματα διαφέρουν. Στο μη ντετερμινιστικό πλαίσιο τα αποτελέσματα μπορούν να προσεγγίσουν το βέλτιστο πίνακα με μεγαλύτερη ακρίβεια, όμως εξαιτίας του ότι βασίζονται σε κατανομή, διέπονται από τυχαιότητα. Αντίθετα, στο ντετερμινιστικό πλαίσιο δεν μπορεί να προσεγγιστεί σε αυτόν τον μεγάλο βαθμό ο βέλτιστος ολοκληρωμένος πίνακας, αλλά μειώνεται ο παράγοντας της τυχαιότητας στην εξαγωγή των αποτελεσμάτων. Στην παρούσα εργασία, θα αναλυθεί η μέθοδος Biased Matrix Completion for Deterministic Setting (BiasMC), η οποία χρησιμοποιήθηκε για την ανάπτυξη της μεθόδου NAGC.

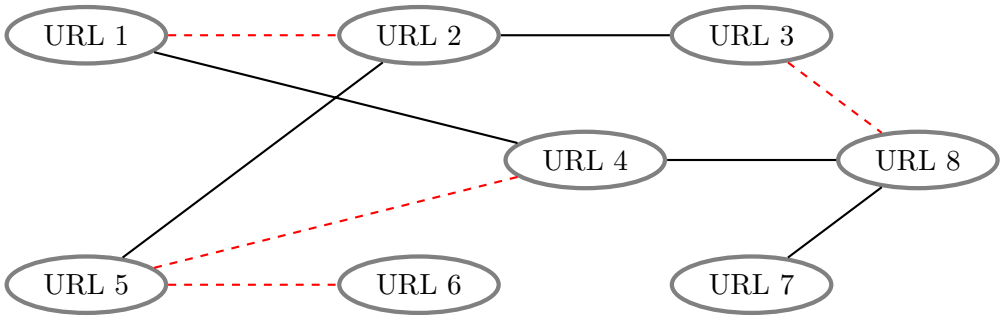
5.1 PU Learning

Σε μια αναδρομή στις βασικές έννοιες της μηχανικής μάθησης, μπορούμε να κατηγοριοποιήσουμε την διαδικασία εκπαίδευσης (training), που ακολουθείται στην περίπτωση του PU Learning, στην ομάδα των semi-supervised μεθόδων εκπαίδευσης μοντέλων/αλγορίθμων μηχανικής μάθησης. Αυτό, γιατί ο αλγόριθμος διατηρεί κάποιες πληροφορίες από το θετικά ετικετοποιημένο υποσύνολο, που λαμβάνει σαν είσοδο, αλλά όλα τα υπόλοιπα στοιχεία εισόδου του, τα οποία μάλιστα, συνήθως, υπερέχουν των προηγούμενων σε μέγεθος, είναι μη ετικετοποιημένα.

Κάνοντας μια μαθηματική μοντελοποίηση, έστω x ένα στοιχείο από τα δεδομένα, $y \in \{0, 1\}$ μεταβλητή, όπου με 1, συμβολίζεται πως το στοιχείο x έχει θετική ετικέτα και $s \in \{0, 1\}$ μεταβλητή, όπου με 1 συμβολίζεται ότι το στοιχείο x έχει ετικέτα. Η πιθανότητα ένα στοιχείο να έχει θετική ετικέτα $P(y = 1|x)$ εκτιμάται ως το πηλίκο της πιθανότητας το στοιχείο να έχει ετικέτα $P(s = 1|x)$ προς την πιθανότητα ένα στοιχείο με θετική ετικέτα, να ανήκει στην κατηγορία των ήδη ετικετοποιημένων στοιχείων $P(s = 1|y = 1)$. Στην μελέτη τους οι Elkan και Noto [37] αποδεικνύουν, πως παρότι μπορεί να μην έχουμε αρκετά ετικετοποιημένα δεδομένα, ώστε να αποφανθούμε άμεσα για το αν ένα τυχαίο στοιχείο ενός δείγματος έχει θετική ετικέτα, όμως έχουμε αρκετές πληροφορίες για να υπολογίσουμε πόσο πιθανό είναι ένα στοιχείο του δείγματος, να έχει, γενικά, ετικέτα ή όχι και αυτό αποδεικνύεται, πως αρκεί για την εξαγωγή αποτελέσματος.

Οι εφαρμογές της μεθόδου είναι πολλές, αλλά στην παρούσα εργασία θα αναλυθεί το πρόβλημα ολοκλήρωσης πίνακα. Ένα παράδειγμα εφαρμογής σε δεδομένα σχετικά των ζητημάτων, που πραγματεύεται η NAGC είναι η εξής :

Έστω δίκτυο, που αποτελείται από συνδέσμους (πχ. ιστοσελίδες) ως κόμβους, οι οποίοι συνδέονται μεταξύ τους με υπερσυνδέσμους ως ακμές. Λαμβάνουμε τις υπάρχουσες ακμές του δείγματος δεδομένων ως το γνωστό θετικά ετικετοποιημένο υποσύνολο και τις μη υπάρχουσες ακμές ως το μη ετικετοποιημένο μέρος του δείγματος, που επιθυμούμε να ετικετοποιήσουμε. Εφαρμόζοντας κατάλληλο αλγόριθμο υλοποίησης, επιτυγχάνεται η προσθήκη θετικών ετικετών σε ακμές, σηματοδοτώντας την ύπαρξη τους, η οποία για κάποιον λόγο δεν καταγράφηκε κατά την λήψη του δείγματος των δεδομένων (Σχήμα 5.1).



Σχήμα 5.1: Προσθήκη ακμών και θετικών ετικετών σε ακμές δικτύου

5.2 Deterministic Setting

Θα χρησιμοποιηθούν οι συμβολισμοί της δημοσίευσής των Hsieh, Natarajan και Dhillon [38].

Έστω πίνακας $M \in \mathbb{R}^{m \times n}$, όπου είναι εφικτό $m = n$, με φραγμένη Schatten-1 (nuclear) νόρμα $||M||_* \leq c$, όπου c σταθερά ανεξάρτητη των m, n. Αν $M_{i,j} \in \{0, 1\}$ για κάθε (i, j) και έστω Ω_1 τυχαία επιλεγμένο σύνολο από το δείγμα $\{(i, j)|M_{i,j} = 1\}$. Σε ένα σενάριο εφαρμογής της μεθόδου PU Learning, σκοπός είναι η επαναφορά του πίνακα M με βάση τα δεδομένα του συνόλου Ω_1. Ο πίνακας M, ουσιαστικά, απεικονίζει την πραγματική δομή των δεδομένων, δηλαδή τον βέλτιστο πίνακα συσχέτισης για τα εκάστοτε δεδομένα. Πρέπει να σημειωθεί, πως για μη συνθετικά δείγματα, ο πίνακας αυτός, δεν είναι εφικτό να περιέχει στοιχεία του συνόλου $\{0, 1\}$ αποκλειστικά και έτσι χρειάζεται να τον μετατρέψουμε με κατάλληλο τρόπο. Παρακάτω αναλύεται το ντετερμινιστικό πλαίσιο εκτέλεσης της μεθόδου, αφού εξηγηθούν οι επιμέρους πίνακες, οι οποίοι είναι απαραίτητο να υπολογιστούν.

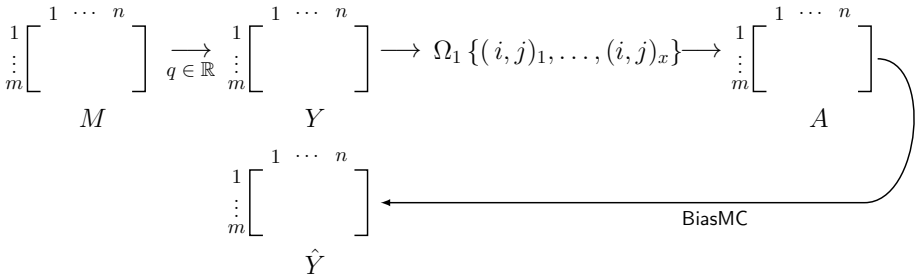
Έστω ο πίνακας M ο οποίος, όπως αναφέρθηκε, δεν είναι εφικτό να αποτελείται αποκλειστικά από στοιχεία του συνόλου $\{0, 1\}$. Έτσι, θέτουμε έναν περιορισμό (threshold), έστω $q \in \mathbb{R}$, τέτοιον ώστε να καταλήξουμε σε έναν πίνακα, έστω Y, με στοιχεία $Y_{ij} = I(M_{ij} > q)$, όπου με I συμβολίζεται η δείκτρια :

$$I = \begin{cases} 1, & \text{αν } M_{ij} > q \\ 0, & \text{αν } M_{ij} \leq q \end{cases}$$

Με τον τρόπο αυτό, παράγουμε τον πίνακα Y, ο οποίος χαρακτηρίζεται με αγγλική ορολογία ως clean 0-1 πίνακας. Ο πίνακας Y είναι ο καλύτερος πίνακας, που είναι εφικτό να ανακτηθεί σε μεγάλο βαθμό, καθώς, όπως είναι σαφές, δεδομένου του περιορισμού q, που εφαρμόζεται, δεν καθίσταται δυνατό να επανέλθουν οι αρχικές τιμές του πίνακα M. Στη συνέχεια, επιλέγεται τυχαία ένα σύνολο θετικών ακμών, έστω Ω_1 από το σύνολο $\{(i, j)|Y_{ij} = 1\}$. Από το σύνολο αυτό και γνωρίζοντας όλους τους κόμβους, δημιουργείται ο πίνακας A με τον παρακάτω τρόπο :

$$A_{ij} = \begin{cases} 1, & \text{αν } (i, j) \in \Omega_1 \\ 0, & \text{αν } (i, j) \notin \Omega_1 \end{cases}$$

Ο πίνακας A αποτελεί τον αρχικό πίνακα γειτνίασης (για $m = n$)/συσχέτισης, τον οποίο θεωρούμε πως διαθέτουμε με την αρχική μορφή των δεδομένων και καλούμαστε, δεχόμενοι αυτόν ως είσοδο, να εντοπίσουμε τις πληροφορίες, που λείπουν και να καταλήξουμε στον πίνακα . Για την επίτευξη αυτού, έχουν προταθεί διάφορες μέθοδοι. Θα αναλυθεί παρακάτω η μέθοδος Biased Matrix Completion (Σχήμα 5.2).



Σχήμα 5.2: Μέθοδος Biased Matrix Completion

5.3 Biased Matrix Completion (BiasMC)

Ο αλγόριθμός Biased Matrix Completion (Bias MC) αποτελεί το μέσο προσέγγισης του πίνακα Y κατά το ντετερμινιστικό πλαίσιο. Εισήχθηκε επίσημα στην δημοσίευση των Hsieh, Natarajan και Dhillon [38], από όπου αντλήθηκαν πληροφορίες και συμβολισμοί, που θα παρασταθούν στη συνέχεια. Η μέθοδος αυτή, ουσιαστικά, αφορά μια συνάρτηση κόστους/ελαχιστοποίησης απόστασης μεταξύ δύο πινάκων με μεροληψία, σχετική με τις ετικέτες του δείγματος. Πιο συγκεκριμένα, για την υλοποίηση της BiasMC ορίζεται ως συνάρτηση σφάλματος (loss function) η τετραγωνική :

$$\ell(x, a) = (x - a)^2 \tag{5.1}$$

, όπου $a \in \{0, 1\}$.

Η συνάρτηση αυτή, μετατρέπεται σε συνάρτηση με εξάρτηση από ετικέτα (label dependent loss function) [39] με τον εξής τρόπο :

$$\ell_\alpha(x, a) = \alpha 1_{a=1}\ell(x, 1) + (1 - \alpha)1_{a=0}\ell(x, 0) \tag{5.2}$$

, όπου $1_{a=1}, 1_{a=0}$ δείκτριες συναρτήσεις,οι οποίες διαδραματίζουν ρόλο βάρους και παίρνουν την τιμή 1 για την αναγραφόμενη στον δείκτη τους τιμή του a και α η μεροληψία, που αποδίδεται στην εκάστοτε ετικέτα. Στην παρούσα εργασία, η τιμή του α ορίζεται στο διάστημα $[0, 1]$. Οι συναρτήσεις με εξάρτηση από ετικέτα χρησιμοποιούνται ευρέως σε μη ισορροπημένα δεδομένα (unbalanced data), καθώς για παράδειγμα σε αραιούς πίνακες (sparse matrices), όπου συνήθως τα περισσότερα στοιχεία είναι μηδενικά, η επιρροή της πλειοψηφίας των στοιχείων οδηγεί με μεγάλη πιθανότητα σε λανθασμένες προβλέψεις και αποτελέσματα.

Ορίζεται ως A ο ημιτελής πίνακας εισόδου και X ο πίνακας, που πρέπει να προσεγγιστεί ως μία εκτίμηση του πίνακα Y. Ο πίνακας X αρχικοποιείται με τυχαίο τρόπο και με κατάλληλους κανόνες ανανέωσης ζητείται να προσεγγίσει κατά το δυνατό περισσότερο τον πίνακα A. Με την προσεγγιση αυτή, εξαιτίας της μεροληψίας, που δίνεται με την μορφή της μεταβλητής α, ο τρόπος με τον οποίο υπολογίζεται η απόσταση μεταβάλλεται, με αποτέλεσμα ο τελικός προκύπτων πίνακας, να περιέχει πολύτιμη πληροφορία σχετικά με την παρουσία επιπλέον ακμών, που δεν συμπεριλαμβάνονταν στον πίνακα A. Η συνάρτηση σφάλματος αφορά ένα μεμονωμένο στοιχείο x και για να γενικευθεί σε συνάρτηση κόστους, θα πρέπει να υπολογιστεί για το σύνολο όλων των στοιχείων του δείγματος. Η τελική συνάρτηση κόστους της μεθόδου εκφράζεται ως :

$$\hat{X} = \underset{X:||X|_* \leq t}{\operatorname{argmin}} \sum_{i,j} \ell_\alpha(X_{ij}, A_{ij}) \tag{5.3}$$

και με χρήση της σχέσης (5.2) μετατρέπεται σε :

$$\hat{X} = \underset{X:||X||_* \leq t}{\operatorname{argmin}} \alpha \sum_{i,j:A_{ij}=1} (X_{ij} - 1)^2 + (1 - \alpha) \sum_{i,j:A_ij=0} X_{ij}^2 \tag{5.4}$$

Όπως φαίνεται από την δομή της συνάρτησης, μεγάλες τιμές της μεταβλητής α υποδεικνύουν απόδοση μεγαλύτερης βαρύτητας στην προσέγγιση στον πίνακα X, των υπαρχόντων στον πίνακα A ακμών (στοιχεία με τιμή 1) και μικρότερη βαρύτητα στην προσέγγιση των στοιχείων του με τιμή 0. Στις περιπτώσεις αυτές, εξαιτίας και της τυχαίας αρχικοποίησης, η προσέγγιση των στοιχείων με τιμή 0 δεν πραγματοποιείται με την ίδια προτεραιότητα και έτσι αρκετά στοιχεία του πίνακα X, τα οποία θα έπρεπε να πάρουν την τιμή 0, δεν θα την πάρουν. Έτσι, παράγονται οι επιπλέον ακμές που οδηγούν στην εύρεση μιας εκτίμησης του πίνακα Y. Προφανώς για $\alpha = 0.5$ οι ετικετοποιημένες/θετικές ακμές αντιμετωπίζονται με όμοιο τρόπο με τις μη ετικετοποιημένες.

Καταληκτικά, με την περάτωση της διαδικασίας της βελτιστοποίησης ο πίνακας $\hat{X}$ μετατρέπεται με χρήση της σταθεράς $q \in \mathbb{R}$, με τον εξής τρόπο :

$$\bar{X}_{ij} = I(\hat{X}_{ij} > q) \tag{5.5}$$

, όπου με I συμβολίζεται η δείκτρια συνάρτηση :

$$I = \begin{cases} 1, & \text{αν } \hat{X}_{ij} > q \\ 0, & \text{αν } \hat{X}_{ij} \leq q \end{cases}$$

Ο πίνακας $\bar{X}$ είναι η βέλτιστη εκτιμήτρια του πίνακα Y.

Στην NAGC, η μέθοδος αυτή χρησιμοποιείται για τον πίνακα S (πίνακας γειτνίασης) και το γινόμενο πινάκων UU^T, με σκοπό την ελαχιστοποίηση της απόστασης μεταξύ τους και παράλληλα τον εμπλουτισμό του πίνακα U, ο οποίος αποτελεί και τον πίνακα, που θα καθορίσει την τελική συμμετοχή των κόμβων σε κοινότητες, με πιο αντικειμενικά ορθή πληροφορία, δηλαδή ακμές, που θα έπρεπε να ληφθούν υπόψιν στον τελικό υπολογισμό του αποτελέσματος. Ο συμβολισμός που χρησιμοποιείται στην μέθοδο NAGC είναι ο εξής :

$$\mathfrak{L}_\rho(Z) = \sum_{(i,j)\in E} \rho(z_{i,j} - 1)^2 + (1-\rho) \sum_{(i,j)\notin E} z_{i,j}^2 \tag{5.6}$$

, όπου με E συμβολίζεται το σύνολο των υπαρχόντων στον S ακμών, ρ είναι η μεταβλητή, η οποία προσδίδει την μεροληψία, Z ο πίνακας εισόδου και z_{ij} τα στοιχεία του.

Παρατηρούμε επίσης τον συμβολισμό $\mathfrak{L}_\rho(S - UU^T)$, γιατί είναι εφικτό να μετατραπεί ο τύπος ως :

$$\mathfrak{L}_\rho(S - UU^T) = \sum_{(i,j)\in E} \rho(U_{ij}U_{ij}^T - S_{ij})^2 + (1-\rho) \sum_{(i,j)\notin E} (U_{ij}U_{ij}^T)^2 \tag{5.7}$$

ή

$$\mathfrak{L}_\rho(S - UU^T) = \sum_{i,j:S_{ij}=1} \rho(U_{ij}U_{ij}^T - S_{ij})^2 + (1-\rho) \sum_{i,j:S_{ij}=0} (U_{ij}U_{ij}^T)^2 \tag{5.8}$$

Κεφάλαιο 6

Βελτιστοποίηση

Η βελτιστοποίηση αποτελεί αναπόσπαστο κομμάτι της επιστήμης των εφαρμοσμένων μαθηματικών, καθώς δίνει λύσεις σε καίρια και σύνθετα ζητήματα. Ο κλάδος αυτός πραγματεύεται τον προσδιορισμό, ταυτόχρονα βέλτιστων τιμών στις παραμέτρους ενός προβλήματος και τις μεθόδους με τις οποίες μπορεί να επιτευχθεί αυτό με τον καλύτερο δυνατό τρόπο, με βάση την φύση του εκάστοτε προβλήματος υπό μελέτη. Ένα πρόβλημα βελτιστοποίησης εκφράζεται από μια συνάρτηση στόχου (objective function) ή αλλιώς συνάρτηση κόστους (cost function) και το σύνολο των περιορισμών, που διέπουν την λύση του (feasibility region), εφόσον υπάρχουν. Η κατανομή των προβλημάτων αυτών μπορεί να γίνει σε πολλές και διαφορετικές κατηγορίες, ανάλογα με τα χαρακτηριστικά τους, η βασικότερη διάκριση τους, όμως, αφορά την μεγιστοποίηση και ελαχιστοποίηση της συνάρτησης στόχου. Εξαιτίας της πολυπλοκότητας, που εμφανίζουν οι συναρτήσεις για την περιγραφή των μη τεχνητών προβλημάτων (μη γραμμικότητα, έλλειψη κυρτότητας κ.α.), έχουν αναπτυχθεί διάφορες και συνήθως αριθμητικές μέθοδοι προς την επίτευξη της καλύτερης δυνατής προσέγγισης της λύσης. Πιο αναλυτικά, στις περιπτώσεις αυτές, δεν μπορεί κανείς να περιμένει μοναδική λύση, δηλαδή την ύπαρξη κάποιου ολικού ακρότατου, αλλά την προσέγγιση κάποιου καλού στάσιμου σημείου (stationary point). Στα ζητήματα με παρουσία περιορισμών οι διαδικασίες ποικίλουν με επικρατέστερη να είναι η μέθοδος Lagrange, η οποία αφορά αποκλειστικά περιορισμούς ισότητας και με την χρήση των συνθηκών Karush-Kahn-Tucker (KKT conditions) επεκτείνεται σε περιορισμούς ανισότητας, επίσης. Ο σκοπός της μεθόδου Lagrange είναι η συγχώνευση των περιορισμών εντός της συνάρτησης προς βελτιστοποίηση με αποτέλεσμα, να μπορεί να εφαρμοστεί ο εκάστοτε αλγόριθμος βελτιστοποίησης καθολικά στο πρόβλημα, χωρίς παράπλευρους υπολογισμούς εξαιτίας των περιορισμών, ως ανεξάρτητες σχέσεις. Συμπληρωματικά, οι συνθήκες Karush-Kahn-Tucker διαδραματίζουν τον ρόλο περιορισμών στις υποψήφιες λύσεις και εξασφαλίζουν την βελτιστότητα, που απαιτείται και την διαμόρφωση κανόνων ανανέωσης εντός της εφικτής περιοχής.

Συγκεκριμένα, η NAGC αναπαράγει ένα μη κυρτό πρόβλημα ελαχιστοποίησης με περιορισμούς. Αρχικά, γίνεται σαφές πως δεν μπορεί να προσδιοριστεί η ακριβής και βέλτιστη λύση, αλλά είναι εφικτή μια καλή προσέγγιση με την χρήση κατάλληλης αριθμητικής μεθόδου. Συμπληρωματικά, η ύπαρξη περιορισμών επιβάλει την χρήση της μεθόδου Lagrange και επειδή οι περιορισμοί αυτοί είναι ανισοτικές σχέσεις, θα πρέπει παράλληλα να εφαρμοστούν οι συνθήκες Karush-Kahn-Tucker. Ο τρόπος διαχείρισης αυτός περιγράφεται αναλυτικά στις επόμενες υποενότητες.

6.1 NAGC

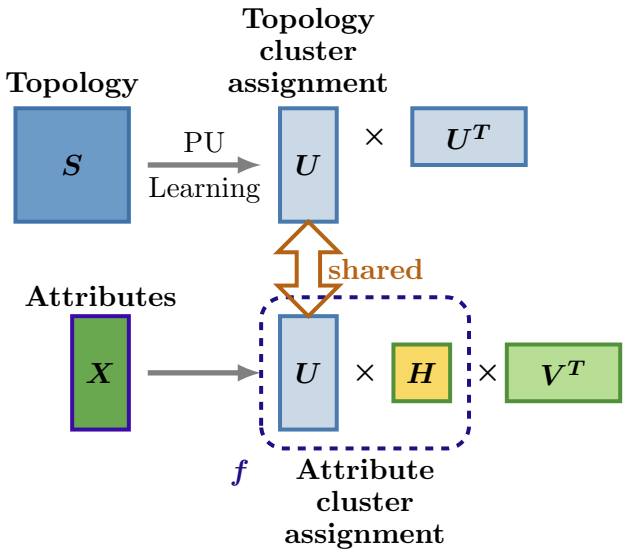
Συνδυάζοντας τις μεθόδους που περιγράφηκαν σε όλα τα προηγούμενα κεφάλαια, καταλήγουμε στην οριστική συνάρτηση της NAGC (6.1) :

$$\min_{U,V,H\geq 0} \mathfrak{L}_\rho(S - UU^T) + \frac{\lambda}{2}||X - f(UH)V^T||_F^2 \tag{6.1}$$

, όπου

- λ μη αρνητική παράμετρος ρύθμισης μεταξύ της βαρύτητας που δίνεται στην τοπολογική διάταξη και τα χαρακτηριστικά αντίστοιχα. Τιμές του λ κοντά στο μηδέν υποδηλώνουν μείωση της επιρροής των χαρακτηριστικών στο τελικό αποτέλεσμα.
- f μη γραμμική συνάρτηση. Στην περίπτωση της παρούσας μελέτης χρησιμοποιούνται είτε η σιγμοειδής συνάρτηση, είτε η συνάρτηση διορθωμένης γραμμικής μονάδας (ReLU), με σκοπό την μετατροπή των στοιχείων των πινάκων-ορισμάτων της σε συγκρίσιμες κλίμακες μεγέθους.
- Οι πίνακες S, U, X, H και V, όπως έχουν ήδη οριστεί στο Κεφάλαιο 2.
- $||\cdot||_F$, η νόρμα Frobenius / Schatten 2-νόρμα.
- $\mathfrak{L}_\rho$, η μεροληπτική μετρική της μεθόδου BiasMC, όπως περιγράφεται από την σχέση (5.6).

Ο πίνακας U είναι κοινός στις παραγοντοποιήσεις των πινάκων S και X. Δεδομένου, πως αποτελεί τον βασικό πίνακα που καταληκτικά θα υποδείξει την τελική κατανομή των ομάδων, με αυτό τον τρόπο λαμβάνει πληροφορία και από τους δύο πίνακες, τοπολογίας και χαρακτηριστικών (Σχήμα 6.1).



Σχήμα 6.1: Παραγοντοποιήσεις των πινάκων S και X.
Πηγή αρχικού σχήματος: Maekawa, Takeuchi και Onizuka [3]

6.2 Η μέθοδος Lagrange - Συνθήκες Karush-Kahn-Tucker

Όπως αναφέρθηκε, ο στόχος της μεθόδου Lagrange είναι να σχηματιστεί μια νέα συνάρτηση στόχου, η οποία θα συμπεριλαμβάνει με κατάλληλο τρόπο τους υπάρχοντες περιορισμούς, ώστε να είναι δυνατό να εφαρμοστούν σε αυτή όλοι οι πιθανοί αλγόριθμοι βελτιστοποίησης. Στην κλασική εκδοχή της, η μέθοδος Lagrange αφορά μόνο περιορισμούς ισότητας και για αυτόν τον λόγο έχουν εισαχθεί στην μελέτη οι συνθήκες Karush-Kahn-Tucker, με στόχο την επέκταση σε περίπτωση ανισοτικών περιορισμών. Παρακάτω παρουσιάζεται μια γενική μεθοδολογία, η οποία ακολουθείται και κατά την βελτιστοποίηση της συνάρτησης στόχου της NAGC [40].

Έστω πρόβλημα βελτιστοποίησης :

$$\min_{x \in X \subset \mathbb{R}^n} f(x) \tag{6.2}$$

subject to

$$\begin{cases} c_i(x) - b_i \geq 0, i = 1, \ldots, k \\ c_i(x) - b_i = 0, i = k+1, \ldots, m \end{cases} \tag{6.3}$$

, όπου X ο χώρος των αποδεκτών λύσεων, x το διάνυσμα των μεταβλητών της συνάρτησης, $c_i(x)$ οι m περιορισμοί του προβλήματος και b_i σταθερές. Οι περιορισμοί ισότητας ονομάζονται ενεργοί περιορισμοί και οι περιορισμοί ανισότητας ονομάζονται ανενεργοί περιορισμοί.

Για το πρόβλημα ελαχιστοποίησης, ορίζεται η συνάρτηση Lagrange από τον γενικό τύπο :

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x) \tag{6.4}$$

, όπου λ_i οι πολλαπλασιαστές Lagrange.

Η στρατηγική αυτή, καλείται Μέθοδος Πολλαπλασιαστών Lagrange και χρησιμοποιείται ευρέως στα προβλήματα μεγιστοποίησης-ελαχιστοποίησης. Ουσιαστικά, ένας πολλαπλασιαστής Lagrange είναι μια τιμή, η οποία είναι απαραίτητη για την εισαγωγή των περιορισμών εντός της συνάρτησης στόχου.

Για να εντοπιστεί η βέλτιστη λύση, θα πρέπει να βρεθούν τα στάσιμα σημεία και να προσδιοριστεί το βέλτιστο. Προς αυτή την κατεύθυνση, στην περίπτωση προβλήματος με ανενεργούς περιορισμούς, θα πρέπει να ικανοποιούνται οι συνθήκες Karush-Kahn-Tucker. Οι συνθήκες αυτές αποτελούν βασικές προϋποθέσεις ύπαρξης ακρότατου σε περιπτώσεις προβλημάτων βελτιστοποίησης με περιορισμούς.

Οι συνθήκες Karush-Kahn-Tucker ορίζονται με τον παρακάτω τρόπο :

Για x^* βέλτιστη λύση, ισχύει :

- **Συνθήκη Στασιμότητας/Συνθήκη Βελτιστότητας (Optimality Condition) :**
  Η συνθήκη αυτή εξασφαλίζει, πως η τιμή x^* είναι η βέλτιστη και οποιαδήποτε άλλη τιμή εντός της εφικτής περιοχής δεν θα βελτιώσει περαιτέρω την συνάρτηση στόχου.

$$\nabla_x L(x^*, \lambda^*) = 0 \iff \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla c_i(x^*) = 0 \tag{6.5}$$

- **Συνθήκες Πρωταρχικής Εφικτότητας (Primal Feasibility Condition) :**
  Οι συνθήκες αυτές, εξασφαλίζουν την τήρηση των περιορισμών για την λύση του προβλήματος.

$$c_i(x^*) = 0, \text{ για περιορισμούς ισότητας-ενεργοί περιορισμοί} \tag{6.6}$$

$$c_i(x^*) \geq 0, \text{για περιορισμούς ανισότητας-ανενεργοί περιορισμοί} \tag{6.7}$$

- **Συνθήκη Εφικτότητας του Δυϊκού (Dual Feasibility Condition) :**
  Η συνθήκη αυτή, εξασφαλίζει την θετικότητα των πολλαπλασιαστών Lagrange, σε περίπτωση ενεργού περιορισμού.

$$\lambda_i^* \geq 0, \text{ για κάθε } i = 1, \ldots, m \tag{6.8}$$

- **Συνθήκη Συμπληρωματικής Χαλαρότητας (Complementary Slackness Condition) :**
  Η συνθήκη αυτή εξασφαλίζει, πως ένας πολλαπλασιαστής Lagrange, μπορεί να πάρει μηδενική τιμή μόνο στην περίπτωση ανενεργού περιορισμού.

$$\lambda_i^* c_i(x^*) = 0, \text{ για κάθε } i = 1, \ldots, m \tag{6.9}$$

Οι συνθήκες (6.5) και (6.6) αφορούν τα προβλήματα με περιορισμούς ισότητας και ονομάζονται δεσμευτικοί περιορισμοί. Όπως αναφέρθηκε, οι Karush-Kahn-Tucker αποτελούν αναγκαίες συνθήκες για την ύπαρξη ακρότατου σημείου σε προβλήματα βελτιστοποίησης, που περιλαμβάνουν περιορισμούς. Ειδικότερα, σε περιπτώσεις, όπου τόσο η συνάρτηση στόχου, αλλά και οι περιορισμοί ανισότητας είναι κυρτές συναρτήσεις και επιπλέον οι περιορισμοί ισότητας είναι γραμμικές συναρτήσεις, οι συνθήκες Karush-Kahn-Tucker είναι αναγκαίες και ικανές για την ύπαρξη ολικού ακρότατου. [41]

6.3 Βελτιστοποίηση στην NAGC

Η συνάρτηση προς βελτιστοποίηση, όπως προκύπτει στην μέθοδο NAGC, είναι μια μη γραμμική και μη κυρτή συνάρτηση με περιορισμούς ανισοτικές σχέσεις. Συγκεκριμένα, ως πρόβλημα βελτιστοποίησης έχει την παρακάτω μορφή :

$$\min_{U,V,H\geq 0} \mathfrak{L}_\rho(S - UU^T) + \frac{\lambda}{2}||X - f(UH)V^T||_F^2 \tag{6.1}$$

$$\text{,όπου} \quad \mathfrak{L}_\rho(Z) = \sum_{(i,j)\in E} \rho(z_{i,j} - 1)^2 + (1-\rho) \sum_{(i,j)\notin E} z_{i,j}^2 \tag{5.6}$$

subject to

$$\begin{cases} U \geq 0 \\ V \geq 0 \\ H \geq 0 \end{cases} \tag{6.10}$$

Σύμφωνα με την προαναφερθείσα θεωρία, η διαχείριση στα προβλήματα αυτού του είδους πραγματοποιείται με τον παρακάτω τρόπο :

- Υπολογίζεται η συνάρτηση Lagrange.

$$\begin{aligned} L(U,V,H;P,Q,R) = \mathfrak{L}_\rho(S - UU^T) + \frac{\lambda}{2}||X - f(UH)V^T||_F^2 + < P, U > \\ + < Q, V > + < R, H > \end{aligned}$$

$$\begin{aligned} L(U,V,H;P,Q,R) = \mathfrak{L}_\rho(S - UU^T) + \frac{\lambda}{2}||X - f(UH)V^T||_F^2 + Tr(P^TU) \\ + Tr(Q^TV) + Tr(R^TH) \end{aligned} \tag{6.11}$$

,όπου P, Q, R οι βέλτιστοι πολλαπλασιαστές Lagrange (πίνακες ορισμένοι στο πεδίο τιμών των μεταβλητών) και U, V, H οι βέλτιστες μεταβλητές του προβλήματος.
Σημειώνεται, πως ισχύει για A,B πίνακες : $< A, B >= Tr(A^TB) = \sum_{i,j} A_{i,j}B_{i,j}$

- Παραγωγίζεται η συνάρτηση Lagrange ως προς όλες τις μεταβλητές.

$$\begin{aligned} \nabla_U L(U,V,H;P,Q,R) = \frac{\partial L}{\partial U} = & -2\rho SU - \lambda((XV) \odot f'(UH))H^T \\ & + 2\rho(UU^T \odot W)U + 2(1-\rho)(UU^T \odot W')U \\ & + \lambda[(f(UH)V^TV) \odot f'(UH)]H^T + P \end{aligned} \tag{6.12}$$

$$\nabla_V L(U,V,H;P,Q,R) = \frac{\partial L}{\partial V} = -\lambda X^T f(UH) + \lambda V f(UH)^T f(UH) + Q \tag{6.13}$$

$$\begin{aligned}\nabla_H L(U,V,H;P,Q,R) = \frac{\partial L}{\partial H} = &-\lambda U^T (f'(UH) \odot (XV)) \\ &+ \lambda U^T (f'(UH) \odot f(UH)) V^T V + R\end{aligned} \tag{6.14}$$

- Καταγράφονται οι συνθήκες Karush-Kahn-Tucker, που διέπουν το πρόβλημα.

1. **Συνθήκη Στασιμότητας/Συνθήκη Βελτιστότητας (Optimality Condition) :**

$$\nabla_U L(U,V,H;P,Q,R) = 0 \implies \frac{\partial L}{\partial U} = 0 \tag{6.15}$$

$$\nabla_V L(U,V,H;P,Q,R) = 0 \implies \frac{\partial L}{\partial V} = 0 \tag{6.16}$$

$$\nabla_H L(U,V,H;P,Q,R) = 0 \implies \frac{\partial L}{\partial H} = 0 \tag{6.17}$$

2. **Συνθήκη Πρωταρχικής Εφικτότητας (Primal Feasibility Condition) :**

$$\begin{cases} U \geq 0 \\ V \geq 0 \\ H \geq 0 \end{cases} \tag{6.18}$$

3. **Συνθήκη Εφικτότητας του Δυϊκού (Dual Feasibility Condition) :**

$$\begin{cases} P \geq 0 \\ Q \geq 0 \\ R \geq 0 \end{cases} \tag{6.19}$$

4. **Συνθήκη Συμπληρωματικής Χαλαρότητας (Complementary Slackness Condition) :**

$$\begin{cases} P \odot U = 0 \\ Q \odot V = 0 \\ R \odot H = 0 \end{cases} \tag{6.20}$$

Με την κατασκευή των τεσσάρων αυτών συνθηκών, ολοκληρώνεται η διαδικασία της βελτιστοποίησης. Στη συνέχεια, παρουσιάζονται αναλυτικά οι αποδείξεις των σχέσεων (6.12), (6.13) και (6.14).

Απόδειξη των σχέσεων (6.12), (6.13) και (6.14)

$$\begin{aligned}
L(U,V,H;P,Q,R) =& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}||X-f(UH)V^T||_F^2+Tr(P^TU)+Tr(Q^TV)\\
&+Tr(R^TH)\\
=& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}\left[Tr\left[(X-f(UH)V^T)(X-f(UH)V^T)^T\right]\right]\\
&+Tr(P^TU)+Tr(Q^TV)+Tr(R^TH)\\
=& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}\left[Tr\left[(X-f(UH)V^T)(X^T-(f(UH)V^T)^T\right]\right]\\
&+Tr(P^TU)+Tr(Q^TV)+Tr(R^TH)\\
=& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}\left[Tr\left[(X-f(UH)V^T)(X^T-Vf(UH)^T\right]\right]\\
&+Tr(P^TU)+Tr(Q^TV)+Tr(R^TH)\\
=& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}Tr\left[XX^T-XVf(UH)^T-f(UH)V^TX^T\right.\\
&\left.+f(UH)V^TVf(UH)^T\right]+Tr(P^TU)\\
&+Tr(Q^TV)+Tr(R^TH)\\
=& \mathfrak{L}_\rho(S-UU^T)+\frac{\lambda}{2}\left[Tr(XX^T)-Tr(f(UH)^TXV)-Tr(V^TX^Tf(UH))\right.\\
&\left.+Tr(Vf(UH)^Tf(UH)V^T)\right]+Tr(P^TU)+Tr(Q^TV)+Tr(R^TH) \qquad (6.21)
\end{aligned}$$

Σύμφωνα με τις ιδιότητες της παραγώγισης του ίχνους (Κεφάλαιο 2) έχουμε :

- Για την σχέση (6.13)

$$\begin{aligned}
\nabla_V L(U,V,H;P,Q,R)=\frac{\partial L}{\partial V} =& \frac{\lambda}{2}[-X^Tf(UH)-X^Tf(UH)+V(2f(UH)^Tf(UH))]+Q\\
=& \frac{\lambda}{2}[-2X^Tf(UH)+2V(f(UH)^Tf(UH))]+Q\\
=& -\lambda X^Tf(UH)+\lambda Vf(UH)^Tf(UH)+Q
\end{aligned}$$

- Για την σχέση (6.14)

$$\begin{aligned}
\nabla_H L(U,V,H;P,Q,R) =& \frac{\partial L}{\partial H}\\
=& \frac{\lambda}{2}[-U^Tf'(UH)\odot(XV)-U^Tf'(UH)\odot(XV)\\
&+2U^T(f'(UH)\odot f(UH))V^TV]+R\\
=& \frac{\lambda}{2}[-2U^Tf'(UH)\odot(XV)+2U^T(f'(UH)\odot f(UH))V^TV]+R\\
=& -\lambda U^T(f'(UH)\odot(XV))+\lambda U^T(f'(UH)\odot f(UH))V^TV+R
\end{aligned}$$

,όπου με $\odot$ θα συμβολίζεται ο "στοιχείο προς στοιχείο" πολλαπλασιασμός μεταξύ πινάκων.

- Για την σχέση (6.12)

  Χωρίζεται η σχέση (6.21) σε τρία μέρη και εφαρμόζεται παραγώγιση σε κάθε μία ξεχωριστά :

$$L_I = \mathfrak{L}_\rho(S - UU^T) \tag{6.21_I}$$

$$L_{II} = \frac{\lambda}{2}[Tr(XX^T) - Tr(f(UH)^T XV) - Tr(V^T X^T f(UH)) + Tr(Vf(UH)^T f(UH)V^T)] \tag{6.21_II}$$

$$L_{III} = Tr(P^T U) + Tr(Q^T V) + Tr(R^T H) \tag{6.21_III}$$

  – Για την σχέση (6.21_I)

$$\mathfrak{L}_\rho(S - UU^T) = \sum_{(i,j)\in E} \left(U_{ij}{U_{ij}}^T - S_{ij}\right)^2 + (1-\rho) \sum_{(i,j)\notin E} U_{ij}^2 {U_{ij}^T}^2 \tag{5.7}$$

  Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο ο παρών τύπος αφορά δύο κατηγορίες ακμών, τις ετικετοποιημένες/θετικές, που υπάρχουν στο σύνολο των ακμών, δηλαδή στον πίνακα γειτνίασης S και στις μη ετικετοποιημένες, που δεν υπάρχουν στο σύνολο των ακμών και κατ' επέκταση στον πίνακα S. Για να είναι εφικτό να απλοποιηθεί αυτή η μαθηματική έκφραση, πολλαπλασιάζεται κάθε άθροισμα με τα στοιχεία του πίνακα W και τα στοιχεία του πίνακα $W' = 1 - W$ αντίστοιχα. Ο πίνακας W ονομάζεται πίνακας μάσκα του S (Mask matrix) και ουσιαστικά για κάθε μη μηδενικό στοιχείο του S λαμβάνει στην αντίστοιχη θέση την τιμή 1 και στην περίπτωση μηδενικού στοιχείου, λαμβάνει την τιμή 0. Με τον τρόπο αυτό, πραγματοποιείται αυτόματα ο διαχωρισμός μεταξύ των στοιχείων που υπολογίζει το κάθε άθροισμα (1 ή 0), χωρίς να χρειάζεται να γίνει ο προσδιορισμός στους δείκτες των αθροισμάτων.

  Εφαρμόζονται πράξεις στην σχέση (6.21_I) προς απλοποίηση της :

$$\begin{aligned}
&\sum_{i,j} \rho \left(U_{ij}{U_{ij}}^T - S_{ij}\right)^2 + (1-\rho) \sum_{i,j} U_{ij}^2 {U_{ij}^T}^2 W'_{ij} = \\
&= \rho \sum_{ij} \left(U_{ij}^2 {U_{ij}^T}^2 - 2U_{ij}U_{ij}^T S_{ij} + S_{ij}^2\right) W_{ij} + (1-\rho) \sum_{ij} U_{ij}^2 {U_{ij}^T}^2 W'_{ij} = \\
&= \rho \sum_{ij} U_{ij}^2 {U_{ij}^T}^2 W_{ij} - 2\rho \sum_{ij} U_{ij}U_{ij}^T S_{ij} W_{ij} + \rho \sum_{ij} S_{ij}^2 W_{ij} + (1-\rho) \sum_{ij} U_{ij}^2 {U_{ij}^T}^2 W'_{ij} = \\
&= \rho \sum_{ij} U_{ij}U_{ij}U_{ij}^T U_{ij}^T W_{ij} - 2\rho \sum_{ij} U_{ij}U_{ij}^T S_{ij} W_{ij} + \rho \sum_{ij} S_{ij}S_{ij} W_{ij} + (1-\rho) \sum_{ij} U_{ij}U_{ij}U_{ij}^T U_{ij}^T W'_{ij} = \\
&\rho Tr\left(U^T UUU^T \odot W\right) - 2\rho Tr\left(UU^T S \odot W\right) + \rho Tr\left(S^T S \odot W\right) + (1-\rho) Tr\left(U^T UUU^T \odot W'\right)
\end{aligned}$$

Σύμφωνα με το παραπάνω αποτέλεσμα τίθεται :

$$L_{IV} = \rho Tr\left(U^T UUU^T \odot W\right) - 2\rho Tr\left(UU^T S \odot W\right) + \rho Tr\left(S^T S \odot W\right) + (1-\rho) Tr\left(U^T UUU^T \odot W'\right) \tag{6.22}$$

– Παραγωγίζοντας την σχέση (6.22)

$$\frac{\partial L_{IV}}{\partial U} = -2\rho SU + 2\rho\left(UU^T \odot W\right)U + 2\left(1-\rho\right)\left(UU^T \odot W'\right)U \tag{6.23}$$

– Παραγωγίζοντας την σχέση (6.21_II)

$$\begin{aligned}
\frac{\partial L_{II}}{\partial U} = & \frac{\lambda}{2}[Tr(XX^T) - Tr(f(UH)^T XV) - Tr(V^T X^T f(UH)) + Tr(Vf(UH)^T f(UH)V^T)] = \\
= & \frac{\lambda}{2}[-(XV) \odot f'(UH)H^T - (XV) \odot f'(UH)H^T + 2(f(UH)V^T V) \odot f'(UH)H^T] = \\
= & \frac{\lambda}{2}[(-2(XV) \odot f'(UH))H^T + 2(f(UH)V^T V) \odot f'(UH)H^T] = \\
= & -\lambda((XV) \odot f'(UH))H^T + \lambda(f(UH)V^T V) \odot f'(UH)H^T
\end{aligned} \tag{6.24}$$

– Παραγωγίζοντας την σχέση (6.21_III)

$$\frac{\partial L_{III}}{\partial U} = P \tag{6.25}$$

6.3.1 Κανόνες Ανανέωσης

Με σκοπό να παραγάγουμε έναν αλγόριθμο, ο οποίος θα καταλήξει σε κάποιο βέλτιστο ακρότατο σημείο, στην NAGC χρησιμοποιείται η μέθοδος των Lee και Seung [31]. Συγκεκριμένα, θα βασιστούμε στον τύπο (4.6) :

$$\theta \to \theta \frac{\nabla_\theta^- f(\theta)}{\nabla_\theta^+ f(\theta)}$$

- Για το U, χρησιμοποιούμε τις σχέσεις (6.12) και (6.20) και προκύπτει :

$$\begin{aligned} U \longleftarrow & U \odot [2\rho SU + \lambda((XV) \odot f'(UH))H^T] \oslash \\ & \big[2\rho(UU^T \odot W)U + 2(1-\rho)(UU^T \odot W')U + \\ & \lambda((f(UH)V^TV) \odot f'(UH))H^T\big] \end{aligned} \tag{6.26}$$

- Για το V, χρησιμοποιούμε τις σχέσεις (6.13) και (6.20) και προκύπτει :

$$V \longleftarrow V \odot (X^T f(UH)) \oslash (V f(UH)^T f(UH)) \tag{6.27}$$

- Για το H, χρησιμοποιούμε τις σχέσεις (6.14) και (6.20) και προκύπτει :

$$H \longleftarrow H \odot \big[U^T(f'(UH) \odot (XV)\big] \oslash \big[U^T(f'(UH) \odot f(UH))V^TV\big] \tag{6.28}$$

, όπου με $\oslash$ θα συμβολίζουμε την διαίρεση "στοιχείο προς στοιχείο" μεταξύ πινάκων.

Καταλήγουμε στους πολλαπλασιαστικούς κανόνες ανανέωσης της NAGC. Σκοπός αυτής της επαναληπτικής διαδικασίας είναι η βελτιστοποίηση, όσο το δυνατόν είναι εφικτό να πραγματοποιηθεί, των πινάκων V και H, ώστε τελικά να βελτιστοποιηθεί ο πίνακας U, ο οποίος αποτελεί τον βασικό πίνακα της μεθόδου και είναι αυτός που θα καθορίσει την τελική μορφή της ομαδοποίησης. Δεδομένης της μη κυρτότητας δεν μπορεί να αναμένει κανείς τον προσδιορισμό ενός ολικού ελάχιστου, αλλά την προσέγγιση ενός τοπικού ακρότατου ή διαφορετικά ενός καλού στάσιμου σημείου. Συμπληρωματικά, δεν μπορεί να αποδειχθεί η μονοτονική σύγκλιση για την προσέγγιση των παραμέτρων με την μέθοδο που χρησιμοποιήθηκε, αλλά μπορούμε να αναμένουμε μια ιδιότητα σύγκλισης για κάθε ακραίο σημείο, να είναι στάσιμο. Σε αυτό το σημείο, διαφαίνεται η σημασία της αρχικοποίησης και της κατάλληλης προετοιμασίας των δεδομένων για τα αποτελέσματα της εκάστοτε μεθόδου. Όσο καλύτερα είναι προετοιμασμένα τα δεδομένα, τόσο πιο σύντομα και με μεγαλύτερη πιθανότητα θα επιτευχθεί η εύρεση των βέλτιστων ή προσεγγιστικά βέλτιστων τιμών των μεταβλητών, που αναζητούμε.

6.4 Αλγόριθμος NAGC

Ο αλγόριθμος δέχεται σαν είσοδο :

- Τον πίνακα γειτνίασης S.
- Τον πίνακα χαρακτηριστικών X.
- Τις βέλτιστες παραμέτρους k_1, k_2, λ και ρ.
- Τον αριθμό των επαναλήψεων t, που θα πραγματοποιήσει ο αλγόριθμος.

Ο ρόλος και ο τρόπος υπολογισμού των παραμέτρων k_1, k_2, λ και ρ για κάθε δείγμα δεδομένων αναλύονται στο επόμενο κεφάλαιο. Οι πίνακες S και X προετοιμάζονται κατάλληλα (κανονικοποίηση). Στη συνέχεια, αρχικοποιούνται :

- Ο πίνακας U.
- Ο πίνακας V.
- Ο πίνακας H.

Οι πίνακες U και V αρχικοποιούνται με τις μεθόδους k-means++ ή agglomerative clustering, ενώ ο πίνακας H αρχικοποιείται με τυχαίες τιμές.

Για t επαναλήψεις ($t' < t$) εκτελούνται :

- $U^{(t'+1)} \longleftarrow update(U^{(t')})$, σύμφωνα με την σχέση (6.26).
- $V^{(t'+1)} \longleftarrow update(V^{(t')})$, σύμφωνα με την σχέση (6.27).
- $H^{(t'+1)} \longleftarrow update(H^{(t')})$, σύμφωνα με την σχέση (6.28).

Ο αλγόριθμος καταλήγει στον πίνακα U.

Για n επαναλήψεις (αριθμός γραμμών) προσπελαύνεται ο πίνακας U. Σε κάθε γραμμή επιλέγεται το μέγιστο στοιχείο και με αυτόν τον τρόπο γίνεται η αντιστοίχιση κάθε κόμβου σε μία ομάδα.

- $c_{n'} \longleftarrow \underset{l}{\operatorname{argmax}} \left\{ u_{n',l} | l = (1, \ldots, k_1) \right\}$

Η πολυπλοκότητα του αλγορίθμου, σύμφωνα με την δημοσίευση των [3], υπολογίζεται ως εξής :

- SNMF $O(n^2kt)$
- Πολλαπλασιαστικοί κανόνες ανανέωσης $O((n^2 + mn)kt)$

, όπου $k = \max(k_1, k_2)$ με $k << n$.

Παρά τις αλλαγές, οι οποίες εφαρμόστηκαν στην αρχικοποίηση των δεδομένων, καταλήγουμε στο συμπέρασμα πως η πολυπλοκότητα της μεθόδου NAGC δεν επηρεάστηκε. Αυτό συμβαίνει καθώς η πολυπλοκότητα του αλγόριθμου k-means, που χρησιμοποιήθηκε στην αρχική δημοσίευση Maekawa, Takeuchi και Onizuka [3] είναι της τάξης $O(n^2)$ και ομοίως η πολυπλοκότητα ενός agglomerative αλγορίθμου μπορεί με κατάλληλη δομή δεδομένων να είναι, επίσης, της τάξης $O(n^2)$.

Αλγόριθμος 1: NAGC

Πηγή αλγόριθμου: Maekawa, Takeuchi και Onizuka [3]

Είσοδος : S, X, k_1, k_2, λ, t, ρ
Έξοδος : Τελική ανάθεση σε κοινότητες C

Προετοιμασία: S, X
Αρχικοποίηση: U, V, H
while $t' < t$ **do**
    // διαδοχική ανανέωση μεταβλητών
    $U^{(t'+1)} \longleftarrow update(U^{(t')})$ σύμφωνα με (6.26).
    $V^{(t'+1)} \longleftarrow update(V^{(t')})$ σύμφωνα με (6.27).
    $H^{(t'+1)} \longleftarrow update(H^{(t')})$ σύμφωνα με (6.28).
end
while $n' < n$ **do**
    // ανάθεση κόμβων σε κοινότητες
    $c_{n'} \longleftarrow \underset{l}{\operatorname{argmax}} \{u_{n',l} | l = (1, \ldots, k_1)\}$
end

Κεφάλαιο 7

Στοιχεία Πειραματικής Διαδικασίας

Πριν καταλήξει κανείς σε συμπεράσματα σχετικά με τις δυνατότητες ενός αλγόριθμου, υποχρεούται να πραγματοποιήσει δοκιμές και πειράματα σε πολλά και διαφορετικά επίπεδα και να εφαρμόσει διάφορες μεθόδους ελέγχου των χαρακτηριστικών του. Μέσω της πειραματικής διαδικασίας, προκύπτουν χρήσιμα συμπεράσματα τα οποία, εφόσον χρησιμοποιηθούν κατάλληλα, μπορούν να οδηγήσουν σε πιο εξελιγμένες και βελτιστοποιημένες μορφές ενός αλγορίθμου. Κρίνεται απαραίτητο, πριν την τελική συμπερασματολογία να ληφθούν υπόψιν κάποιοι καταλυτικοί παράγοντες, όπως οι βασικοί στόχοι της μεθόδου, τα δείγματα δεδομένων που χρησιμοποιήθηκαν, οι παράμετροι του προβλήματος και ο τρόπος επιλογής τους κ.α. Επιπρόσθετα, η αξιολόγηση αυτή των αποτελεσμάτων, θα πρέπει να βασιστεί σε συγκεκριμένα μεγέθη αξιολόγησης. Τα μεγέθη αυτά, θα πρέπει να επιλέγονται με βάση συγκεκριμένα κριτήρια : να διέπονται όσο το δυνατόν περισσότερο από αμεροληψία και να μπορούν να χαρακτηρίσουν τα αποτελέσματα πληθώρας αλγορίθμων με ορθό τρόπο. Ένας ακόμη τρόπος ελέγχου αποδοτικότητας είναι και η σύγκριση με άλλους γνωστούς και σύγχρονους αλγορίθμους, που έχουν αναπτυχθεί σε όμοιο ερευνητικό πεδίο με την χρήση των ίδιων δειγμάτων δεδομένων και μεγεθών αξιολόγησης, ζήτημα που πργαματεύτηκε η αρχική δημοσίευση. Παράλληλα, σημαντικές είναι και οι συγκρίσεις ενός αλγορίθμου με διαφορετικές παραλλαγές του, με σκοπό την επιλογή τελικά της καλύτερης από αυτές, μια ανάλυση που πραγματοποιήθηκε στην παρούσα εργασία και παρουσιάζεται στην συνέχεια. Η πολυπλοκότητα είναι, επίσης, ένας βασικός παράγοντας, ο οποίος λαμβάνεται υπόψιν στις σύγχρονες μελέτες και πέρα από την υπολογιστική πολυπλοκότητα ενός αλγορίθμου, μεγάλο ρόλο διαδραματίζει στο τελικό χρονικό αποτέλεσμα και η ισχύς των πόρων, που χρησιμοποιούνται για την εκτέλεση του εκάστοτε πειράματος. Με την πειραματική εκτέλεση της NAGC καλούμαστε να αναπαράγουμε και να ερμηνεύσουμε αποτελέσματα, σημαντικά για την τελική απόφανση σχετικά με την αποδοτικότητά της, αλλά και εν γένει για την σημασία της ομαδοποίησης. Στο παρόν κεφάλαιο, θα αναλυθούν οι προαναφερθέντες παράγοντες : τα δείγματα δεδομένων και ο τρόπος προετοιμασίας και επεξεργασίας τους, οι μετρικές αξιολόγησης των αποτελεσμάτων, που έχουν επιλεχθεί, οι βέλτιστες παράμετροι, η φυσική τους σημασία, αλλά και ο τρόπος επιλογής τους. Οι τελικές παρατηρήσεις, όπως αναλύονται στην συνέχεια αξιολογούνται συνολικά και συνδυαστικά.

7.1 Δείγματα Δεδομένων

Τα δείγματα δεδομένων που χρησιμοποιούνται στην αρχική δημοσίευση της μεθόδου NAGC έχουν συγκεκριμένη μορφολογία και αφορούν ομαδοποίηση κειμένου (text clustering). Πιο συγκεκριμένα, για την δόμηση τους χρησιμοποιείται μια παραλλαγή της μεθόδου bag of words [42]. Η λογική της μεθόδου αυτής, για παράδειγμα σε ένα κείμενο, είναι να καταγράφεται για κάθε λέξη, η συχνότητα εμφάνιση της. Για την περίπτωση της NAGC, δεν διατηρείται ο αριθμός που υποδεικνύει την συχνότητα εμφάνισης, αλλά καταγράφεται μόνο η ύπαρξη (1) ή η ανυπαρξία (0) μιας λέξης στο αντίστοιχο κείμενο. Για τις δοκιμές του αλγορίθμου, χρησιμοποιήθηκαν τέσσερα πολύ γνωστά δείγματα δεδομένων στην προαναφερθείσα μορφή. Τα δείγματα αυτά, για να γίνουν δεκτά από τον αλγόριθμο προσαρμόζονται κατάλληλα, ώστε να αναπαριστούν μη κατευθυνόμενους, χωρίς βάρη κόμβων ή ακμών γράφους με χαρακτηριστικά, που δεν περιλαμβάνουν βρόχους. Συμπληρωματικά στην επεξεργασία, έχουν αφαιρεθεί οι ενδιάμεσες λέξεις και εκείνες με συχνότητα εμφάνισης μικρότερης του δέκα. Επίσης, κάθε κόμβος συνδέεται με τουλάχιστον έναν άλλον κόμβο. Πρέπει να σημειωθεί, πως σε όλες τις περιπτώσεις διαθέτουμε την πραγματική ομαδοποίηση (groundtruth). Τα δείγματα δεδομένων παρουσιάζονται αναλυτικά παρακάτω.

1. **WebKB :** Αποτελείται από 877 ιστοσελίδες (κόμβοι) από τέσσερα πανεπιστημιακά ιδρύματα με αντικείμενο την επιστήμη των ηλεκτρονικών υπολογιστών : Πανεπιστήμιο Κορνέλ, Πανεπιστήμιο του Τέξας, Πανεπιστήμιο της Ουάσινγκτον και Πανεπιστήμιο του Ουισκόνσιν. Μεταξύ των ιστοσελίδων υπάρχουν 1480 υπερσύνδεσμοι (ακμές). Οι ιστοσελίδες είναι κατανεμημένες σε πέντε κατηγορίες : Course, Faculty, Student, Project, Staff. Περιέχει 1703 μοναδικές λέξεις. [43]

2. **Cora :** Αποτελείται από 2708 επιστημονικές δημοσιεύσεις (κόμβοι) σχετικές με την μηχανική μάθηση, οι οποίες συνδέονται μεταξύ τους με 5278 συνδέσμους αναφοράς (ακμές) (citation links). Οι δημοσιεύσεις κατανέμονται σε επτά κατηγορίες : Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory. Περιέχει 1433 λέξεις. [44]

3. **Citeseer :** Αποτελείται από 3312 δημοσιεύσεις, οι οποίες συνδέονται με 4660 συνδέσμους αναφοράς (ακμές). Οι δημοσιεύσεις κατανέμονται σε 6 κατηγορίες : Agents, AI, DB, IR, ML, HCI. Περιέχει 3703 μοναδικές λέξεις. [45]

4. **PolBlog :** Αποτελείται από 1490 ιστοσελίδες πολιτικού περιεχομένου των Ηνωμένων Πολιτειών, οι οποίες συνδέονται μεταξύ τους με 16630 υπερσυνδέσμους (ακμές). Οι ιστοσελίδες κατανέμονται σε δύο κατηγορίες : Liberal , Conservative. Περίεχει 7 μοναδικές λέξεις. [46]

Οι κατηγορίες αφορούν τις κοινότητες των οποίων στοχεύουμε να προσεγγίσουμε, όσο το δυνατό ακριβέστερα το περιεχόμενο. Οι δημοσιεύσεις-ιστοσελίδες αποτελούν τους κόμβους του αντίστοιχου γραφήματος και οι αναφορές-υπερσύνδεσμοι τις ακμές του. Τέλος, οι μεμονωμένες λέξεις αποτελούν τα χαρακτηριστικά των κόμβων του γραφήματος. Οι αριθμοί που χαρακτηρίζουν τους κόμβους, τις ακμές και τα χαρακτηριστικα καταγράφονται, σύμφωνα με την αρχική δημοσίευση των Maekawa, Takeuchi και Onizuka [3].

7.2 Μεγέθη Αξιολόγησης

Τα μεγέθη αξιολόγησης αποτελούν ένα από τα βασικότερα μέσα για τον υπολογισμό της απόδοσης ενός αλγορίθμου και την σύγκριση του με άλλους και παράλληλα αποτελούν σημαντικό εργαλείο για την εκτενέστερη κατανόηση των δεδομένων υπό μελέτη. Είναι μεγάλης σημασίας, να προσδιοριστούν οι σωστές μετρικές προς αποφυγή λανθασμένης συμπερασματολογίας. Στην ευρύτερη μελέτη της NAGC, μίας μεθόδου που δεν επιτρέπει την ανίχνευση επικαλυπτόμενων ομάδων, χρησιμοποιούνται τα εξής μεγέθη :

- **Average Entropy :** Χρησιμοποιείται για να χαρακτηρίσει ένα δείγμα ως προς την ομοιότητα των χαρακτηριστικών εντός των κοινοτήτων. Ορίζεται ως [3] :

$$\text{Average Entropy} = \sum_{i=1}^{m}\sum_{j=1}^{k}\frac{|C_j|}{nm}H(\alpha_i, C_j) \tag{7.1}$$

, όπου $H(\alpha_i, C_i)$ είναι η εντροπία, όπως ορίζεται στην θεωρία πληροφορίας για το χαρακτηριστικό α_i στην ομάδα C_j, n ο αριθμός των κόμβων, m ο αριθμός των χαρακτηριστικών και k ο αριθμός των ομάδων. Η εντροπία, γενικά, αποτελεί μέτρο της αβεβαιότητας σε ένα σύστημα. Ορίζεται ως [47] :

$$H(\alpha_i, C_j) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{7.2}$$

, για την ομαδοποίηση $p_i = p(\alpha_i|C_j)$, δηλαδή η πιθανότητα να επιλεγεί το α_i σε μία τυχαία αναζήτηση στο σύνολο C_j. Συχνά, στην στατιστική συναντάται ως $H(X)$ και αφορά τυχαίες μεταβήτές και ορίζεται με τον εξής τρόπο [48] :

$$\text{H(X)} = -\sum_{i=1}^{n} p_X(x) log_2 p_X(x) \tag{7.3}$$

, όπου $p_X(x), x \in \mathbb{X}$ η συνάρτηση μάζας πιθανότητας της τυχαίας διακρητής μεταβλητής X, η οποία παίρνει τιμές στο $\mathbb{X}$. Ομοίως, για τον υπολογισμό της εντροπίας μεταξύ δύο τυχαίων διακρητών μεταβλητών (από κοινού εντροπία) έστω $\in \mathbb{X}$ και $Y \in \mathbb{Y}$, χρησιμοποιείται ο παρακάτω τύπος [48] :

$$\text{H(X,Y)} = -\sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} p(x,y) log_2 p(x,y) \tag{7.4}$$

, όπου $p(x,y), x \in \mathbb{X}$ και $y \in \mathbb{Y}$ η από κοινού συνάρτηση μάζας πιθανότητας των τυχαίων διακρητών μεταβλητών X και Y.

Λαμβάνει τιμές στο διάστημα $[0, 1]$. Όταν αυτό το μέγεθος λαμβάνει μικρή τιμή, υποδεικνύει μεγάλη ομοιότητα χαρακτηριστικών εντός των διακεκριμένων από την μέθοδο κοινοτήτων και μικρή ομοιότητα μεταξύ διαφορετικών κοινοτήτων.

- **Modularity :** Χρησιμοποιείται για να χαρακτηρίσει ένα δείγμα με βάση την πυκνότητα των ακμών εντός των κοινοτήτων. Ορίζεται ως :

$$Q = \frac{1}{2z}\sum_{ij}[\ A_{ij} - \frac{k_i k_j}{2z}]\ \delta(c_i, c_j) \tag{7.5}$$

, όπου z ο αριθμός των ακμών στον γράφο, $k_i k_j$ οι βαθμοί των κόμβων i, j, A ο πίνακας γειτνίασης,c_i, c_j οι ομάδες στις οποίες ανήκουν οι κόμβοι i, j.

Με $\delta(c_i, c_j)$ συμβολίζεται το δέλτα του Kronecker :

$$\delta(c_i, c_j) = \begin{cases} 1, \text{ για } c_i = c_j \\ 0, \text{ για } c_i \neq c_j \end{cases}$$

Λαμβάνει τιμές στο διάστημα $[-1, 1]$. Όταν αυτό το μέγεθος λαμβάνει μεγάλη τιμή, υποδεικνύει μεγάλη πυκνότητα ακμών εντός των διακεκριμένων από την μέθοδο ομάδων και μικρή πυκνότητα ακμών μεταξύ διαφορετικών ομάδων. [49],[5]

Τα δύο αυτά μεγέθη φαίνεται πως δεν αποδίδουν τόσο καλά σε περιπτώσεις ανίχνευσης κοινοτήτων με ταυτόχρονη αξιολόγηση των χαρακτηριστικών των κόμβων και της τοπολογικής τους διάταξης και έτσι εισάγονται για την NAGC επιπλέον τα μεγέθη Adjusted Rand Index (ARI) και Adjusted Mutual Information (AMI). Στις περιπτώσεις των ARI και AMI είναι απαραίτητο, να υπάρχει η πραγματική ομαδοποίηση (groundtruth), ώστε να πραγματοποιηθεί σύγκριση. Πριν αναλυθούν οι ορισμοί των μεγεθών αυτών, χρειάζεται να εισαχθούν κάποιες βασικές έννοιες.

– **Mutual Information (MI) :** Σαν μέγεθος προέρχεται από την Θεωρία Πληροφορίας και παρέχει την δυνατότητα σύγκρισης δύο τυχαίων μεταβλητών. Στην διαδικασία εύρεσης κοινοτήτων χρησιμοποιείται για την σύγκριση μεταξύ δύο ομαδοποιήσεων και στην περίπτωση της NAGC, μεταξύ της πρόβλεψης και της πραγματικής ομαδοποίησης. Ανάμεσα στους διάφορους τύπους που χαρακτηρίζουν αυτό το μέγεθος, επικρατέστερος και καταλληλότερος είναι ο εξής :

$$MI(A, B) = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{n_{ij}}{N} log \frac{n_{ij} N}{a_i b_j} \tag{7.6}$$

, όπου A, B δύο ομαδοποιήσεις και με βάση το παρακάτω σχήμα:

Πίνακας 7.1
Πηγή αρχικού σχήματος Bailey [50]

	b_1	$\cdots$	b_j	$\cdots$	b_c
α_1	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1c}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	n_{ic}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_r	n_{r1}	$\cdots$	n_{rj}	$\cdots$	n_{rc}

, με n_{ij} ο αριθμός των κοινών κόμβων μεταξύ των δύο ομάδων, όπως υποδεικνύονται από την εκάστοτε γραμμή και στήλη, c και r ο αριθμός των ομάδων σε κάθε ομαδοποίηση, $a_i = \sum_j n_{ij}$, $b_j = \sum_i n_{ij}$ τα αθροίσματα ως προς γραμμή και στήλη αντίστοιχα και N ο αριθμός των κόμβων. Η τιμή αυτού του μεγέθους, όμως, μπορεί να είναι οποιοσδήποτε μη αρνητικός αριθμός, κι έτσι τα αποτελέσματα του δεν είναι εύκολα ερμηνεύσιμα. Για τον λόγο αυτό έχουν προταθεί κάποιες κανονικοποιήσεις του μεγέθους, ώστε να περιοριστούν οι τιμές στο διάστημα [0,1]. Πριν εισαχθούν τα μεγέθη αυτά, χρειάζεται να εξηγηθεί, επίσης, η έννοια του Expected Mutual Information. [50]

– **Expected Mutual Information E(MI) :**

$$E[MI(A,B)] = \sum_{i,j} \sum_{n_{ij}} \frac{n_{ij}}{N} log \frac{n_{ij} N}{a_i b_j} P(n_{ij}) \tag{7.7}$$

, όπου $P(n_{ij})$ γνωστή, υπεργεωμετρική κατανομή του n_{ij}. [50]

- **Normalized Mutual Information (NMI) :** Ορίζεται ως :

$$NMI = \frac{MI(A,B)}{\max MI(A,B)} \tag{7.8}$$

Ανάλογα με το τι θα θεωρήσουμε για άνω φράγμα της συνάρτησης MI(A,B) μπορούμε να λάβουμε και διαφορετικό τύπο. Συχνότερα συναντάμε :

$$NMI(A,B) = \frac{MI(A,B)}{H(A,B)} \tag{7.9}$$

Λαμβάνει τιμές στο διάστημα $[0,1]$. [50]

- **Adjusted Mutual Information (AMI) :** Ορίζεται ως :

$$AMI(A,B) = \frac{MI(A,B) - E[MI(A,B)]}{\max MI(A,B) - E[MI(A,B)]} \tag{7.10}$$

Ανάλογα με το τι θα θεωρήσουμε για άνω φράγμα της συνάρτησης MI(A,B) μπορούμε να λάβουμε και διαφορετικό τύπο. Συχνότερα συναντάμε :

$$AMI(A,B) = \frac{MI(A,B) - E[MI(A,B)]}{\max H(A), H(B) - E[MI(A,B)]} \tag{7.11}$$

Λαμβάνει τιμές στο διάστημα $[0,1]$. [50]

- **Adjusted Rand Index :** Ομοίως με το μέγεθος MI, βασίζεται στον πίνακα 7.1 και ορίζεται ως :

$$ARI(A,B) = \frac{\sum_{i,j} \binom{n_{ij}}{2} - [\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{\alpha_i}{2} + \sum_j \binom{\beta_j}{2}] - [\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}]/\binom{N}{2}} \tag{7.12}$$

Λαμβάνει τιμές στο διάστημα $[0,1]$. Η τιμή 1 επιτυγχάνεται μόνο στην περίπτωση, που η πρόβλεψη της ανάθεσης των κόμβων σε κοινότητες είναι ακριβώς ίδια με την πραγματική (groundtruth) ανάθεση. [51] Αποτελεί προσαρμοσμένη μορφή του μεγέθους Random Index, με τρόπο τέτοιο, ώστε για τυχαίες ομαδοποιήσεις να παίρνει την τιμή 0 (corrected-for-chance).

Τα προσαρμοσμένα μεγέθη (adjusted measures) προτιμώνται, καθώς για τυχαίες ομαδοποιήσεις λαμβάνουν τιμές κοντά στο μηδέν. [3]

7.3 Παράμετροι

Σύμφωνα με την τελική συνάρτηση της NAGC (6.1) κρίνεται απαραίτητο να προσδιοριστούν εξ αρχής οι παράμετροι k_1, k_2, λ και ρ, με βέλτιστο τρόπο, ώστε να επιτευχθούν ορθά και κατά το δυνατόν ακριβή αποτελέσματα. Με k συμβολίζεται ο αριθμός των πραγματικών (τοπολογικών k_1) κοινοτήτων, με k_2 ο αριθμός των κοινοτήτων, όπως αυτές θα προέκυπταν με βάση τα χαρακτηριστικά, $\lambda \geq 0$ ο ισοσταθμιστικός παράγοντας της βαρύτητας, που αποδίδεται κάθε φορά μεταξύ της τοπολογικής διάταξης και της μορφολογίας των χαρακτηριστικών ανάλογα με την δομή των εκάστοτε δεδομένων υπό μελέτη και $\rho \in [0, 1]$ η μεροληψία, που εφαρμόζεται ανάμεσα στις θετικές και τις μη ετικετοποιημένες ακμές (PU Learning). Η τακτική επιλογής παραμέτρων, η οποία ακολουθείται στην αρχική δημοσίευσή Maekawa, Takeuchi και Onizuka [3] και αναπαράχθηκε στην παρούσα εργασία είναι η εξής :

- Η παράμετρος k (k_1) δίνεται χειροκίνητα, βάσει των groundtruth δεδομένων.
- Η παράμετρος k_2 επιλέγεται από το σύνολο :

$$\{k, 5, 7, 10, 15, 20\}$$

- Η παράμετρος λ επιλέγεται από το σύνολο :

$$\{10^{-10}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100, 1000\}$$

- Ἡ παράμετρος ρ επιλέγεται από το σύνολο :

$$\{0.5, 0.55, 0.75, 0.95, 0.995\}$$

Για να προσδιοριστούν οι κατάλληλες παράμετροι από τα προαναφερθέντα εφικτά διαστήματα, για κάθε δείγμα δεδομένων (WebKB, Cora, Citeseer, PolBlog), για κάθε αρχικοποίηση (k-means++ clustering, agglomerative clustering) και για κάθε μη γραμμική συνάρτηση (sigmoid, ReLU), πραγματοποιήθηκαν όλοι οι πιθανοί συνδυασμοί των k_2, λ, ρ για δεδομένα k (k_1). Για κάθε έναν συνδυασμό, υπολογίστηκαν τα μεγέθη Entropy, Modularity και ARI, που αναλύθηκαν στην προηγούμενη ενότητα. Με γνώμονα το σημείο που επιτυγχάνεται η μέγιστη τιμή των ARI και Modularity, γίνεται η επιλογή των καλύτερων συνδυασμών τους (ένας για κάθε περίπτωση), ο οποίος εν τέλει χρησιμοποιείται για την τελική εκτέλεση του αλγορίθμου. Η διαδικασία αυτή, υπήρξε αρκετά χρονοβόρα, εξαιτίας του μεγάλου αριθμού των συνδυασμών που προκύπτουν, αλλά είναι εφικτό να εξαχθούν αποτελέσματα μεγάλης ακρίβειας με μία εκτέλεση της. Οι παράμετροι αυτές, διαδραματίζουν σημαίνοντα ρόλο στην τελική ανίχνευση των κοινοτήτων και καθορίζουν σε μεγάλο βαθμό το αποτέλεσμα. Η ανάλυση των αποτελεσμάτων και η συσχέτιση τους με την επιλογή των παραμέτρων πραγματοποιείται στο επόμενο κεφάλαιο.

Κεφάλαιο 8

Μελέτη Αποτελεσμάτων

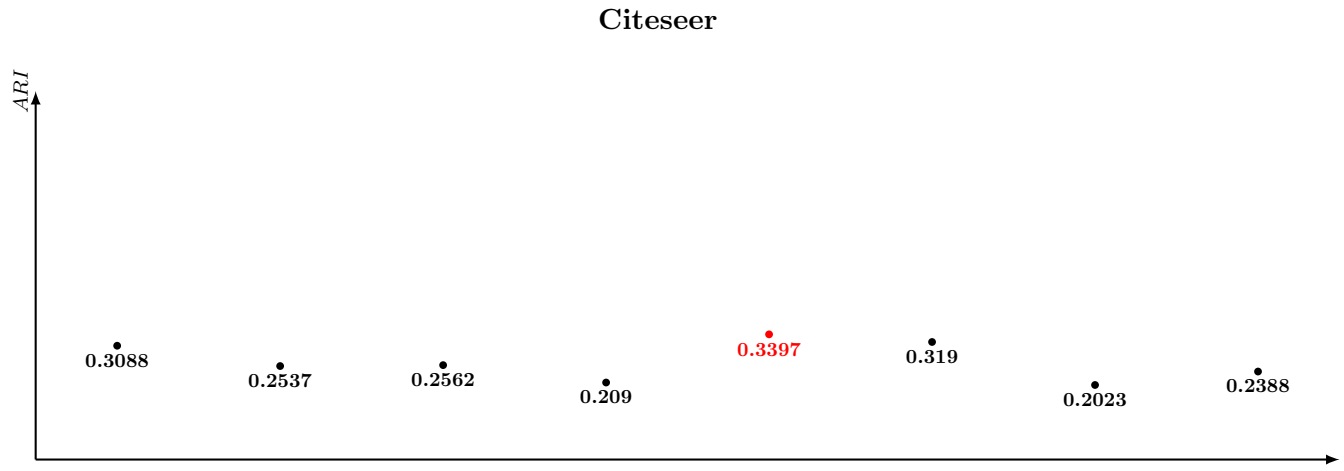
Στην αρχική δημοσίευση, των Maekawa, Takeuchi και Onizuka [3] πραγματοποιήθηκε μια εκτενής μελέτη της απόδοσης της μεθόδου NAGC, με αλγόριθμο αρχικοποίησης των πινάκων U και V αυτόν της μεθόδου k-means++ και μη γραμμική συνάρτηση μετατροπής του γινομένου πινάκων UH, την σιγμοειδή. Η μελέτη αυτή αξιολογήθηκε κυρίως με τα μεγέθη ARI και AMI και έγιναν συγκρίσεις με βάση τις τιμές, που αυτά λαμβάνουν σε άλλες παρόμοιες μεθόδους ανίχνευσης κοινοτήτων σε γράφους με χαρακτηριστικά κόμβων. Επίσης, σχολιάστηκε η επιρροή των παραμέτρων ρ, λ και k_2 στο τελικό αποτέλεσμα και ο τρόπος, που αλληλεπιδρούν μεταξύ τους. Στην παρούσα διπλωματική εργασία, σαν επέκταση των ήδη γνωστών αποτελεσμάτων, τα οποία επί τω πλείστον αναπαράχθηκαν, εξετάστηκε η απόδοση του αλγορίθμου σε σχέση με την ακριβέστερη ανάθεση των κόμβων στις ομάδες, αφού εφαρμόστηκαν σε αυτόν τροποποιήσεις. Η πρώτη τροποποίηση αφορά την αρχικοποίηση των πινάκων U και V με διάφορες γνωστές και αποδοτικές μεθόδους ομαδοποίησης και η δεύτερη την δοκιμή της συνάρτησης ReLU, ως συνάρτηση μετατροπής του γινομένου πινάκων UH. Με βάση αυτές τις παραλλαγές του αρχικού αλγορίθμου, στις επόμενες ενότητες παρουσιάζονται και σχολιάζονται τα αποτελέσματα, καθώς συγκρίνονται μεταξύ τους με βάση δύο μετρικές : ARI και Modularity. Παράλληλα, κάθε φορά, για κάθε συνδυασμό αλγορίθμου αρχικοποίησης και συνάρτησης μετατροπής, αναγράφονται οι βέλτιστες παράμετροι για την εκάστοτε μετρική, όπως υπολογίστηκαν στην ενότητα 7.3. Παρακάτω, παρουσιάζονται όλοι οι πιθανοί συνδυασμοί μεταξύ των αλγορίθμων αρχικοποίησης και των συναρτήσεων μετατροπής. Αυτοί είναι :

- k-means++ - sigmoid
- k-means++ - ReLU
- agglomerative single linkage - sigmoid
- agglomerative single linkage - ReLU
- agglomerative complete linkage - sigmoid
- agglomerative complete linkage - ReLU
- agglomerative average linkage - sigmoid
- agglomerative average linkage - ReLU

Πρέπει να αναφερθεί, πως, επίσης, πραγματοποιήθηκαν πειράματα, με σκοπό την αρχικοποίηση των πινάκων U και V με τους αλγορίθμους Spectral Clustering και DBSCAN, ώστε να μην δοθεί ο αριθμός k_1, δηλαδή ο αριθμός των τοπολογικών ομάδων, ως είσοδος. Φάνηκε, πως η δομή των δεδομένων στην περίπτωση της παράθεσης των χαρακτηριστικών των κόμβων (bag of words), όπως περιγράφηκε στην ενότητα 7.1, δεν ήταν κατάλληλη γα την απόδοση αυτών των δύο αλγορίθμων, καθώς είτε στην περίπτωση του DBSCAN προβλεπόταν μόνο μια κοινότητα, είτε στην περίπτωση του Spectral Clustering ταυτιζόταν ο αριθμός k_1 με τον αριθμό k_2, παρά την είσοδο που δινόταν, δηλαδή τον αριθμό των ομάδων, όπως προκύπτουν με βάση τα χαρακτηριστικά των κόμβων. Στα αποτελέσματα, που παρουσιάζονται στη συνέχεια, προτιμώνται οι συνδυασμοί για τους οποίους η παράμετρος λ λαμβάνει τιμές διάφορες του μηδενός και η παράμετρος ρ, μεγαλύτερες του 0.5, ώστε να δίνεται ανάλογη βαρύτητα στις πληροφορίες, που δίνουν τα χαρακτηριστικά των κόμβων και παράλληλα, οι θετικά ετικετοποιημένες ακμές να συνεισφέρουν με μεγαλύτερο βάρος στην διαμόρφωση του τελικού αποτελέσματος.

8.1 ARI

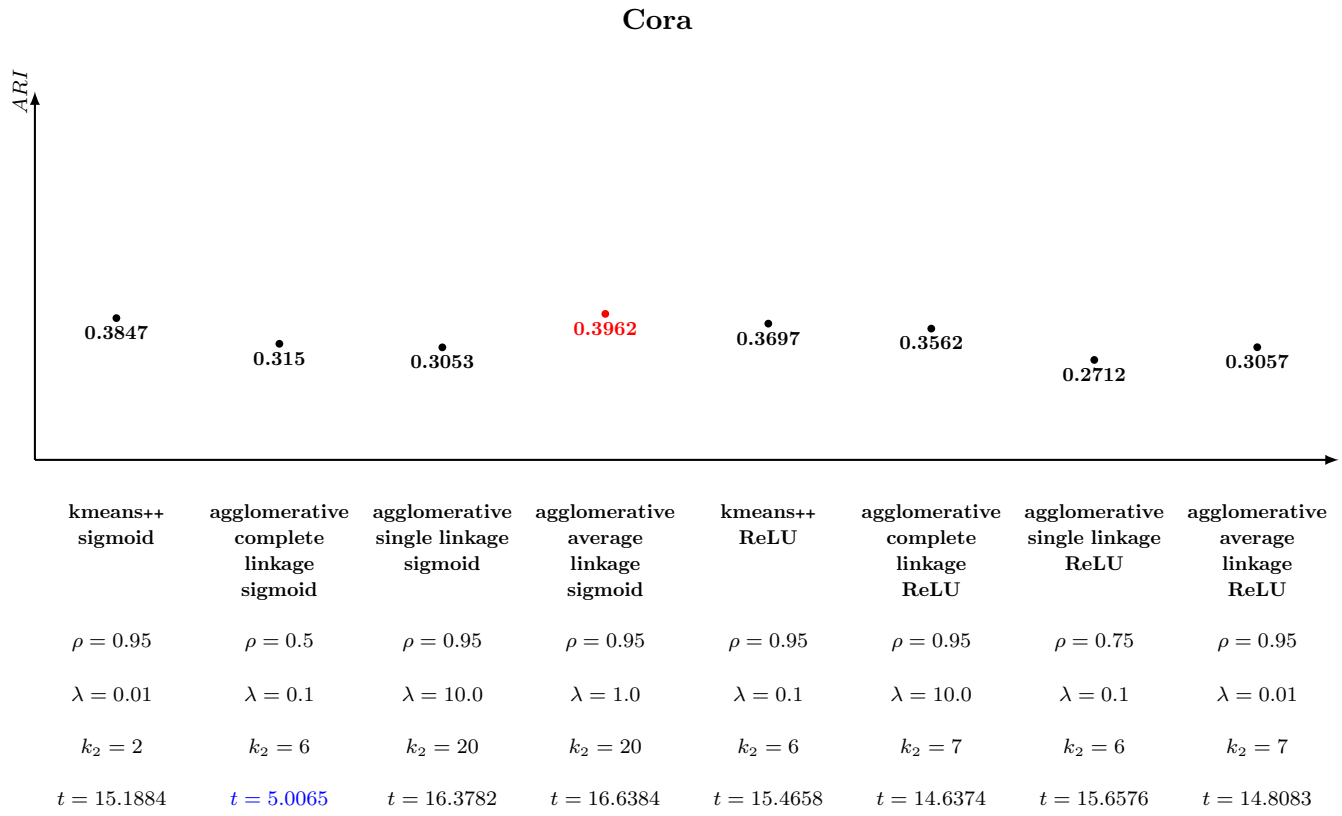
Παρακάτω παρουσιάζονται τα αποτελέσματα των διάφορων παραλλαγών της μεθόδου NAGC και αξιολογούνται με κριτήριο το μέγεθος αξιολόγησης ARI.



kmeans++ sigmoid	agglomerative complete linkage sigmoid	agglomerative single linkage sigmoid	agglomerative average linkage sigmoid	kmeans++ ReLU	agglomerative complete linkage ReLU	agglomerative single linkage ReLU	agglomerative average linkage ReLU
$\rho = 0.995$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.995$	$\rho = 0.5$	$\rho = 0.5$	$\rho = 0.5$
$\lambda = 0.0$	$\lambda = 10.0$	$\lambda = 10.0$	$\lambda = 0.1$	$\lambda = 10.0$	$\lambda = 10.0$	$\lambda = 10.0$	$\lambda = 10.0$
$k_2 = 10$	$k_2 = 15$	$k_2 = 20$	$k_2 = 15$	$k_2 = 15$	$k_2 = 15$	$k_2 = 15$	$k_2 = 15$
$t = 32.6349$	$t = 30.8594$	$t = 30.6724$	$t = 29.9056$	$t = 30.307$	$t = 11.5561$	$t = 11.5447$	$t = 11.3811$

Σχήμα 8.1: Τα αποτελέσματα της μεθόδου για το δείγμα “Citeseer”

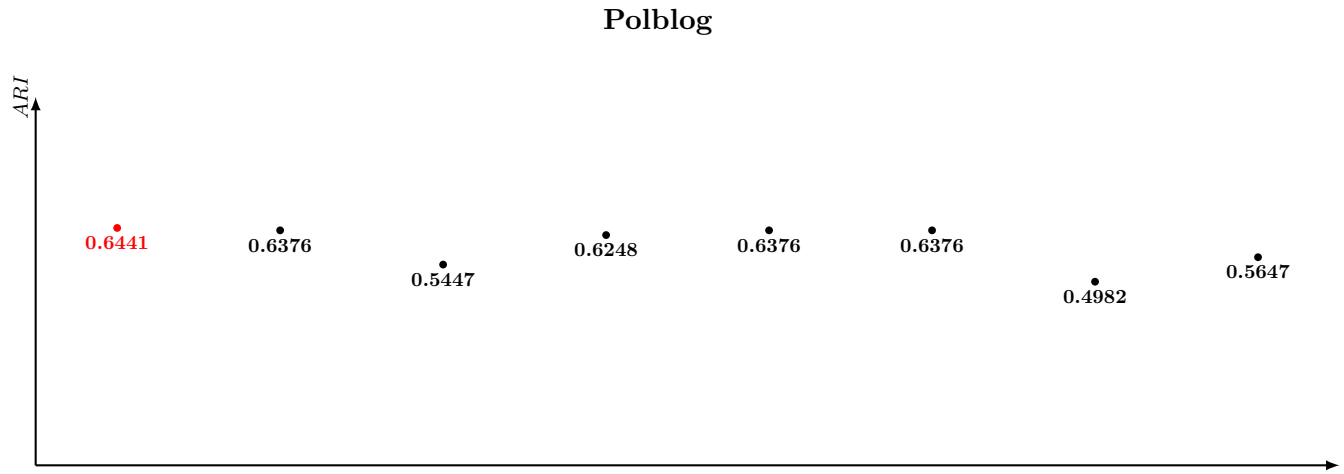
Όπως παρατηρούμε απο το Σχήμα 8.1, στο δείγμα δεδομένων “Citeseer” η μέγιστη τιμή του ARI επιτυγχάνεται για την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την ReLU. Από τις τιμές των παραμέτρων $\rho = 0.995$, $\lambda = 10$ και $k_2 = 15$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγάλη βαρύτητα στις θετικές (υπάρχουσες) ακμές για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), αξιολογούνται σε σημαίνοντα βαθμό τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 15. Συγκρίνοντας με τον συνδυασμό k-means++ - sigmoid ($\rho = 0.995$, $\lambda = 0$ και $k_2 = 10$) της αρχικής δημοσίευσης, παρότι ο τρόπος αντιμετώπισης των θετικών ακμών είναι η ίδιος με την k-means++ - ReLU, η αξιολόγηση των χαρακτηριστικών δεν λαμβάνεται καθόλου υπόψιν στην ανάθεση κόμβων σε κοινότητες. Τέλος, το βέλτιστο χρονικό αποτέλεσμα επιτυγχάνεται με τον συνδυασμό agglomerative με average linkage και συνάρτηση μετατροπής την ReLU.



kmeans++ sigmoid	**agglomerative complete linkage sigmoid**	**agglomerative single linkage sigmoid**	**agglomerative average linkage sigmoid**	**kmeans++ ReLU**	**agglomerative complete linkage ReLU**	**agglomerative single linkage ReLU**	**agglomerative average linkage ReLU**
$\rho = 0.95$	$\rho = 0.5$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.75$	$\rho = 0.95$
$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 10.0$	$\lambda = 1.0$	$\lambda = 0.1$	$\lambda = 10.0$	$\lambda = 0.1$	$\lambda = 0.01$
$k_2 = 2$	$k_2 = 6$	$k_2 = 20$	$k_2 = 20$	$k_2 = 6$	$k_2 = 7$	$k_2 = 6$	$k_2 = 7$
$t = 15.1884$	$t = 5.0065$	$t = 16.3782$	$t = 16.6384$	$t = 15.4658$	$t = 14.6374$	$t = 15.6576$	$t = 14.8083$

Σχήμα 8.2: Τα αποτελέσματα της μεθόδου για το δείγμα "Cora"

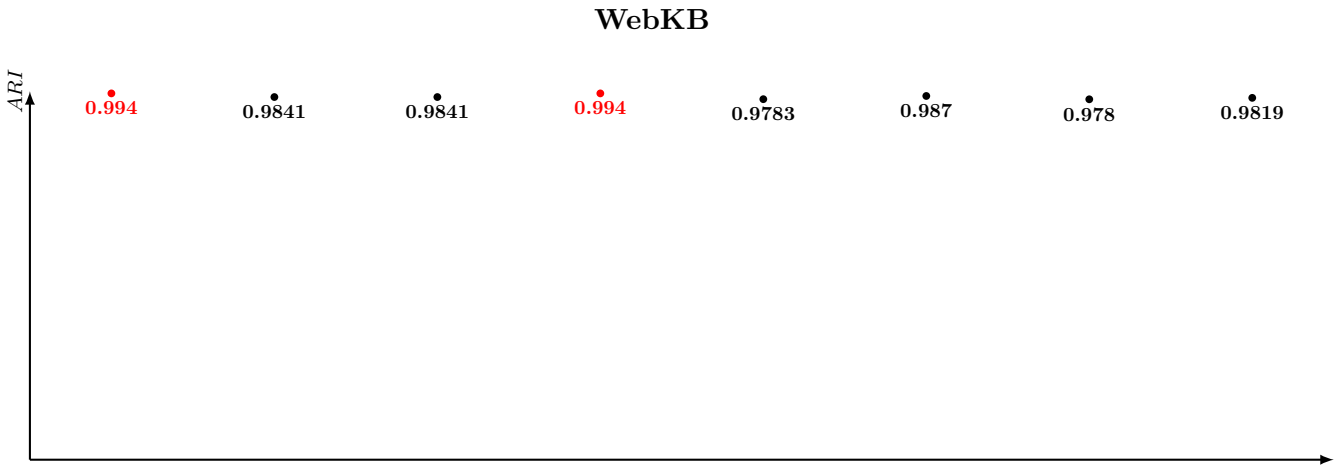
Όπως παρατηρούμε απο το Σχήμα 8.2, στο δείγμα δεδομένων "Cora" η μέγιστη τιμή του ARI επιτυγχάνεται για την αρχικοποίηση των δεδομένων με τον συνδυασμό agglomerative με average linkage και συνάρτηση μετατροπής την sigmoid. Από τις τιμές των παραμέτρων $\rho = 0.95$, $\lambda = 1$ και $k_2 = 20$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγάλη βαρύτητα στις θετικές (υπάρχουσες) ακμές για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), αξιολογούνται μεν, αλλά όχι σημαντικά, τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 20. Συγκρίνοντας με τον συνδυασμό k-means++ - sigmoid ($\rho = 0.95$, $\lambda = 0.01$ και $k_2 = 2$) της αρχικής δημοσίευσης, παρότι ο τρόπος αντιμετώπισης των θετικών ακμών είναι η ίδιος με την agglomerative average linkage - sigmoid, η αξιολόγηση των χαρακτηριστικών λαμβάνεται σε ακόμα μικρότερο βαθμό υπόψιν στην ανάθεση κόμβων σε κοινότητες. Πρέπει να σημειωθεί, πως ο χρόνος εκτέλεσης του αλγορίθμου για τον συνδυασμό agglomerative με complete linkage και συνάρτηση μετατροπής την sigmoid.



kmeans++ sigmoid	**agglomerative complete linkage sigmoid**	**agglomerative single linkage sigmoid**	**agglomerative average linkage sigmoid**	**kmeans++ ReLU**	**agglomerative complete linkage ReLU**	**agglomerative single linkage ReLU**	**agglomerative average linkage ReLU**
$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.995$
$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0001$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.001$	$\lambda = 0.0$
$k_2 = 2$	$k_2 = 10$	$k_2 = 6$	$k_2 = 15$	$k_2 = 15$	$k_2 = 2$	$k_2 = 10$	$k_2 = 5$
$t = 3.9206$	$t = 3.9719$	$t = 3.9215$	$t = 3.6532$	$t = 3.3766$	$t = 3.2724$	$t = 3.1796$	$t = 3.743$

Σχήμα 8.3: Τα αποτελέσματα της μεθόδου για το δείγμα “Polblog”

Όπως παρατηρούμε απο το Σχήμα 8.3, στο δείγμα δεδομένων “Polblog” η μέγιστη τιμή του ARI επιτυγχάνεται για την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την sigmoid. Από τις τιμές των παραμέτρων $\rho = 0.95$, $\lambda = 0$ και $k_2 = 2$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγάλη βαρύτητα στις θετικές (υπάρχουσες) ακμές για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), δεν αξιολογούνται καθόλου τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 2. Πρέπει να σημειωθεί, πως ο χρόνος εκτέλεσης του αλγορίθμου για τον συνδυασμό k-means++ - sigmoid, όμως, δεν είναι ο καλύτερος. Φαίνεται, πως ο βέλτιστος χρόνος εκτέλεσης επιτυγχάνεται για αρχικοποίηση με τον αλγόριθμο agglomerative με average linkage και συνάρτηση μετατροπής την ReLU.



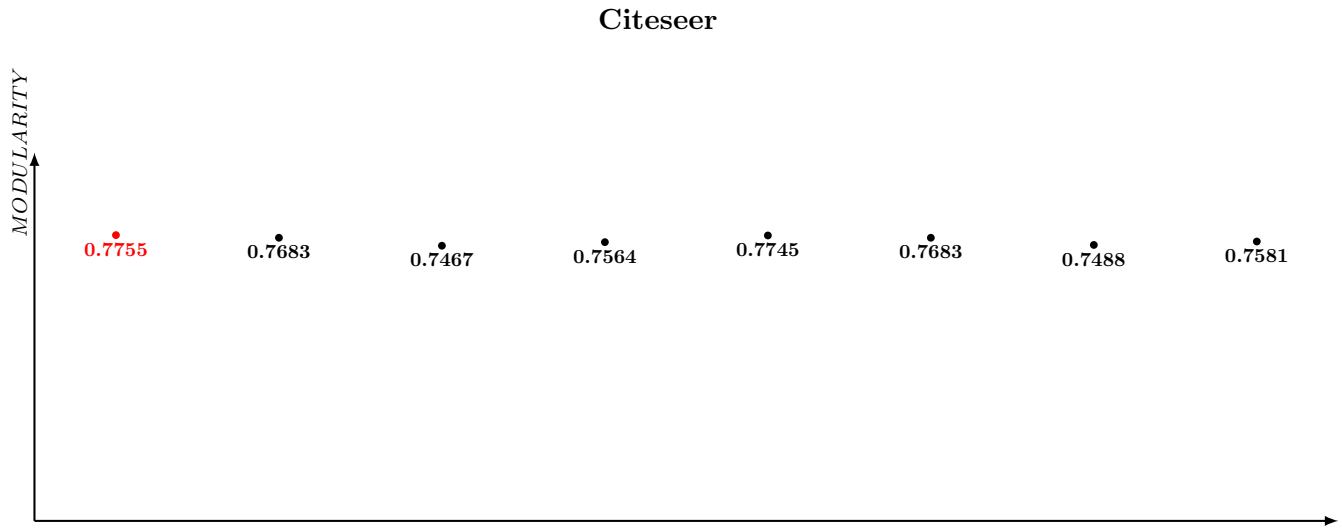
kmeans++ sigmoid	**agglomerative complete linkage sigmoid**	**agglomerative single linkage sigmoid**	**agglomerative average linkage sigmoid**	**kmeans++ ReLU**	**agglomerative complete linkage ReLU**	**agglomerative single linkage ReLU**	**agglomerative average linkage ReLU**
$\rho = 0.5$	$\rho = 0.75$	$\rho = 0.55$	$\rho = 0.75$	$\rho = 0.55$	$\rho = 0.5$	$\rho = 0.5$	$\rho = 0.5$
$\lambda = 0.001$	$\lambda = 1.0$	$\lambda = 0.0001$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 0.1$	$\lambda = 0.001$	$\lambda = 0.0$
$k_2 = 20$	$k_2 = 15$	$k_2 = 20$	$k_2 = 20$	$k_2 = 10$	$k_2 = 10$	$k_2 = 15$	$k_2 = 15$
$t = 1.5813$	$t = 2.4007$	$t = 2.3456$	$t = 2.5673$	$t = 2.1665$	$t = 1.1206$	$t = 1.1862$	$t = 1.189$

Σχήμα 8.4: Τα αποτελέσματα της μεθόδου για το δείγμα "WebKB"

Όπως παρατηρούμε απο το Σχήμα 8.4, στο δείγμα δεδομένων "WebKB" η μέγιστη τιμή του ARI επιτυγχάνεται με δύο συνδυασμούς : την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την sigmoid και την αρχικοποίηση των δεδομένων με την μέθοδο agglomerative με average linkage και συνάρτηση μετατροπής την sigmoid. Για τον συνδυασμό k-means++ - sigmoid, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0.001$ και $k_2 = 20$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αντιμετωπίζονται με κοινό τρόπο οι θετικές (υπάρχουσες) ακμές και οι μη ετικετοποιημένες, για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), αξιολογούνται σε μικρό βαθμό τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 20. Για τον συνδυασμό agglomerative average linkage - sigmoid, από τις τιμές των παραμέτρων $\rho = 0.75$, $\lambda = 0.01$ και $k_2 = 20$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, αποδίδεται μεγαλύτερη βαρύτητα στις θετικές (υπάρχουσες) ακμές, αξιολογούνται σε μικρό βαθμό, αλλά μεγαλύτερο από την k-means++ - sigmoid, τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 20. Πρέπει να σημειωθεί, πως ο χρόνος εκτέλεσης του αλγορίθμου για τον συνδυασμό agglomerative με complete linkage και συνάρτηση μετατροπής την ReLU.

8.2 Modularity

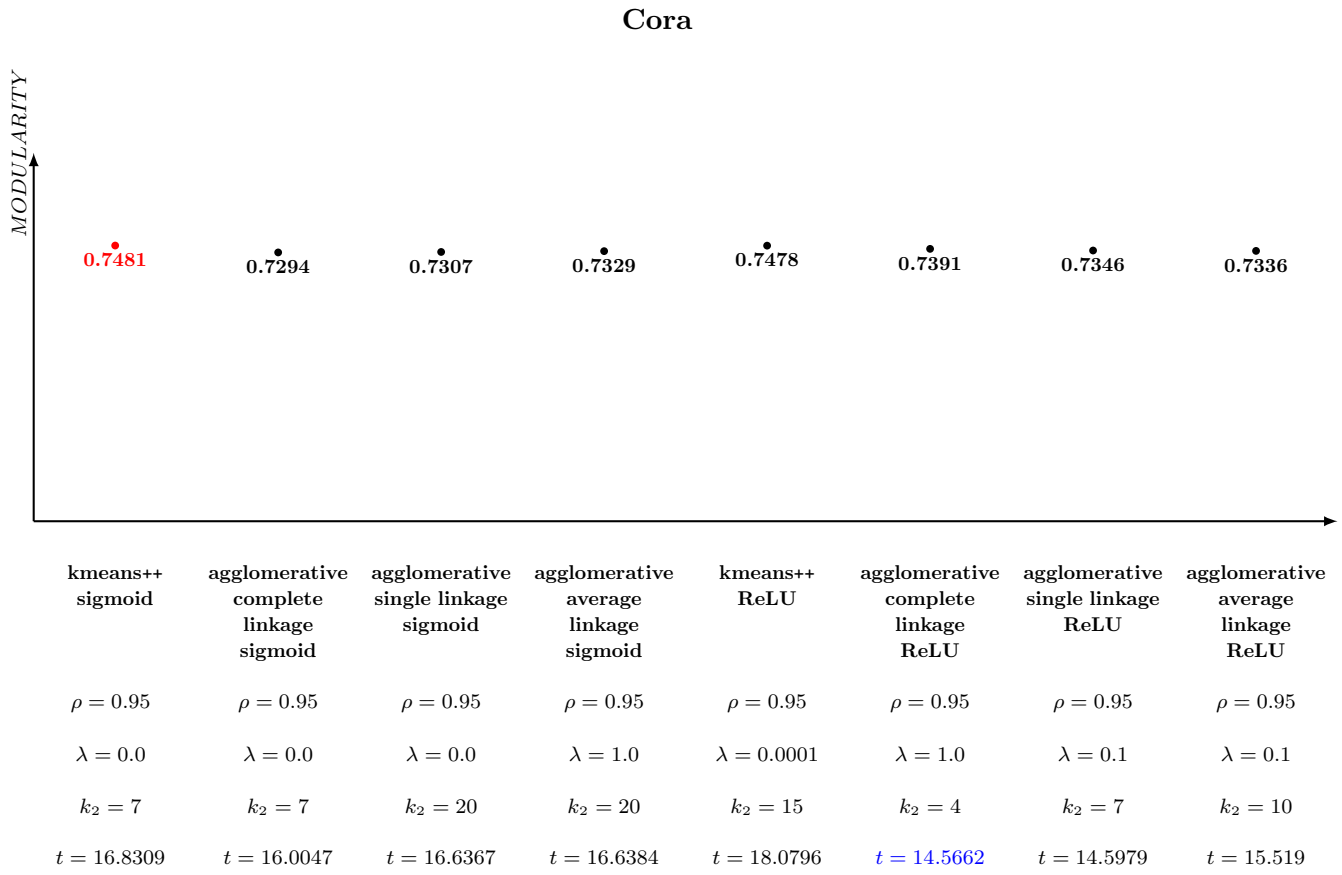
Παρακάτω παρουσιάζονται τα αποτελέσματα των διάφορων παραλλαγών της μεθόδου NAGC και αξιολογούνται με κριτήριο το μέγεθος αξιολόγησης Modularity.



kmeans++ sigmoid	agglomerative complete linkage sigmoid	agglomerative single linkage sigmoid	agglomerative average linkage sigmoid	kmeans++ ReLU	agglomerative complete linkage ReLU	agglomerative single linkage ReLU	agglomerative average linkage ReLU
$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.995$
$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 1.0$	$\lambda = 0.01$
$k_2 = 7$	$k_2 = 2$	$k_2 = 2$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 4$	$k_2 = 10$
$t = 27.3234$	$t = 24.6853$	$t = 24.6875$	$t = 25.8211$	$t = 26.9135$	$t = 26.0323$	$t = 25.2582$	$t = 29.0539$

Σχήμα 8.5: Τα αποτελέσματα της μεθόδου για το δείγμα “Citeseer”

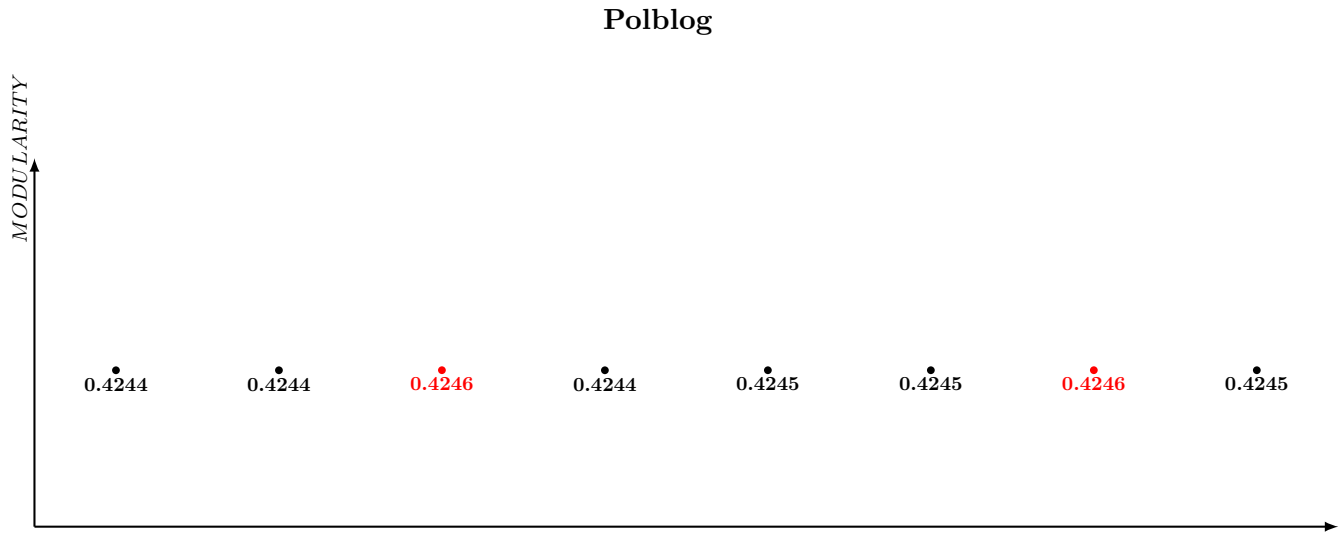
Όπως παρατηρούμε απο το Σχήμα 8.5, στο δείγμα δεδομένων “Citeseer” η μέγιστη τιμή του Modularity επιτυγχάνεται για την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την sigmoid. Από τις τιμές των παραμέτρων $\rho = 0.95$, $\lambda = 0$ και $k_2 = 7$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγάλη βαρύτητα στις θετικές (υπάρχουσες) ακμές για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), δεν αξιολογούνται καθόλου τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 7. Πρέπει να σημειωθεί, πως ο καλύτερος χρόνος εκτέλεσης επιτυγχάνεται από τον συνδυασμό agglomerative complete linkage - sigmoid, που είναι τρίτος συνδυασμός σε σειρά απόδοσης.



kmeans++ sigmoid	**agglomerative complete linkage sigmoid**	**agglomerative single linkage sigmoid**	**agglomerative average linkage sigmoid**	**kmeans++ ReLU**	**agglomerative complete linkage ReLU**	**agglomerative single linkage ReLU**	**agglomerative average linkage ReLU**
$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$	$\rho = 0.95$
$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 1.0$	$\lambda = 0.0001$	$\lambda = 1.0$	$\lambda = 0.1$	$\lambda = 0.1$
$k_2 = 7$	$k_2 = 7$	$k_2 = 20$	$k_2 = 20$	$k_2 = 15$	$k_2 = 4$	$k_2 = 7$	$k_2 = 10$
$t = 16.8309$	$t = 16.0047$	$t = 16.6367$	$t = 16.6384$	$t = 18.0796$	$t = 14.5662$	$t = 14.5979$	$t = 15.519$

Σχήμα 8.6: Τα αποτελέσματα της μεθόδου για το δείγμα “Cora”

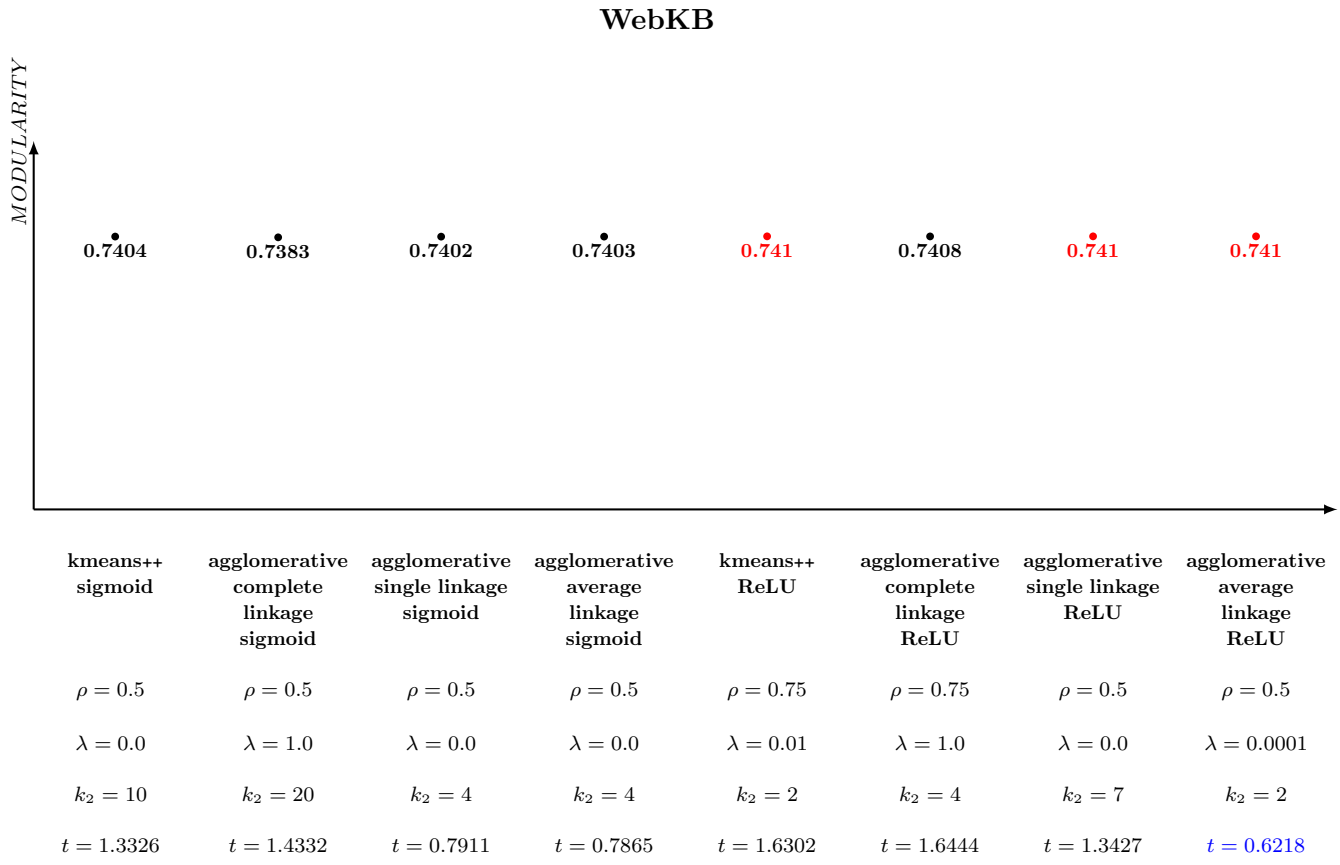
Όπως παρατηρούμε απο το Σχήμα 8.6, στο δείγμα δεδομένων “Cora” η μέγιστη τιμή του Modularity επιτυγχάνεται για την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την sigmoid. Από τις τιμές των παραμέτρων $\rho = 0.95$, $\lambda = 0$ και $k_2 = 7$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγάλη βαρύτητα στις θετικές (υπάρχουσες) ακμές για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), δεν αξιολογούνται καθόλου τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 7. Πρέπει να σημειωθεί, πως ο καλύτερος χρόνος εκτέλεσης επιτυγχάνεται από τον συνδυασμό agglomerative complete linkage - ReLU, που είναι τρίτος συνδυασμός σε σειρά απόδοσης.



kmeans++ sigmoid	agglomerative complete linkage sigmoid	agglomerative single linkage sigmoid	agglomerative average linkage sigmoid	kmeans++ ReLU	agglomerative complete linkage ReLU	agglomerative single linkage ReLU	agglomerative average linkage ReLU
$\rho = 0.75$	$\rho = 0.55$	$\rho = 0.5$	$\rho = 0.55$	$\rho = 0.75$	$\rho = 0.75$	$\rho = 0.5$	$\rho = 0.75$
$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.0001$	$\lambda = 0.0$
$k_2 = 6$	$k_2 = 5$	$k_2 = 15$	$k_2 = 5$	$k_2 = 2$	$k_2 = 4$	$k_2 = 7$	$k_2 = 10$
$t = 4.0871$	$t = 3.8935$	$t = 0.7896$	$t = 3.3483$	$t = 4.0042$	$t = 3.1818$	$t = 0.3756$	$t = 3.9104$

Σχήμα 8.7: Τα αποτελέσματα της μεθόδου για το δείγμα “Polblog”

Όπως παρατηρούμε απο το Σχήμα 8.7, στο δείγμα δεδομένων “Polblog” η μέγιστη τιμή του Modularity επιτυγχάνεται με δύο συνδυασμούς : την αρχικοποίηση των δεδομένων με την μέθοδο agglomerative με single linkage και συνάρτηση μετατροπής την sigmoid και την αρχικοποίηση των δεδομένων με την μέθοδο agglomerative με single linkage και συνάρτηση μετατροπής την ReLU, όπου, παράλληλα, επιτυγχάνεται και ο βέλτιστος χρόνος εκτέλεσης του αλγορίθμου. Για τον συνδυασμό agglomerative single linkage - sigmoid, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0$ και $k_2 = 15$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται ίση βαρύτητα στις θετικές (υπάρχουσες) ακμές και τις μη ετικετοποιημένες, για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), αξιολογούνται σε πολύ μικρό βαθμό τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 15. Για τον συνδυασμό agglomerative single linkage - ReLU, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0.0001$ και $k_2 = 7$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται μεγαλύτερη βαρύτητα στις θετικές(υπάρχουσες) ακμές, δεν αξιολογούνται καθόλου τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες βάσει χαρακτηριστικών, να είναι ιδανικά ίσες με 7.



kmeans++ sigmoid	agglomerative complete linkage sigmoid	agglomerative single linkage sigmoid	agglomerative average linkage sigmoid	kmeans++ ReLU	agglomerative complete linkage ReLU	agglomerative single linkage ReLU	agglomerative average linkage ReLU
$\rho = 0.5$	$\rho = 0.5$	$\rho = 0.5$	$\rho = 0.5$	$\rho = 0.75$	$\rho = 0.75$	$\rho = 0.5$	$\rho = 0.5$
$\lambda = 0.0$	$\lambda = 1.0$	$\lambda = 0.0$	$\lambda = 0.0$	$\lambda = 0.01$	$\lambda = 1.0$	$\lambda = 0.0$	$\lambda = 0.0001$
$k_2 = 10$	$k_2 = 20$	$k_2 = 4$	$k_2 = 4$	$k_2 = 2$	$k_2 = 4$	$k_2 = 7$	$k_2 = 2$
$t = 1.3326$	$t = 1.4332$	$t = 0.7911$	$t = 0.7865$	$t = 1.6302$	$t = 1.6444$	$t = 1.3427$	$t = 0.6218$

Σχήμα 8.8: Τα αποτελέσματα της μεθόδου για το δείγμα “WebKB”

Όπως παρατηρούμε απο το Σχήμα 8.8, στο δείγμα δεδομένων “WebKB” η μέγιστη τιμή του Modularity επιτυγχάνεται με τρεις συνδυασμούς : την αρχικοποίηση των δεδομένων με τον συνδυασμό agglomerative με single linkage και συνάρτηση μετατροπής την ReLU, την αρχικοποίηση των δεδομένων με τον συνδυασμό agglomerative με average linkage και συνάρτηση μετατροπής την ReLU, όπου, παράλληλα, επιτυγχάνεται και ο βέλτιστος χρόνος εκτέλεσης του αλγορίθμου και την αρχικοποίηση των δεδομένων με την μέθοδο k-means++ και συνάρτηση μετατροπής την ReLU. Για τον συνδυασμό agglomerative single linkage - ReLU, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0.01$ και $k_2 = 7$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, σε αυτή την περίπτωση, αποδίδεται ίση βαρύτητα στις θετικές (υπάρχουσες) ακμές και τις μη ετικετοποιημένες, για τον υπολογισμό του πίνακα U (μέσω της παραγοντοποίησης του πίνακα γειτνίασης S), αξιολογούνται σε μικρό βαθμό τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες των χαρακτηριστικών, να είναι ιδανικά ίσες με 7. Για τον συνδυασμό agglomerative average linkage - ReLU, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0.0001$ και $k_2 = 7$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, ίση βαρύτητα στις θετικές (υπάρχουσες) ακμές και τις μη ετικετοποιημένες, αξιολογούνται σε ακόμα μικρότερο βαθμό τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες των χαρακτηριστικών, να είναι ιδανικά ίσες με 7. Για τον συνδυασμό k-means++ - ReLU, από τις τιμές των παραμέτρων $\rho = 0.5$, $\lambda = 0$ και $k_2 = 4$ φαίνεται, πως για την βέλτιστη δυνατή ανίχνευση των κοινοτήτων, αποδίδεται ίση βαρύτητα στις θετικές (υπάρχουσες) ακμές και τις μη ετικετοποιημένες, δεν αξιολογούνται καθόλου τα χαρακτηριστικά των κόμβων και εκτιμώνται οι κοινότητες των χαρακτηριστικών, να είναι ιδανικά ίσες με 2.

8.3 Συμπεράσματα

Με την εξαγωγή των πειραμάτων φαίνεται, πως, πράγματι, η αρχικοποίηση των δεδομένων διαδραματίζει σημαίνοντα ρόλο στην απόδοση του αλγορίθμου χρονικά και ποιοτικά. Συμπληρωματικά, επιβεβαιώνεται πως η διατήρηση των μεγεθών σε πιο πραγματική κλίμακα, με την χρήση της συνάρτησης ReLU, έναντι της sigmoid, εξυπηρετεί στην βελτιστοποίηση της τελικής πρόβλεψης. Κατά τις συγκρίσεις των διαφορετικών συνδυασμών αλγορίθμων, τόσο με βάση την μετρική ARI, αλλά και με βάση την μετρική Modularity, σε πολλές περιπτώσεις με τον agglomerative αλγόριθμο και τα εκάστοτε linkages, αλλά και με τη χρήση της συνάρτησης ReLU, φαίνεται να επιτυγχάνονται καλύτερες προβλέψεις και μικρότεροι χρόνοι εκτέλεσης, σε σχέση με το πως προκύπτουν με την αρχική διαδικασία της NAGC. Σχετικά με τις παραμέτρους, σε όλες τις περιπτώσεις η τιμή του ρ διατηρείται σε τιμές πολύ κοντινές στο 1, το οποίο θεωρείται λογικό και επιθυμητό, καθώς όπως αναλύθηκε, υψηλές τιμές του ρ υποδεικνύουν μεγαλύτερη βαρύτητα στην αξιοποίηση των labeled, υπάρχοντων ακμών του γραφήματος για την τελική ανάθεση κόμβων σε κοινότητες. Η παράμετρος λ προτιμάται να λαμβάνει τιμές διαφορετικές του 0. Όπως αναφέρθηκε στην δημοσίευση των Maekawa, Takeuchi και Onizuka [3] η επιρροή των χαρακτηριστικών στο τελικό αποτέλεσμα είναι αξιόλογη ακόμα και για τιμές κοντά στο 0, ενώ για $\lambda = 0$ η απόδοση του ARI δείχνει να μειώνεται αρκετά. Οι τιμές του λ, που επιλέγονται για τις μέγιστες τιμές του ARI και πάλι δεν είναι αρκετά μεγάλες και αυτό αποδίδεται στην δομή των χαρακτηριστικών με την μέθοδο bag of words χωρίς συχνότητες. Σε μελέτη που πραγματοποιήθηκε για τους σκοπούς της ίδιας δημοσίευσης, αποδεικνύεται ότι η επιλογή του k_2 μπορεί να αφεθεί ελεύθερη σε ένα εύρος τιμών χωρίς να επηρεάζει την τελική ανάθεση κόμβων σε κοινότητες, ζήτημα που μελετήθηκε με βάση την μετρική ARI. Γενικότερα, οι συγγραφείς αναφέρουν πως οι non adjusted μετρικές δεν αποδίδουν καλά σε περιπτώσεις γράφων με χαρακτηριστικά. Πράγματι, προς αυτή την κατεύθυνση, πραγματοποιήθηκαν πειράματα τα οποία αξιολογήθηκαν με την μετρική Modularity. Γίνεται σαφές από την επιλογή των παραμέτρων για τις μέγιστες τιμές την μετρικής, πως η συμμετοχή των χαρακτηριστικών λαμβάνεται λίγο και πολλές φορές καθόλου υπόψιν στην ομαδοποίηση και μάλιστα, με αυτόν τον τρόπο, δεν φαίνεται να μπορούν να επιτευχθούν μεγάλα ποσοστά αξιοπιστίας σχετικά με την πρόβλεψη σωστών κοινοτήτων.

8.4 Μελλοντικές Κατευθύνσεις

Σαν μελλοντική κατεύθυνση, θα ήταν δόκιμο, να δομηθούν τα χαρακτηριστικά των κόμβων με εναλλακτικές, σύγχρονες μεθόδους, με σκοπό, την κατά τον δυνατόν πιο αποδοτική αξιοποιήση τους. Παράλληλα, με αυτόν τον τρόπο, θα ήταν εφικτό κατά την αρχικοποίηση των δεδομένων, να προβλεφθεί ο αριθμός των τοπολογικών ομάδων (k_1), χωρίς να χρειάζεται να δοθεί ως είσοδος στον αλγόριθμο. Τέλος, θα ήταν χρήσιμο να πραγματοποιηθούν επιπλέον πειράματα, ώστε να είναι δυνατό να εξαχθούν στατιστικές μελέτες σχετικά με την σημαντικότητα των διαφορών των τιμών, που λαμβάνουν οι μετρικές αξιολόγησης, με σκοπό την εξαγωγή ακριβέστερων συμπερασμάτων.

8.5 Κώδικας

Ο πρωτότυπος κώδικας, που αναπαράχθηκε για την αναπαράσταση και υλοποίηση της μεθόδου NAGC, γράφτηκε από τους συγγραφείς της δημοσίευσης Maekawa, Takeuchi και Onizuka [3] και μπορεί να εντοπιστεί στην διεύθυνση : https://github.com/seijimaekawa/NAGC/. Για τους σκοπούς της παρούσας εργασίας, πραγματοποιήθηκαν μετατροπές στον πρωτότυπο κώδικα με στόχο την διαφοροποίηση στην αρχικοποίηση των δεδομένων (k-means++, agglomerative, spectral, DBSCAN), που δέχεται ο αλγόριθμος και την εναλλαγή μεταξύ συναρτήσεων μετατροπής (sigmoid και ReLU). Ο κώδικας είναι γραμμένος στην γλώσσα Python και παρατίθεται αυτούσιος στο Παράρτημα Α′, στο τέλος της εργασίας. Στην ενότητα αυτή, πραγματοποιείται μια σύντομη αναφορά στην λειτουργία των βασικών κλάσεων και τα σημεία στα οποία έγιναν οι αλλαγές.

- **build_graph :** Διαβάζει τα δεδομένα από το γράφημα, σε συγκεκριμένες μορφές `.cite` και `.content` και δημιουργεί τους πίνακες S (πίνακας γειτνίασης) και X (πίνακας κόμβων-χαρακτηριστικών). Παράλληλα, αποθηκεύει σε κατάλληλες παραμέτρους την groundtruth ανάθεση των κόμβων σε ομάδες.
- **init_kmeans :** Ορίζει την μέθοδο k-means++.
- **init_agglomerative :** Ορίζει την μέθοδο agglomerative.
- **init_spectral :** Ορίζει την μέθοδο spectral.
- **NAGC_class :** Υλοποιεί την διαδικασία της βελτιστοποίησης/μάθησης του αλγορίθμου με βάση τους σχετικούς κανόνες ανανέωσης. Επίσης, εκεί ορίζεται η συνάρτηση μετατροπής sigmoid.
- **NAGC_ReLU :** Υλοποιεί την διαδικασία της βελτιστοποίησης/μάθησης του αλγορίθμου με βάση τους σχετικούς κανόνες ανανέωσης. Επίσης, εκεί ορίζεται η συνάρτηση μετατροπής ReLU.
- **VU_init :** Αρχικοποιεί τους πίνακες U,V με βάση την αντίστοιχη μέθοδο αρχικοποιήσεις όπως ορίζεται στις κλάσεις `init_kmeans`, `init_agglomerative` και τυπικά τις `init_spectral` και `init_dbscan`.
- **evaluate :** Ορίζει τα μεγέθη αξιολόγησης ARI, entropy, modularity.
- **main :** Με αυτή την κλάση εκτελείται ο αλγόριθμος και παρουσιάζονται τα αποτελέσματα των μετρικών.

Οι κλάσεις `init_agglomerative`, `init_spectral` και `NAGC_ReLU` δημιουργήθηκαν από την αρχή, με βάση τις `init_kmeans` και `NAGC_class` με σκοπό να εξυπηρετηθούν οι στόχοι της εργασίας. Στον κώδικα, επίσης, προστέθηκαν επεξηγηματικά σχόλια για την καλύτερη κατανοήση του. Για την εκτέλεση του αλγορίθμου, έχει υλοποιηθεί σχετικό αρχείο "`requirements.txt`" στο οποίο εμπεριέχονται όλα τα απαραίτητα πακέτα/βιβλιοθήκες της Python, που είναι απαραίτητα.

Βιβλιογραφία

[1] Santo Fortunato. «Community Detection in Graphs». Στο: *Physics Reports* 486 (3–5 Φεβ. 2010), σσ. 75–174. doi: 10.1016/j.physrep.2009.11.002.

[2] Saurav Kaushik. *An Introduction to Clustering and Different Methods of Clustering*. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/. Online; Accessed: 03 Sep 2022. Νοέ. 2016.

[3] Seiji Maekawa, Koh Takeuchi και Makoto Onizuka. «New Attributed Graph Clustering by Bridging Attribute and Topology Spaces». Στο: *Journal of Information Processing* 28 (Αύγ. 2020), σσ. 427–435. doi: 10.2197/ipsjjip.28.427.

[4] Aris Pagourtzis κ.ά. «Overlapping Community Detection via Minimum Spanning Tree Computations». Στο: *6th IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2020, Oxford, UK, August 3-6, 2020*. IEEE, 2020, σσ. 62–65. doi: 10.1109/BigDataService49289.2020.00017. url: https://doi.org/10.1109/BigDataService49289.2020.00017.

[5] Ketki Kulkarni κ.ά. «Community Detection via Neighborhood Overlap and Spanning Tree Computations». Στο: *Algorithmic Aspects of Cloud Computing - 4th International Symposium, ALGOCLOUD 2018, Helsinki, Finland, August 20-21, 2018, Revised Selected Papers*. Επιμέλεια υπό Yann Disser και Vassilios S. Verykios. Τόμ. 11409. Lecture Notes in Computer Science. Springer, 2018, σσ. 13–24. doi: 10.1007/978-3-030-19759-9_2. url: https://doi.org/10.1007/978-3-030-19759-9%5C_2.

[6] Dora Souliou κ.ά. «Weight assignment on edges towards improved community detection». Στο: *Proceedings of the 23rd International Database Applications & Engineering Symposium, IDEAS 2019, Athens, Greece, June 10-12, 2019*. Επιμέλεια υπό Bipin C. Desai κ.ά. ACM, 2019, 3:1–3:5. doi: 10.1145/3331076.3331121. url: https://doi.org/10.1145/3331076.3331121.

[7] Petros Potikas κ.ά. «A parallel community detection algorithm based on spanning trees». Στο: *The 8th IEEE International Conference on Big Data Computing Service and Machine Learning Applications*. IEEE BigDataService 2022. 2022.

[8] Καφεσάκη Μαρία. *Τελεστές και Πίνακες*. http://esperia.iesl.forth.gr/~kafesaki/Applied-Mathematics/linear-algebra/tensors-matrices-part-print.htm. Online; Accessed: 03 Sep 2022. 2021.

[9] Eric W. Weisstein. *Conjugate Transpose*. https://mathworld.wolfram.com/ConjugateTranspose.html. Online; Accessed: 03 Sep 2022. 2003.

[10] Kaare Brandt Petersen και Michael Syskind Pedersen. *The Matrix Cookbook*. http://www.cse.cuhk.edu.hk/~cslui/CSCI3320_LECTURES/The_Matrix_Cookbook.pdf. Online; Accessed: 03 Sep 2022. Φεβ. 2006.

[11] Chien-Hao Huang, Jein-Shan Chen και Chu-Chin Hu. «The Schatten p-norm on R^n». Στο: *Journal of Nonlinear and Convex Analysis* (Νοέ. 2020), σσ. 21–29. url: https://math.ntnu.edu.tw/~jschen/Papers/schatten-p-norm-JNCA-2020.pdf.

[12] Er Raqabi El Mehdi. «Non-convex optimization in deep learning». Στο: *The Startup* (Ιούλ. 2020). url: https://medium.com/swlh/non-convex-optimization-in-deep-learning-26fa30a2b2b3.

[13] Aston Zhang κ.ά. «Convexity». Στο: *Dive into deep learning*. d2l.ai, Μάι. 2019. url: https://d2l.ai/chapter_optimization/convexity.html.

[14] *Optimization Problem Types — Convex Optimization*. https://www.solver.com/convex-optimization. Online; Accessed: 03 Sep 2022. Αύγ. 2012.

[15] Martin J. Osborne. *Mathematical methods for economic theory*. https://mjo.osborne.economics.utoronto.ca/index.php/tutorial/index/1/cv1/. Online; Accessed 03 Sep 2022. 2022.

[16] Jason Brownlee. «A Gentle Introduction to the Rectified Linear Unit (ReLU)». Στο: *Deep Learning Performance* (Αύγ. 2020). url: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[17] Wikiwand.com. *Σιμοειδής Συνάρτηση*. Online; Accessed: 03 Sep 2022. 2013. url: https://www.wikiwand.com/el/%CE%A3%CE%B9%CE%B3%CE%BC%CE%BF%CE%B5%CE%B9%CE%B4%CE%AE%CF%82_%CF%83%CF%85%CE%BD%CE%AC%CF%81%CF%84%CE%B7%CF%83%CE%B7.

[18] Deepak Jain. *Data preprocessing in Data Mining*. https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/. Online; Accessed: 03 Sep 2022. Ιούν. 2021.

[19] Raghav Vashisht. «Machine learning: When to perform a feature scaling?» Στο: *Explore further with atoti* (Ιαν. 2021). url: https://www.atoti.io/articles/when-to-perform-a-feature-scaling/.

[20] Azika Amelia. «K-Means Clustering: From A to Z». Στο: *Towards Data Science* (Σεπτ. 2018). url: https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a.

[21] Stuart P. Lloyd. «Least squares quantization in PCM». Στο: *IEEE Transactions on Information Theory* 28.2 (Μαρ. 1982), σσ. 129–137. doi: 10.1109/TIT.1982.1056489.

[22] David Arthur και Sergei Vassilvitskii. «K-Means++: The Advantages of Careful Seeding». Στο: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial και Applied Mathematics, 2007, σσ. 1027–35. isbn: 9780898716245.

[23] Pulkit Sharma. «Hierarchical clustering: Hierarchical clustering python». Στο: *Analytics Vidhya* (Μάι. 2019). url: https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/.

[24] Marina Chatterjee. *Introduction to Spectral Clustering*. https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/. Online; Accessed 03 Sep 2022. Αύγ. 2020.

[25] Eric Bunch. *Spectral Clustering*. https://eric-bunch.github.io/blog/spectral-clustering. Online; Accessed 03 Sep 2022. Ιούν. 2018.

[26] William Fleshman. «Spectral clustering». Στο: *Towards Data Science* (Φεβ. 2019). url: https://towardsdatascience.com/spectral-clustering-aba2640c0d5b.

[27] deepai.org. *Affinity matrix*. Ιούν. 2020. url: https://deepai.org/machine-learning-glossary-and-terms/affinity-matrix.

[28] Ulrike von Luxburg. «A Tutorial on Spectral Clustering». Στο: *Statistics and Computing* 17.4 (Αύγ. 2007), σσ. 395–416. doi: 10.1007/s11222-007-9033-z.

[29] Sam Gutekunst. *ORIE 6334 Spectral Graph Theory*. https://people.orie.cornell.edu/dpw/orie6334/Fall2016/lecture7.pdf/. Online; Accessed 03 Sep 2022. 2016.

[30] Soner Yıldırım. «DBSCAN clustering - explained». Στο: *Towards Data Science* (Απρ. 2020). url: https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556.

[31] Daniel Lee και Hyunjune Sebastian Seung. «Algorithms for Non-negative Matrix Factorization». Στο: *Advances in Neural Information Processing Systems*. Επιμέλεια υπό T. Leen, T. Dietterich και V. Tresp. Τόμ. 13. MIT Press, 2001. isbn: 9781713845393. url: https://proceedings.neurips.cc/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.

[32] Yaroslav Shitov. «A short proof that NMF is NP-hard». Στο: *ArXiv* abs/1605.04000 (Μάι. 2016). arXiv: 1605.04000 [math.CO].

[33] Jingu Kim και Haesun Park. «Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons». Στο: *2008 Eighth IEEE International Conference on Data Mining*. IEEE, Φεβ. 2009, σσ. 353–362. doi: 10.1109/ICDM.2008.149.

[34] Jason Brownlee. *A Gentle Introduction to Expectation-Maximization (EM Algorithm)*. https://machinelearningmastery.com/expectation-maximization-em-algorithm/. Online; Accessed 03 Sep 2022. 2019.

[35] Da Kuang, Chris Ding και Haesun Park. «Symmetric Nonnegative Matrix Factorization for Graph Clustering». Στο: *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*. 2012, σσ. 106–117. doi: 10.1137/1.9781611972825.10.

[36] Chris Ding κ.ά. «Orthogonal Nonnegative Matrix T-Factorizations for Clustering». Στο: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: Association for Computing Machinery, 2006, σσ. 126–135. isbn: 1595933395. doi: 10.1145/1150402.1150420.

[37] Charles Elkan και Keith Noto. «Learning Classifiers from Only Positive and Unlabeled Data». Στο: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, σσ. 213–220. isbn: 9781605581934. doi: 10.1145/1401890.1401920.

[38] Cho-Jui Hsieh, Nagarajan Natarajan και Inderjit S. Dhillon. «PU Learning for Matrix Completion». Στο: abs/1411.6081 (Νοέ. 2014). arXiv: 1411.6081 [cs.LG].

[39] Clayton D. Scott. «Calibrated asymmetric surrogate losses». Στο: *Electronic Journal of Statistics* 6 (2012), σσ. 958–992. doi: 10.1214/12-EJS699.

[40] Καρούζου Μαριάννα. «Επίλυση Προβλημάτων Βελτιστοποίησης δυο Στόχων με Βάση τη θεωρεία της Συνέχισης του Μετώπου PARETO». Bachelor's Thesis. Σεπτ. 2015. url: http://velos0.ltt.mech.ntua.gr/kgianna/diplomat/fp/karouzou.pdf.

[41] apmonitor.com. *Design optimization*. http://apmonitor.com/me575/index.php/Main/KuhnTucker. Online; Accessed: 03 Sep 2022. Ιούν. 2020.

[42] Jason Brownlee. *A Gentle Introduction to the Bag-of-Words Model*. https://machinelearningmastery.com/gentle-introduction-bag-words-model/. Online; Accessed 03 Sep 2022. Αύγ. 2019.

[43] Mark Craven κ.ά. «Learning to Extract Symbolic Knowledge from the World Wide Web». Στο: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. AAAI '98/IAAI '98. Madison, Wisconsin, USA: American Association for Artificial Intelligence, 1998, σσ. 509–516. isbn: 0262510987.

[44] Katastros. *Introduction to the cora data set of the graph data set-processed by pyton-can be used for GCN tasks*. https://blog.katastros.com/a?ID=01150-ffc8fdc3-5cfc-4040-b9ea-6d888a1db663. Online; Accessed 03 Sep 2022. 2020.

[45] Notebook Community. *Citeseer citation network*. https://notebook.community/gear/motifwalk/notebooks/Citeseer%20network%20analysis. Online; Accessed 03 Sep 2022. 2020.

[46] Raiyan Abdul Baten. *Partitioning a network of US political blogs*. https://github.com/raiyan1102006/Partitioning_US_Political_Blogs. Online; Accessed 03 Sep 2022. 2018.

[47] Jason Brownlee. *A Gentle Introduction to Cross-Entropy for Machine Learning*. https://machinelearningmastery.com/cross-entropy-for-machine-learning/. Online; Accessed 03 Sep 2022. 2019.

[48] Αθανάσιος Π. Λιάβας. *Στοιχεία Θεωρίας Πληροφορίας*. https://eclass.aueb.gr/modules/document/file.php/INF242/%CE%A3%CE%B7%CE%BC%CE%B5%CE%B9%CF%8E%CF%83%CE%B5%CE%B9%CF%82%20%CE%9B%CE%B9%CE%AC%CE%B2%CE%B1%20%28update%29.pdf. Online; Accessed 13 Sep 2022. 2012.

[49] Xin Liu, Hui-Min Cheng και Zhong-Yuan Zhang. «Evaluation of Community Detection Methods». Στο: abs/1807.01130 (Ιούλ. 2018). arXiv: 1807.01130 [qcs.SI].

[50] James Bailey. *Statistically Correcting for Chance using the Adjusted and Standardized Mutual Information Measures*. https://cbrc3.cbrc.jp/twsdm2015/pdf/bailey.pdf. Φεβ. 2015.

[51] Yun Yang. «Chapter 3 - Temporal Data Clustering». Στο: *Temporal Data Mining Via Unsupervised Ensemble Learning*. Elsevier, 2017, σσ. 19–34. isbn: 978-0-12-811654-8. doi: 10.1016/B978-0-12-811654-8.00003-8.

Παράρτημα Α΄

Κώδικας (Python)

Α΄.1 build_graph

```
# This program build adjacency matrix and attribute matrix from graph data.
# This program can deal with two format(.mat and .cite&.content)
# Outputs are below
'''
S : adjacency matrix with prepocess
S_ori : original adjacency matrix
A : attribute matrix with preprocess
clus : true cluster of nodes
flag : with ground truth or without
A_ori : original attribute matrix
'''

import numpy as np
import glob
import scipy.io

def build_graph(path): # switch for two form of file
    if "mat" in path:
        print ("mat")
        S,S_ori,A,clus,A_ori = for_mat(path)
        flag=1
        if clus==[]:
            flag=0
        return S,S_ori,A,clus,flag,A_ori
    else:
        print ("cite")
        S,S_ori,A,clus,A_ori = for_cites_contents(path)
        return S,S_ori,A,clus,1,A_ori


def for_mat(path):
    mat_contents = scipy.io.loadmat(path)
```

```
    G = mat_contents["Network"]
    X = mat_contents["Attributes"]
    Label = list(map(int,mat_contents["Label"]))
    node_size = G.shape[0]
    att_size = X.shape[1]
    # S = lil_matrix((node_size,node_size))
    # S = np.zeros((node_size,node_size))
    # A = np.zeros((node_size,att_size))
    S = np.zeros((node_size,node_size))
    A = X.toarray()
    # fill the adjacency matrix and attribute matrix
    nonzeros = G.nonzero()
    print ("no.nodes: " + str(node_size))
    print ("no.attributes: " + str(att_size))
    edgecount=0
    for i in range(len(nonzeros[0])):
        S[nonzeros[0][i],nonzeros[1][i]] = 1
        S[nonzeros[1][i],nonzeros[0][i]] = 1
    # erase diagonal elements
    diag = 0
    for i in range(node_size):
        diag += S[i,i]
    nonzeros = S.nonzero()
    edge_count = int((len(nonzeros[0])+diag)/2)
    print ("number of edges : " + str(edge_count))

    S_pre,A_pre=preprocess(S,A)
    return S_pre,S,A_pre,Label,A #scaling S and A


def for_cites_contents(path):
    node={}# key value pairs e.g vaso: 1
    counter=0
    att_list=[]
    clus=[]
########## download atttributes ###############
    infiles = glob.glob(path+'/*.content')
    for infile in infiles:
        with open(infile) as f:
            while True:
```

```
                line = f.readline()
                if line == '':
                    break
                tmp = line.split("\t")
                # assign an index to each node
                # in the nodes dictionary go to the
                # node entry and assign the number of the counter (index)
                node[tmp[0]]=counter
                counter+=1
                # get the remaining elements of each line and append to the
                # attributes list (which becomes a list of lists 2-d array)
                att_list.append((tmp[1:-1]))
                # get the last element of each line and add to the end
                # of the list (array like)
                clus.append(tmp[-1].replace("\n",""))
    print ("number of nodes : " + str(len(node)))
    # print the number of attributes of the first entry
    print ("number of attributes : " + str(len(att_list[0])))

########## download edges ###############
    edges=[]
    infiles = glob.glob(path+'/*.cites')
    for infile in infiles:
        with open(infile) as f:
            while True:
                line = f.readline()
                if line == '':
                    break
                line = line.replace("\n","")
                tmp = line.split() # when without passing parameters python assumes split on space
                # if each url exists in nodes
                if tmp[0] in node and tmp[1] in node:
                    # get the index of the beginning and end of the edge
                    ind0 = node[tmp[0]]
                    ind1 = node[tmp[1]]
                    # append the edge as a tuple in the edges
                    edges.append((ind0,ind1))
    node_size = len(node) # the number of nodes
    att_size = len(att_list[0]) # the number of attributes
```

```
# create the proximity matrix and initialize all distances to 0
S = np.zeros((node_size,node_size))

# create the node attribute matrix and initialize all to 0
A = np.zeros((node_size,att_size))
# iterate over the edges and go to the proximity matrix
# and set distance from node 1 to node 2 to 1
# and from node 2 to node 1 to 1 (convert to undirected)
for i in range(len(edges)):
    S[edges[i][0],edges[i][1]] = 1
    S[edges[i][1],edges[i][0]] = 1

diag = 0
# for each row of the proximity matrix
for i in range(node_size):
    # count the self edges (diagonal elements)
    diag += S[i,i]
# get all edges (both directions and self-edges)
nonzeros = S.nonzero()
# Calculate number of edges
# adding the number of node indexes (rows) that are non-zero (starts of edges)
# to the number of diagonal elements and divides by two because matrix is symmetric
# so does not double count edges
edge_count = int((len(nonzeros[0]) + diag)/2)
# prints the number
print ("number of edges : " + str(edge_count))

# iterates over the attributes of each node
# len(att_list) gives the number of rows and len(att_list[0]) gives the
# number of columns, rows are nodes and columns are attributes
for i in range(len(att_list)):
    for j in range(len(att_list[0])):
        # fills the Attribute matrix (A)
        # with the attributes of each node converted to
        # floating point
        A[i,j] = float(att_list[i][j])
# A=A.tocsr()
# print("Maximum value of A:" + str(A.max(axis=1)))
# print("Maximum value of A:" + str(A.max(axis=0)))
# print("Minimum value of A:" + str(A.min(axis=1)))
```

```
    # print("Minimum value of A:" + str(A.min(axis=0)))

    S_pre,A_pre=preprocess(S,A)

    return S_pre, S, A_pre, clus, A #scaling S and A

def preprocess(S,A):
    # initialization in the paper(JWNMF)
    # S = S / S.sum()
    # A = A / A.sum()

    # Normalization based on size of S
    # A = A * S.sum() / A.sum()

    # Sums over all the elements of A and S
    # and normalized matrix using the following formula
    S = S * A.sum() / S.sum()
    return S, A
```

Α′.2 `init_kmeans`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from sklearn.cluster import KMeans
import argparse
import build_graph
import numpy as np
import csv
import evaluate

def clustering(A,k):
    # initializes cluster list as empty
    kmeans_clus=[]
    # for each cluster adds and empty list
    # to the clusters
    for i in range(k):
        kmeans_clus.append([])
########### regularization for input attributes
    # for each colums of attribute matrix (attribute)
    # get the max value and divide all values with the max
```

```
    for i in range(A.shape[1]): # A.shape gives the num_rows and num_cols of A - index 1 is
        ↪ columns
        max_att = max(A[:,i])
        if max_att != 0:
            A[:,i] = A[:,i]/max_att

########## Learning step of kmeans
    kmeans = KMeans(n_clusters=k).fit(A)
    pred = kmeans.labels_
    centroids = kmeans.cluster_centers_
    # for each node
    for i in range(len(pred)):
        # go to the cluster center assignment list and append the nodes assigned to each cluster
        kmeans_clus[pred[i]].append(i)
    return pred, centroids, kmeans_clus

def initialize_U(A, centroids):
    # create matrix U with rows equal to nodes and columns equal to the number of clusters
    # and initialize all values to zero
    U = np.zeros((len(A),len(centroids)))
    # for each node
    for i in range(U.shape[0]):
        dis_list = []
        # for each cluster center
        for j in range(U.shape[1]):
            # append to the distance list the norm (Euclidean distance) of the node attributes to the
                ↪ centroid
            dis_list.append(np.linalg.norm(A[i]-centroids[j]))
        # for each cluster center
        for j in range(U.shape[1]):
            # assign the following value to the (node,centroid) location in the U matrix
            U[i,j]= (sum(dis_list)-dis_list[j]) / sum(dis_list)
        # convert the cluster assignments to probability
        U[i,:] = U[i,:] / sum(U[i,:])
    return U

def init_kmeans(k,data):
    print(data)

    path = "data/"+data
```

```
S, S_ori, A, true_clus, flag, A_ori = build_graph.build_graph(path)
# convert the list of true clusters to a set (unique ocurrence of each cluster)
# and then back to a list
clus_list = list(set(true_clus))
print(clus_list)
# initializes a cluster dictionary
# and assign to each cluster a unique index
clus_dic = {}
for i in range(len(clus_list)):
    clus_dic[clus_list[i]] = i
# replace each true cluster by its index
for i in range(len(true_clus)):
    true_clus[i] = clus_dic[true_clus[i]]

# Initialize all these empty lists
pred_l=[];cent_l=[];km_l=[]
mod=[];ent=[];nmi=[];ari=[]
# repeates experiment 5 times
for j in range(5): # for i=0 to 4
    # get the predicted assignment of each node, the centroids and the
    # nodes assigned to each centroid
    pred, centroids, kmeans_clus = clustering(A_ori,k)
    # append the predicted assignments of the run to the pred_l list
    pred_l.append(pred)
    # append the cetroids of the run to the cent_l list
    cent_l.append(centroids)
    # append the nodes assigned to each cluster for the run to the km_l list
    km_l.append(kmeans_clus)
    # calculate ARI score for the run and append to ari list
    ari.append(evaluate.ARI(true_clus,pred))
# sorts the ari score vector, get the middle (3rd element) and gets the index (location)
# of the element in the original ari score array
ind = ari.index(sorted(ari)[2])
# gets the predicted cluster assignments for the above index (run)
pred=pred_l[ind]
# gets the centroids of the above index (run)
centroids=cent_l[ind]
# gets the nodes assigned to each cluster for the above index (run)
kmeans_clus=km_l[ind]
# creates U matrix
```

```
    U = initialize_U(A_ori, centroids)
    # writes the contents of the U matrix to the corresponding csv files
    f = open('initialize/'+data+'_U_'+str(k)+'.csv','w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(U)
    f.close()
    # writes the contents of the centroids list to the corresponding file
    f = open('initialize/'+data+'_V_'+str(k)+'.csv','w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(centroids)
    f.close()
```

Α΄.3 `init_agglomerative`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


from sklearn.cluster import AgglomerativeClustering

import argparse
import build_graph
import numpy as np
import csv
import evaluate
from typing import Dict


def clustering(A, k, linkage):
    # initializes cluster list as empty
    clus = []
    # for each cluster adds and empty list3
    # to the clusters
    for i in range(k):
        clus.append([])
    ########### regularization for input attributes
    # for each colums of attribute matrix (attribute)
    # get the max value and divide all values with the max
    for i in range(A.shape[1]): # A.shape gives the num_rows and num_cols of A - index 1 is
    ↪ columns
        max_att = max(A[:, i])
```

```
        if max_att != 0:
            A[:, i] = A[:, i] / max_att

    ########## Learning step of clustering method
    model = AgglomerativeClustering(n_clusters=k, linkage=linkage).fit(A)
    pred = model.labels_
    centroids = np.zeros((k, A.shape[1]))
    frequencies = {}
    for i in range(pred.shape[0]):
        centroids[pred[i], :] = np.add(centroids[pred[i], :], A[i, :])
        if pred[i] in frequencies:
            frequencies[pred[i]] += 1
        else:
            frequencies[pred[i]] = 1
    for i in range(k):
        centroids[i, :] = centroids[i, :] / frequencies[i]

    # for each node
    for i in range(len(pred)):
        # go to the cluster center assignment list and append the nodes assigned to each cluster
        clus[pred[i]].append(i)
    return pred, centroids, clus


def initialize_U(A, centroids):
    # create matrix U with rows equal to nodes and columns equal to the number of clusters
    # and initialize all values to zero
    U = np.zeros((len(A), len(centroids)))
    # for each node
    for i in range(U.shape[0]):
        dis_list = []
        # for each cluster center
        for j in range(U.shape[1]):
            # append to the distance list the norm (Euclidean distance) of the node attributes to the
            ↪ centroid
            dis_list.append(np.linalg.norm(A[i] - centroids[j]))
        # for each cluster center
        for j in range(U.shape[1]):
            # assign the following value to the (node,centroid) location in the U matrix
            U[i, j] = (sum(dis_list) - dis_list[j]) / sum(dis_list)
```

```
        # convert the cluster assignments to probability
        U[i, :] = U[i, :] / sum(U[i, :])
    return U


def init_agglomerative(k, data, linkage: str = 'complete'):
    print(data)

    path = "data/" + data
    S, S_ori, A, true_clus, flag, A_ori = build_graph.build_graph(path)
    # convert the list of true clusters to a set (unique ocurrence of each cluster)
    # and then back to a list
    clus_list = list(set(true_clus))
    print(clus_list)
    # initializes a cluster dictionary
    # and assign to each cluster a unique index
    clus_dic = {}
    for i in range(len(clus_list)):
        clus_dic[clus_list[i]] = i
    # replace each true cluster by its index
    for i in range(len(true_clus)):
        true_clus[i] = clus_dic[true_clus[i]]

    # Initialize all these empty lists
    pred_l = []
    cent_l = []
    km_l = []
    mod = []
    ent = []
    nmi = []
    ari = []
    # repeates experiment 5 times
    for j in range(5): # for i=0 to 4
        # get the predicted assignment of each node, the centroids and the
        # nodes assigned to each centroid
        pred, centroids, clusters = clustering(A_ori, k, linkage)
        # append the predicted assignments of the run to the pred_l list
        pred_l.append(pred)
        # append the cetroids of the run to the cent_l list
        cent_l.append(centroids)
```

```
        # append the nodes assigned to each cluster for the run to the km_l list
        km_l.append(clusters)
        # calculate ARI score for the run and append to ari list
        ari.append(evaluate.ARI(true_clus, pred))
    # sorts the ari score vector, get the middle (3rd element) and gets the index (location)
    # of the element in the original ari score array
    ind = ari.index(sorted(ari)[2])
    # gets the predicted cluster assignments for the above index (run)
    pred = pred_l[ind]
    # gets the centroids of the above index (run)
    centroids = cent_l[ind]
    # gets the nodes assigned to each cluster for the above index (run)
    kmeans_clus = km_l[ind]
    # creates U matrix
    U = initialize_U(A_ori, centroids)
    # writes the contents of the U matrix to the corresponding csv files
    f = open('initialize/' + data + '_U_' + str(k) + '.csv', 'w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(U)
    f.close()
    # writes the contents of the centroids list to the corresponding file
    f = open('initialize/' + data + '_V_' + str(k) + '.csv', 'w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(centroids)
    f.close()
```

Α′.4 `init_spectral`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from sklearn.cluster import SpectralClustering
import argparse
import build_graph
import numpy as np
import csv
import evaluate

def clustering(A, k):
    # initializes cluster list as empty
    clus = []
```

```
    # for each cluster adds and empty list3
    # to the clusters
    for i in range(k):
        clus.append([])
    ########### regularization for input attributes
    # for each colums of attribute matrix (attribute)
    # get the max value and divide all values with the max
    for i in range(A.shape[1]): # A.shape gives the num_rows and num_cols of A - index 1 is
        ↪ columns
        max_att = max(A[:, i])
        if max_att != 0:
            A[:, i] = A[:, i] / max_att

    ########### Learning step of clustering method
    model = SpectralClustering(n_clusters=k).fit(A)
    pred = model.labels_
    centroids = np.zeros((k, A.shape[1]))
    frequencies = {}
    for i in range(pred.shape[0]):
        centroids[pred[i], :] = np.add(centroids[pred[i], :], A[i, :])
        if pred[i] in frequencies:
            frequencies[pred[i]] += 1
        else:
            frequencies[pred[i]] = 1
    k_new = np.unique(pred)
    for i in range(len(k_new)):
        centroids[k_new[i], :] = centroids[k_new[i], :] / frequencies[k_new[i]]

    # for each node
    for i in range(len(pred)):
        # go to the cluster center assignment list and append the nodes assigned to each cluster
        clus[pred[i]].append(i)
    return pred, centroids, clus

def initialize_U(A, centroids):
    # create matrix U with rows equal to nodes and columns equal to the number of clusters
    # and initialize all values to zero
    U = np.zeros((len(A),len(centroids)))
    # for each node
    for i in range(U.shape[0]):
```

```
        dis_list = []
        # for each cluster center
        for j in range(U.shape[1]):
            # append to the distance list the norm (Euclidean distance) of the node attributes to the
                ↪ centroid
            dis_list.append(np.linalg.norm(A[i]−centroids[j]))
        # for each cluster center
        for j in range(U.shape[1]):
            # assign the following value to the (node,centroid) location in the U matrix
            U[i,j]= (sum(dis_list)−dis_list[j]) / sum(dis_list)
        # convert the cluster assignments to probability
        U[i,:] = U[i,:] / sum(U[i,:])
    return U

def init_spectral(k,data):
    print(data)

    path = ”data/”+data
    S, S_ori, A, true_clus, flag, A_ori = build_graph.build_graph(path)
    # convert the list of true clusters to a set (unique ocurrence of each cluster)
    # and then back to a list
    clus_list = list(set(true_clus))
    print(clus_list)
    # initializes a cluster dictionary
    # and assign to each cluster a unique index
    clus_dic = {}
    for i in range(len(clus_list)):
        clus_dic[clus_list[i]] = i
    # replace each true cluster by its index
    for i in range(len(true_clus)):
        true_clus[i] = clus_dic[true_clus[i]]

    # Initialize all these empty lists
    pred_l=[];cent_l=[];km_l=[]
    mod=[];ent=[];nmi=[];ari=[]
    # repeates experiment 5 times
    for j in range(5): # for i=0 to 4
        # get the predicted assignment of each node, the centroids and the
        # nodes assigned to each centroid
        pred, centroids, kmeans_clus = clustering(A_ori,k)
```

```
        # append the predicted assignments of the run to the pred_l list
        pred_l.append(pred)
        # append the cetroids of the run to the cent_l list
        cent_l.append(centroids)
        # append the nodes assigned to each cluster for the run to the km_l list
        km_l.append(kmeans_clus)
        # calculate ARI score for the run and append to ari list
        ari.append(evaluate.ARI(true_clus,pred))
    # sorts the ari score vector, get the middle (3rd element) and gets the index (location)
    # of the element in the original ari score array
    ind = ari.index(sorted(ari)[2])
    # gets the predicted cluster assignments for the above index (run)
    pred=pred_l[ind]
    # gets the centroids of the above index (run)
    centroids=cent_l[ind]
    # gets the nodes assigned to each cluster for the above index (run)
    kmeans_clus=km_l[ind]
    # creates U matrix
    U = initialize_U(A_ori, centroids)
    # writes the contents of the U matrix to the corresponding csv files
    f = open('initialize/'+data+'_U_'+str(k)+'.csv','w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(U)
    f.close()
    # writes the contents of the centroids list to the corresponding file
    f = open('initialize/'+data+'_V_'+str(k)+'.csv','w')
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(centroids)
    f.close()
```

Α′.5 NAGC_class

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
NJNMF with PU-learning is described in this program.
We recommend using kmeans initialization (init=1).
In that case, you run kmeans_init.py in advance.
"""
```

```
import time
import sys
import scipy.stats
import numpy as np
import scipy as sp
import VU_init
# import networkx as nx
# from matplotlib import pyplot
# max_iter=100 #number of iterations
b=0.0 #bias for sigmoid function

class NAGC:
    def __init__(self,k1,k2,lam,S,W,X,data,init=1,rho=0.5,max_iter=100):
        node_size = S.shape[0]
        att_size = X.shape[1]
        # H is a matrix between topic1 and topic2
        H = np.random.random((k1,k2))
        #create initialized matrices
        U,V = VU_init.VU_init(X,k1,k2,init,data)
        # a = (1.0+rho)/2
        # W = np.abs(S_ori-1)
        W_ = 1-W
        # general params
        self.S = S
        self.W = W
        self.W_ = W_
        self.X = X
        self.max_iter = max_iter
        self.U = U
        self.H = H
        self.V = V
        self.lam = lam
        self.rho = rho
    def fit_predict(self):
        def update_U(S,W,W_,X,lam,U,H,V,rho):
            fUH = sigmoid(U.dot(H))
            fdUH = dif_sig(U.dot(H))
            UUT = U.dot(U.transpose())
            U = U*((rho*2.0)*S.dot(U)+(lam*X.dot(V) * fdUH).dot(H.transpose()))/((2.0*rho)*(UUT
                ↪ *W).dot(U)+(2.0*(1.0-rho))*(UUT*W_).dot(U)+(lam*fUH.dot(V.transpose().dot(
```

```
        ↪ V)) * fdUH).dot(H.transpose()))
    # V = V*((a*2.0)*S.dot(V)+(lam*A.dot(U) * fdVT).dot(T.transpose()))/(((2.0*a)*(VVT*
        ↪ S_ori)+(2.0*(1.0-a))*(VVT*S_)).dot(V)+(lam*fVT.dot(U.transpose().dot(U)) *
        ↪ fdVT).dot(T.transpose()))
    return U
def update_U_woPU(S,X,lam,U,H,V):
    fUH = sigmoid(U.dot(H))
    fdUH = dif_sig(U.dot(H))
    U = U*(2.0*S.dot(U)+(lam*X.dot(V) * fdUH).dot(H.transpose()))/(2*U.dot(U.transpose().
        ↪ dot(U))+(lam*fUH.dot(V.transpose().dot(V)) * fdUH).dot(H.transpose()))
    return U
def update_V(S,X,U,H,V):
    # fVT = sigmoid(V.dot(T))
    return V*(X.transpose().dot(sigmoid(U.dot(H))))/(V.dot(sigmoid(U.dot(H).transpose()).
        ↪ dot(sigmoid(U.dot(H)))))
def update_H(S,X,U,H,V):
    fUH = sigmoid(U.dot(H))
    fdUH = dif_sig(U.dot(H))
    H = H*(U.transpose().dot(fdUH*(X.dot(V))))/(U.transpose().dot((fdUH*fUH).dot(V.
        ↪ transpose().dot(V))))
    # print("The dimensions of H matrix is: ")
    # print(H.shape[0])
    # print(H.shape[1])
    return H

# replaces values that are non-a-number e.g.divisions by 0
# with a small error value epsilon so that calculations can be peformed
def removing_nan(mat):
    nan_list = np.argwhere(np.isnan(mat))
    for i in nan_list:
        mat[i[0],i[1]]=sys.float_info.epsilon
    return mat

def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-(x-b)))
def dif_sig(x):
    return 1.0 * np.exp(-(x-b)) / pow(1.0+np.exp(-(x-b)),2)

# FOR RELU ACTIVATION UNCOMMENT THESE AND COMMENT THE ABOVE
# def sigmoid(x):
```

```
        # return np.maximum(x, 0)
        #
        # def dif_sig(x):
        # return (x > 0).astype(int)

        start = time.time() #memo start time
        # learning step
        count = 0
        while 1:
            count += 1
            # print loss_function(S,V,U,Z,A,T,lam)
            if self.rho == 0.5:
                self.U = removing_nan(update_U_woPU(self.S,self.X,self.lam,self.U,self.H,self.V))
            else:
                self.U = removing_nan(update_U(self.S,self.W,self.W_,self.X,self.lam,self.U,self.H,self.
                    ↪ V,self.rho))
            self.V = removing_nan(update_V(self.S,self.X,self.U,self.H,self.V))
            self.H = removing_nan(update_H(self.S,self.X,self.U,self.H,self.V))
            if count>=self.max_iter:
                break
        elapsed_time = time.time() - start #measure elapsed time
        print (("optimizing_time:{0}".format(elapsed_time)) + "[sec]")
        return self.U, elapsed_time
```

Α′.6 VU_init

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import numpy as np
import random

def VU_init(A,k1,k2,flag,data):
    # set n to the number of rows of Attribute matrix (nodes)
    # set m to number of columns of attribute matrix (attributes)
    n = A.shape[0]
    m = A.shape[1]
    # if there is no ground truth initialize matrices randomly
    if flag == 0:
        U = np.random.random((n,k1))
        V = np.random.random((m,k2))
    else:
```

```
    # Create a string with path to the U data for k1 number
    infile = 'initialize/'+data+'_U_'+str(k1)+'.csv'
    # create an empty U list
    U = []
    # open the file
    with open(infile) as f:
        while True:
            row = []
            # read a line from the file
            line = f.readline()
            if line == '':
                break
            # strip the newline character from the string (line)
            line = line.rstrip('\n')
            # split the string on commas (separate values)
            tmp = line.split(',')
            # add each value to the row list
            for i in tmp:
                row.append(float(i))
            # add the row to the U list
            U.append(row)
    infile = 'initialize/'+data+'_V_'+str(k2)+'.csv'
    V = []
    with open(infile) as f:
        while True:
            row = []
            line = f.readline()
            if line == '':
                break
            line = line.rstrip('\n')
            tmp = line.split(',')
            for i in tmp:
                row.append(float(i))
            V.append(row)
    # convert the U and V lists to numpy arrays
    # to perform matrix calculations
    U = np.array(U)
    V = np.array(V).transpose()
return U,V
```

Α′.7 evaluate

```
#!/usr/bin/env python
# −∗− coding: utf−8 −∗−
# this program include calculater of modularity, entropy, NMI and ARI
import sys
import scipy.stats
import numpy as np
import scipy as sp
import scipy.io
from scipy.sparse import lil_matrix, csr_matrix
import math
from sklearn.metrics.cluster import adjusted_rand_score

def evaluate(mod,ent,spl,nmi,ari,true_clus,clus,pred,S,A,ent_flag,with_gt):
    k = len(clus)
    mod.append(cal_modularity(clus,S,k))
    if ent_flag == 0:
        ent.append(cal_entropy(clus,A,k))
    elif ent_flag == 1:
        ent.append(cal_dif_entropy(clus,A_copy,k))
        spl.append(split_entropy(clus,A_ori,k))
    if with_gt >= 3:
        nmi.append(cal_nmi(true_clus,clus))
        ari.append(ARI(true_clus,pred))
    return mod,ent,spl,nmi,ari
# this function calculates modularity from clustering result
def cal_modularity(clus,S,k):
    a = [0]∗k
    e = np.zeros((k,k))
    for i in range(k):
        for j in range(k):
            for l in clus[i]:
                for m in clus[j]:
                    e[i][j]+=float(S[l,m])
    # regularize e on the sum of S
    e = e / S.sum()
    for i in range(k):
        a[i] = sum(e[i][:])
    Q=0
```

```
    for i in range(k):
        Q+=e[i][i]-a[i]*a[i]
    return Q

def cal_entropy(clus,A,k):
    E=0
    for t in range(A.shape[1]):
        for j in range(k):
            if len(clus[j])==0:
                continue
            att=0
            for n in clus[j]:
                if A[n,t]==0:
                    att+=1
            pk=[1.0*att/len(clus[j]),1.0-1.0*att/len(clus[j])]
            # print pk
            E += len(clus[j]) * scipy.stats.entropy(pk)
    return E/A.shape[0]/A.shape[1]

def cal_nmi(true_clus, clus):
    t_label_list = list(set(true_clus))
    n_h=[]
    for i in range(len(t_label_list)):
        n_h.append(true_clus.count(i))
    t_clus_ind = [[] for i in range(len(n_h))]
    for i in range(len(true_clus)):
        t_clus_ind[true_clus[i]].append(i)
    n_l = []
    for i in range(len(clus)):
        n_l.append(len(clus[i]))

########## print No. labels ############
    print("estimate_No.label_:_[_",end="")
    for i in range(len(n_l)):
        print(str(n_l[i])+"_",end="")
    print(']')
#######################################
    n=sum(n_l)
    nmi = 0
    for h in range(len(n_h)):
```

```
        for l in range(len(n_l)):
            n_hl = len(list(set(t_clus_ind[h]) & set(clus[l])))
            if n_hl != 0 and n_h[h] != 0 and n_l[l] != 0:
                nmi += n_hl * math.log(1.0*n*n_hl/n_h[h]/n_l[l],2)
    deno_h = 0
    for h in range(len(n_h)):
        if n_h[h] != 0:
            deno_h += n_h[h] * math.log(1.0*n_h[h]/n,2)
    deno_l = 0
    for l in range(len(n_l)):
        if n_l[l] != 0:
            deno_l += n_l[l] * math.log(1.0*n_l[l]/n,2)
    if deno_h*deno_l != 0.0:
        nmi = nmi/math.sqrt(deno_h*deno_l)
    else:
        nmi = 0.0
    return nmi

def ARI(true_clus,clus):
    return adjusted_rand_score(true_clus,clus)
```

Α′.8 main

```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 16 17:17:54 2021

@author: vaso
"""
import os
import NAGC_class
import NAGC_relu
import build_graph
import evaluate
import init_kmeans
from init_agglomerative import init_agglomerative

# The directory that the results will be saved in
result_path = os.path.join(os.getcwd(), "Results")

# if the result directory does not exist create it
```

```
if not os.path.exists(result_path):
    os.mkdir(result_path)

# The datasets used in the experiments
datasets = ["citeseer", "WebKB_univ", "citeseer", "polblog", "cora"]

# For each dataset
for data in datasets:
    # create a results dir for the dataset
    res_folder = os.path.join(result_path, data)
    if not os.path.exists(res_folder):
        os.mkdir(res_folder)
    data_path = "data/" + data
    # load the dataset and build the graph
    S, W, X, true_clus, flag, A_ori = build_graph.build_graph(data_path)

    # The clustering methods to use
    clustering_method = [ "kmeans","agglomerative"]
    # The linkages (applicable in agglomerative only)
    linkages = ["complete", 'average', 'single']
    # The k2 parameter values used in the experiments
    k_2s = [2, 4, 5, 6, 7, 10, 15,20]
    # The lambda values used in the experiments
    lambda_values = [0, 1e-10, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10]
    # The rho values used in the experiments
    rho_values = [0.5, 0.55, 0.75, 0.95, 0.995]
    # The non-linear activation functions used in the experiments
    activations = ['sigmoid', 'relu']
    # Initialize the k1 parameter based on the values proposed by Makoto Onizuka et al., 2018
    if data == "WebKB_univ":
        k1 = 4
    elif data == "citeseer":
        k1 = 6
    elif data == "polblog":
        k1 = 2
    else:
        k1 = 7
    # For each clustering method, open the corresponding results CSV file,
    # run all the experiments and output the parameters, runtime and metrics (ARI, entropy,
    ↪ modularity)
```

```
for method in clustering_method:
    if method == "kmeans":
        res_file = open(os.path.join(res_folder, "res" + "_" + method + ".csv"), "a+")
        res_file.write("MODULARITY;ENTROPY;ARI;K2;LAMBDA;RHO;TIME;ACTIVATION\
            ↪ n")
        for k2 in k_2s:
            for lam in lambda_values:
                for rho in rho_values:
                    for activation in activations:
                        init_kmeans.init_kmeans(k1, data)
                        init_kmeans.init_kmeans(k2, data)

                        iteration = 100 #100 number of iterations
                        init = 1 # 0: random initialization, 1: kmeans initialization
                        if activation == "sigmoid":
                            NAGC = NAGC_class.NAGC(k1, k2, lam, S, W, X, data, init, rho)
                        else:
                            NAGC = NAGC_relu.NAGC(k1, k2, lam, S, W, X, data, init, rho)
                        U, elapsed_time = NAGC.fit_predict()

                        k = k1
                        clus = []
                        for i in range(U.shape[1]):
                            clus.append([])
                        pred = U.argmax(1)
                        for i in range(U.shape[0]):
                            clus[pred[i]].append(i)

                        modularity = evaluate.cal_modularity(clus, S, k)
                        entropy = evaluate.cal_entropy(clus, X, k)
                        ari = evaluate.ARI(true_clus, pred) if true_clus else 0.0
                        print("modularity: " + str(evaluate.cal_modularity(clus, S, k)))
                        print("entropy: " + str(evaluate.cal_entropy(clus, X, k)))
                        if true_clus:
                            print("ARI: " + str(evaluate.ARI(true_clus, pred)))

                        res_file.write("{:.4f};".format(modularity))
                        res_file.write("{:.4f};".format(entropy))
                        res_file.write("{:.4f};".format(ari))
                        res_file.write(str(k2) + ";")
```

```
                    res_file.write("{:.4f};".format(lam))
                    res_file.write("{:.4f};".format(rho))
                    res_file.write("{:.4f};".format(elapsed_time))
                    res_file.write(activation + "\n")
    res_file.close()
    print ("Experiments Complete.")
else:
    res_file = open(os.path.join(res_folder, "res" + "_" + method + ".csv"), "a+")
    res_file.write("MODULARITY;ENTROPY;ARI;K2;LAMBDA;RHO;LINKAGE;TIME;
        ↪ ACTIVATION\n")
    for k2 in k_2s:
        for lam in lambda_values:
            for rho in rho_values:
                for activation in activations:
                    for linkage in linkages:

                        init_agglomerative(k1, data, linkage=linkage)
                        init_agglomerative(k2, data, linkage=linkage)
                        iteration = 100 # number of iterations
                        init = 1 # 0: random initialization, 1: kmeans initialization
                        if activation == "sigmoid":
                            NAGC = NAGC_class.NAGC(k1, k2, lam, S, W, X, data, init, rho
                                ↪ )
                        else:
                            NAGC = NAGC_relu.NAGC(k1, k2, lam, S, W, X, data, init, rho)
                        U, elapsed_time = NAGC.fit_predict()

                        k = k1
                        clus = []
                        for i in range(U.shape[1]):
                            clus.append([])
                        pred = U.argmax(1)
                        for i in range(U.shape[0]):
                            clus[pred[i]].append(i)

                        modularity = evaluate.cal_modularity(clus, S, k)
                        entropy = evaluate.cal_entropy(clus, X, k)
                        ari = evaluate.ARI(true_clus, pred) if true_clus else 0.0
                        print("modularity: " + str(evaluate.cal_modularity(clus, S, k)))
                        print("entropy: " + str(evaluate.cal_entropy(clus, X, k)))
```

```
                    if true_clus:
                        print("ARI: " + str(evaluate.ARI(true_clus, pred)))
                    res_file.write("{:.4f};".format(modularity))
                    res_file.write("{:.4f};".format(entropy))
                    res_file.write("{:.4f};".format(ari))
                    res_file.write(str(k2) + ";")
                    res_file.write("{:.4f};".format(lam))
                    res_file.write("{:.4f};".format(rho))
                    res_file.write(linkage + ";")
                    res_file.write("{:.4f};".format(elapsed_time))
                    res_file.write(activation + "\n")
res_file.close()
print("Experiments Completed.")
```

Παράρτημα Β′

Αποτελέσματα (.csv)

Β′.1 ARI

Β′.1.1 citeseer

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.5456	0.0388	0.319	15	10.0	0.5	complete	11.5561	relu
0.722	0.039	0.2562	20	10.0	0.95	single	30.6724	sigmoid
0.7182	0.039	0.2537	10	10.0	0.95	complete	30.8594	sigmoid
0.3596	0.0391	0.2388	15	10.0	0.5	average	11.3811	relu
0.697	0.0392	0.209	15	0.1	0.95	average	29.9056	sigmoid
0.3318	0.0391	0.2023	15	10.0	0.5	single	11.5447	relu

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.7428	0.0388	0.3397	15	10.0	0.995	30.307	relu
0.7467	0.0388	0.3088	10	0.0	0.995	32.6349	sigmoid

Β′.1.2 cora

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.7329	0.0544	0.3962	20	1.0	0.95	average	16.6384	sigmoid
0.6468	0.0545	0.3562	7	10.0	0.95	complete	14.6374	relu
0.589	0.0552	0.315	15	0.1	0.5	complete	5.0065	sigmoid
0.7267	0.0547	0.3057	7	0.01	0.95	average	14.8083	relu
0.6754	0.0549	0.3053	20	10.0	0.95	single	16.3782	sigmoid
0.6956	0.055	0.2712	6	0.1	0.75	single	15.6576	relu

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.743	0.0544	0.3847	2	0.01	0.95	15.1884	sigmoid
0.7444	0.0544	0.3697	6	0.1	0.95	15.4658	relu

Β′.1.3 polblog

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.4241	0.3779	0.6376	6	0.0	0.95	complete	3.9719	sigmoid
0.4241	0.3779	0.6376	2	0.0	0.95	complete	3.2724	relu
0.4241	0.3764	0.6248	15	0.0	0.95	average	3.6532	sigmoid
0.4242	0.3737	0.5647	5	0.0	0.995	average	3.743	relu
0.4241	0.3719	0.5447	6	0.0001	0.95	single	3.9215	sigmoid
0.4241	0.3713	0.4982	10	0.001	0.95	single	3.1796	relu

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.4241	0.3779	0.6441	2	0.0	0.95	3.9206	sigmoid
0.4241	0.3779	0.6376	15	0.0	0.95	3.3766	relu

Β′.1.4 WebKB_univ

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.7378	0.1523	0.994	20	0.01	0.75	average	2.5673	sigmoid
0.736	0.1524	0.987	10	0.1	0.5	complete	1.1206	relu
0.7359	0.1524	0.9841	20	0.0001	0.55	single	2.3456	sigmoid
0.7359	0.1524	0.9841	15	1.0	0.75	complete	2.4007	sigmoid
0.7358	0.1524	0.9819	15	0.0	0.5	average	1.189	relu
0.7383	0.1524	0.978	15	0.001	0.5	single	1.1862	relu

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.7378	0.1523	0.994	20	0.001	0.5	1.5813	sigmoid
0.7349	0.1524	0.9783	10	0.1	0.55	2.1665	relu

Β′.2 MODULARITY

Β′.2.1 citeseer

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.7683	0.0391	0.1293	6	0.0	0.95	complete	26.0323	relu
0.7683	0.0391	0.1293	2	0.0	0.95	complete	24.6853	sigmoid
0.7581	0.0394	0.1024	10	0.01	0.995	average	29.0539	relu
0.7564	0.0394	0.0964	4	0.0	0.95	average	25.8211	sigmoid
0.7488	0.0393	0.0916	4	1.0	0.95	single	25.2582	relu
0.7467	0.0395	0.0965	2	0.0	0.95	single	24.6875	sigmoid

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.7755	0.0388	0.2416	7	0.0	0.95	27.3234	sigmoid
0.7745	0.0391	0.1963	5	0.0	0.95	26.9135	relu

Β′.2.2 cora

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.7391	0.0547	0.3108	4	1.0	0.95	complete	14.5662	relu
0.7346	0.0548	0.2332	7	0.1	0.95	single	14.5979	relu
0.7336	0.0548	0.2511	10	0.1	0.95	average	15.519	relu
0.7329	0.0544	0.3962	20	1.0	0.95	average	16.6384	sigmoid
0.7307	0.0548	0.1869	20	0.0	0.95	single	16.6367	sigmoid
0.7294	0.0549	0.1915	7	0.0	0.95	complete	16.0047	sigmoid

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.7481	0.0543	0.3704	7	0.0	0.95	16.8309	sigmoid
0.7478	0.0545	0.33	15	0.0001	0.95	18.0796	relu

Β′.2.3 polblog

res_agglomerative_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.4246	0.3716	0.4812	7	0.0001	0.5	single	0.3756	relu
0.4246	0.374	0.4701	15	0.0	0.5	single	0.7896	sigmoid
0.4245	0.3714	0.4812	4	0.0	0.75	complete	3.1818	relu
0.4245	0.3714	0.4812	10	0.0	0.75	average	3.9104	relu
0.4244	0.3718	0.4887	5	0.0	0.55	average	3.3483	sigmoid
0.4244	0.3764	0.6017	5	0.0	0.55	complete	3.8935	sigmoid

res_kmeans_res.csv

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.4245	0.3712	0.4794	2	0.0	0.75	4.0042	relu
0.4244	0.3713	0.4812	6	0.0	0.75	4.0871	sigmoid

Β′.2.4 `WebKB_univ`

`res_agglomerative_res.csv`

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	LINKAGE	TIME	ACTIVATION
0.741	0.153	0.9052	7	0.0	0.5	single	1.3427	relu
0.741	0.1529	0.9105	2	0.0001	0.5	average	0.6218	relu
0.7408	0.1529	0.9114	4	1.0	0.75	complete	1.6444	relu
0.7403	0.1531	0.9052	4	0.0	0.5	average	0.7865	sigmoid
0.7402	0.1531	0.8979	4	0.0	0.5	single	0.7911	sigmoid
0.7383	0.1524	0.9832	20	1.0	0.5	complete	1.4332	sigmoid

`res_kmeans_res.csv`

MODULARITY	ENTROPY	ARI	K2	LAMBDA	RHO	TIME	ACTIVATION
0.741	0.153	0.902	2	0.01	0.75	1.6302	relu
0.7404	0.1531	0.898	10	0.0	0.5	1.3326	sigmoid